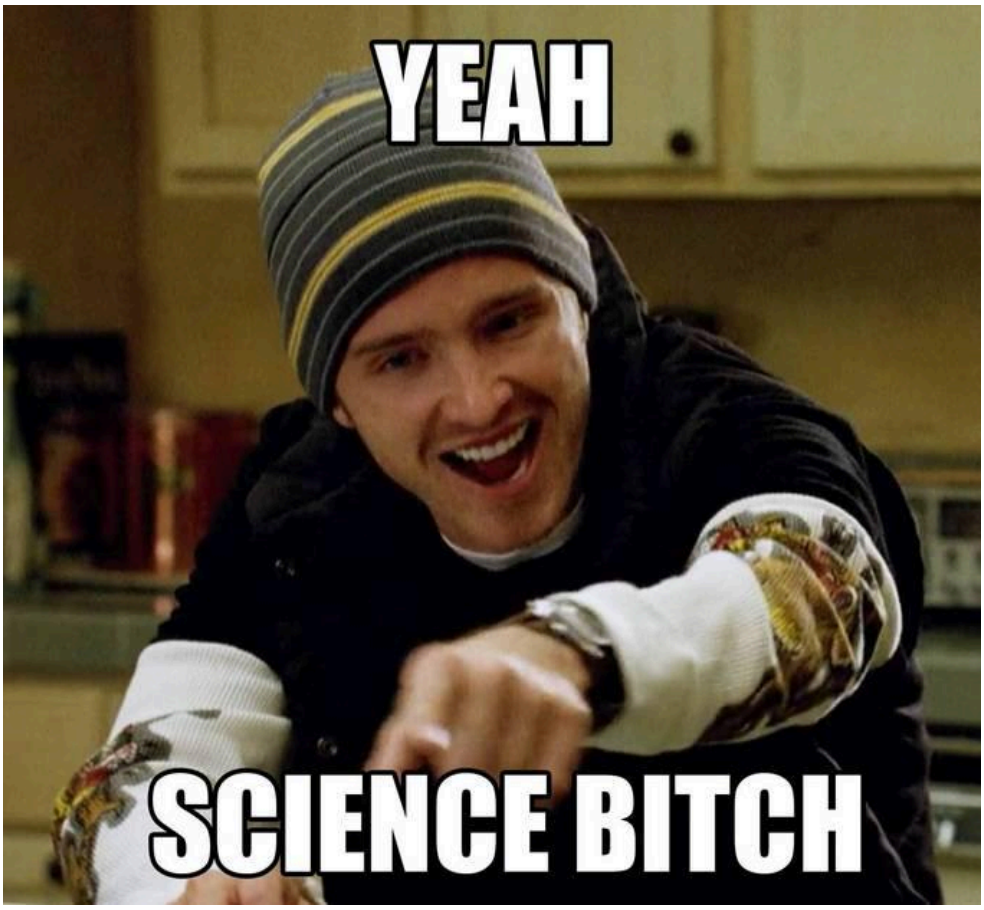
Christian Bartz,
Reducing the Annotation Burden:
Deep Learning for Optical Character Recognition using less Manual Annotations



"Yeah Science, Bitch" meme, Pass-A-Fist, 28.07.2012,
https://knowyourmeme.com/photos/517111-yeah-science-bitch (last accessed 31. August 2021)

Reducing the Annotation Burden: Deep Learning for Optical Character Recognition using less Manual Annotations

Christian Bartz

Dissertation
zur Erlangung des Doktorgrades der
Digital Engineering Fakultät
der Universität Potsdam

Betreuer: Prof. Dr. Christoph Meinel,
Universität Potsdam,
Faculty of Digital Engineering,
Internet Technologies and Systems

Gutachter: Prof. Dr. Hans Siegfried Stiehl,
Universität Hamburg,
Department of Informatics

Prof. Dr.Robert Sablatnig,
TU Wien,
Faculty of Informatics,
Institute of Visual Computing & Human-Centered Technology,
Computer Vision Lab

Datum der Einreichung: 31. August 2021
Datum der Disputation: 22. April 2022

Published online on the
Publication Server of the University of Potsdam:
https://doi.org/10.25932/publishup-55540
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-555407

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertation selbst angefertigt und nur die im Literaturverzeichnis aufgeführten Quellen und Hilfsmittel verwendet habe.

Diese Dissertation oder Teile davon wurden nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht.

Ich versichere weiterhin, dass ich diese Arbeit oder eine andere Abhandlung nicht bei einer anderen Fakultät oder einer anderen Universität eingereicht habe.

Potsdam, den 31. August 2021

Christian Bartz

Abstract

Text is a ubiquitous entity in our world and daily life. We encounter it nearly everywhere in shops, on the street, or in our flats. Nowadays, more and more text is contained in digital images. These images are either taken using cameras, *e.g.*, smartphone cameras, or taken using scanning devices such as document scanners. The sheer amount of available data, *e.g.*, millions of images taken by Google Streetview, prohibits manual analysis and metadata extraction. Although much progress was made in the area of optical character recognition (OCR) for printed text in documents, broad areas of OCR are still not fully explored and hold many research challenges. With the mainstream usage of machine learning and especially deep learning, one of the most pressing problems is the availability and acquisition of annotated ground truth for the training of machine learning models because obtaining annotated training data using manual annotation mechanisms is time-consuming and costly.

In this thesis, we address of how we can reduce the costs of acquiring ground truth annotations for the application of state-of-the-art machine learning methods to optical character recognition pipelines. To this end, we investigate how we can reduce the annotation cost by using only a fraction of the typically required ground truth annotations, *e.g.*, for scene text recognition systems. We also investigate how we can use synthetic data to reduce the need of manual annotation work, *e.g.*, in the area of document analysis for archival material.

In the area of scene text recognition, we have developed a novel end-to-end scene text recognition system that can be trained using inexact supervision and shows competitive/state-of-the-art performance on standard benchmark datasets for scene text recognition. Our method consists of two independent neural networks, combined using spatial transformer networks. Both networks learn together to perform text localization and text recognition at the same time while only using annotations for the recognition task. We apply our model to end-to-end scene text recognition (meaning localization and recognition of words) and pure scene text recognition without any changes in the network architecture.

In the second part of this thesis, we introduce novel approaches for using and generating synthetic data to analyze handwriting in archival data. First, we propose a novel preprocessing method to determine whether a given document page contains any handwriting. We propose a novel data synthe-

sis strategy to train a classification model and show that our data synthesis strategy is viable by evaluating the trained model on real images from an archive. Second, we introduce the new analysis task of handwriting classification. Handwriting classification entails classifying a given handwritten word image into classes such as date, word, or number. Such an analysis step allows us to select the best fitting recognition model for subsequent text recognition; it also allows us to reason about the semantic content of a given document page without the need for fine-grained text recognition and further analysis steps, such as Named Entity Recognition. We show that our proposed approaches work well when trained on synthetic data. Further, we propose a flexible metric learning approach to allow zero-shot classification of classes unseen during the network's training. Last, we propose a novel data synthesis algorithm to train off-the-shelf pixel-wise semantic segmentation networks for documents. Our data synthesis pipeline is based on the famous StyleGAN architecture and can synthesize realistic document images with their corresponding segmentation annotation without the need for any annotated data!

Zusammenfassung

Text umgibt uns überall. Wir finden Text in allen Lebenslagen, z.B. in einem Geschäft, an Gebäuden, oder in unserer Wohnung. Viele dieser Textentitäten können heutzutage auch in digitalen Bildern gefunden werden, welche auf verschiedene Art und Weise erstellt werden können, z.B. mittels einer Kamera in einem Smartphone oder durch einen Dokumentenscanner. Die Anzahl verfügbarer digitaler Bilder, z.B. Millionen – wenn nicht Milliarden von Bildern – in Google Streetview, macht eine manuelle Analyse der Bilddaten unmöglich. Obwohl es im Gebiet der Optical Character Recognition (OCR) in den letzten Jahren viel Fortschritt gab, gibt es doch noch viele Bereiche, die noch nicht vollständig erforscht worden sind. Der immer zunehmende Einsatz von Methoden des maschinellen Lernens, insbesondere der Einsatz von Deep Learning Technologien, im Bereich der OCR, führt zu dem großen Problem der Verfügbarkeit von annotierten Trainingsdaten. Die Beschaffung annotierter Daten mittels manueller Annotation ist zeitintensiv und sehr teuer.

In dieser Arbeit zeigen wir neue Wege und Verfahren auf, wie das Problem der Beschaffung annotierter Daten für die Anwendung von modernsten Deep Learning Verfahren im Bereich der OCR gelöst werden könnte. Hierbei zeigen wir neue Verfahren in zwei Unterbereichen der OCR. Einerseits untersuchen wir, wie wir die Annotationskosten reduzieren könnten, indem wir inexakte Annotationen benutzen um z.B. die Kosten der Annotation von echten Daten im Bereich der Texterkennung aus natürlichen Bildern zu reduzieren. Dieses System wird mittels weak supervision trainiert und erreicht Ergebnisse, die auf dem Stand der Technik bzw. darüber liegen. Unsere Methode basiert auf zwei unabhängigen neuronalen Netzwerken, die mittels eines Spatial Transformers verbunden werden. Beide Netzwerke werden zusammen trainiert und lernen zusammen, wie Text gefunden und gelesen werden kann. Dabei nutzen wir aber nur Annotationen und Supervision für das Lesen (recognition) des Textes, nicht für die Textfindung. Wir zeigen weiterhin, dass unser System für eine Mehrzahl von Aufgaben im Bereich der Texterkennung aus natürlichen Bildern genutzt werden kann, ohne Veränderungen im Netzwerk vornehmen zu müssen.

Andererseits untersuchen wir, wie wir Verfahren zur Erstellung von synthetischen Daten benutzen können, um die Kosten und den Aufwand der manuellen Annotation zu verringern und zeigen Ergebnisse aus dem Bereich der Analyse von Handschrift in historischen Archivdokumenten. Zuerst prä-

sentieren wir ein System zur Erkennung, ob ein Bild überhaupt Handschrift enthält. Hier schlagen wir eine neue Datengenerierungsmethode vor. Die generierten Daten werden zum Training eines Klassifizierungsmodells genutzt. Unsere experimentellen Ergebnisse belegen, dass unsere Idee auch auf echten Daten aus einem Archiv eingesetzt werden kann. Als Zweites führen wir einen neuen Schritt in einer Dokumentenanalyseplattform ein: Handschriftklassifizierung. Hier ordnen wir Bilder einzelner handgeschriebener Wörter anhand ihrer visuellen Struktur in Klassen, wie Zahlen, Datumsangaben oder Wörter ein. Die Einführung dieses Analyseschrittes erlaubt es uns den besten Algorithmus für den nächsten Schritt, die eigentliche Handschrifterkennung, zu finden. Der Analyseschritt erlaubt es uns auch, bereits Aussagen über den semantischen Inhalt eines Dokumentes zu treffen, ohne weitere Analyseschritte, wie Named Entity Recognition, durchführen zu müssen. Wir zeigen, dass unser Ansatz sehr gut funktioniert, wenn er auf synthetischen Daten trainiert wird; wir zeigen weiterhin, dass unser Ansatz auch für zero-shot Klassifikation eingesetzt werden kann. Zum Schluss präsentieren wir ein neues Verfahren zur Generierung von Trainingsdaten für die pixelgenaue semantische Segmentierung in Bildern von Dokumenten. Unser Verfahren basiert auf der bekannten StyleGAN Architektur und ist in der Lage Bilder mit entsprechender Annotation automatisch zu generieren. Hierbei werden keine echten annotierten Daten benötigt und das Verfahren kann auf jeder Form von Dokumenten eingesetzt werden.

Acknowledgments

First, I would like to express my gratitude to Prof. Dr. Christoph Meinel, my supervisor at the Hasso Plattner Institute, for his inspiration and support. You have given me everything I could wish for during my research activities. Second, I wish to thank Dr. Haojin Yang for his ongoing support, guidance, and inspiration for many years, even before the time of my Ph.D. studies! I am glad that you asked me to do a Ph.D. and provided support at any time, even when we were geographically separated.

I also wish to thank all of our (former) team members: Dr. Cheng Wang, Dr. Xiaoyin Chen, Dr. Mina Rezaei, Joseph Bethge, Gonçalo Mordido, Ting Hu, Ziyun Li, and Hendrik Rätz; I enjoyed the discussions about ideas and research in general, learning with and from you, all your questions regarding the computing infrastructure, and the possibilities to provide all of you with tech support.

I further wish to thank all my friends at HPI who made the time so enjoyable and remarkable. Most of all, I wish to thank Johannes Wolf, Lukas Wagner, and Norman Kluge for all the great times we had!

Last but not least, I wish to express my gratitude towards my parents, who always believed in me and supported every decision of mine. Thank you for sharing my excitement and encouraging me to do my thing to get the most out of it!

Without any of you, I would never have been able to finish my thesis!

Thank you all! Vielen Dank! - Christian aka Bartzi

Contents

List of Figures

List of Tables

Acronyms

5CHPT Five Classes Handwritten and Printed Text

AdaIN Adaptive Instance Normalization

BatchNorm batch normalization
BCE binary cross entropy
BLSTM bidirectional long short-term memory

CNN convolutional neural network
CRF conditional random field
CUTE CUTE80

DSLR digital single-lens reflex camera

FSNS French Street Name Signs

GAN generative adversarial network
GroupNorm group normalization
GRU gated recurrent unit

HOG histogram of oriented gradient

IAMDB IAM Handwriting Database
IAM-HistDB IAM Historical Handwriting Database
IAMONDB IAM Online Handwriting Database
IC13 ICDAR 2013
IC15 ICDAR 2015
IC15-1811 ICDAR 2015-1811
IIIT5K IIIT5K-Words
IOU intersection over union

kNN k-nearest neighbors

llr log-likelihood ratio
LSTM long short-term memory

MLP multilayer perceptron
MSE mean squared error

MSER maximally stable extremal region

NER named entity recognition

OCR optical character recognition

ReLU rectified linear unit
RNN recurrent neural network

SGD stochastic gradient descent
SVHN StreetView House Number
SVM support vector machine
SVT Street View Text
SVTP Steet View Text Perspective
SWT stroke width transform

UI user interface

WPI Wildenstein Plattner Institute

Glossary

ACCURACY The accuracy is an evaluation metric that denotes the amount of correctly classified samples compared to the total amount of samples. The accuracy is calculated as: $\frac{\text{true positive+true negative}}{\text{true positive+true negative+false positive+false negative}}$.

ADAIN Adaptive Instance Normalization, a normalization method originally developed by Huang and Belongie [104] for arbitrary style transfer.

F1-SCORE The F1-Score is an evaluation metric that measures the harmonic mean of precision and recall, it is calculated as $2 \times \frac{\text{precision} \times \text{recall}}{\text{precision+recall}}$.

GPU Graphics Processing Unit - an electronic circuit specifically designed to accelerate creation and processing of images. Machine learning benefits from the possibility to execute computations of dot products and matrix multiplications in a massive-parallel way.

INTERSECTION OVER UNION (IOU) The intersection over union is a metric to measure the similarity and diversity of sample sets. In computer vision it is mostly used to measure the matching ratio of bounding boxes or contours. The IOU of two bounding boxes A and B is calculated as $\text{IOU} = \frac{\text{intersection}(A,B)}{\text{union}(A,B)}$.

MULTILAYER PERCEPTION A Neural Network consisting of at least one hidden layer.

NAMED ENTITY RECOGNITION Named Entity Recognition is a procedure to locate and classify named entities, *e.g.*, names, brands, *etc*. in unstructured texts.

NATURAL LANGUAGE PROCESSING A field of artificial intelligence that deals with the analysis of interactions between humans and computers in the field of language.

PCA Principal Component Analysis, a dimensionality reduction algorithm that makes it, *e.g.*, possible to visualize high-dimensional vectors in 2D or 3D.

PRECISION The Precision is an evaluation metric that denotes the amount of positive classifications, *i.e.*, the true positives that were correct. It is defined as: $\frac{\text{true positives}}{\text{false positives+true positives}}$.

RECALL The Recall is an evaluation metric that measures the ratio of positive samples that were identified by the classifier.
It is definded as: $\frac{\text{true positives}}{\text{true positives+false negatives}}$.

RESNET Residual Neural Network. A convolutional neural network architecture that allows the training of very deep neural networks, which was introduced by He *et al*. [94].

WILDENSTEIN PLATTNER INSTITUTE A private foundation dedicated to the compilation of digital catalogue raisonnés and archives.

1 Introduction

In this thesis, we present possible approaches to lower the costs of acquiring ground truth annotations for the application of state-of-the-art machine learning methods to optical character recognition pipelines as the central topic. Thus, main contributions revolve around the development of mechanisms to solve one of the most pressing problems in deep learning: the availability and cost of annotated training data. In this chapter, we will provide a short introduction into the research domain of optical character recognition (OCR) and the specific sub-tasks we deal with and provide solutions for. Furthermore, we will summarize our contributions and conclude the chapter with a structural overview of the thesis.

1.1 Motivation and Scope

Being able to read is one of the most important cultural achievements of humankind. The ability to read and understand information allows the passing of information through centuries and even millennia. Reading is a seemingly simple task for a human. Once trained to understand the basic shapes of a simple alphabet, such as the Latin alphabet, a child can learn to read in the first year of school or even before. When looking at the reading capabilities of machines and computers, we quickly see that the process of reading does not seem to be that simple. An image containing text is merely a matrix of numerical values with no semantic information to a modern computer. Thus, algorithms to extract semantic information from a given image need to be developed.

In the past decades, much research effort was put into the recognition of printed text, which led to the development of commercial and open-source OCR tools, such as ABBYY FineReader[1] or Tesseract OCR [214]. Later, the analysis of handwriting, especially in the online case was in the focus of research [54, 155, 156]. Following online handwriting recognition, the community tended to the more complicated tasks of offline handwriting recognition [122, 241] and scene text recognition [114, 141, 150].

With the upcoming and successes of deep learning in many computer vision related tasks, such as image classification [94, 136], object detection [194, 195],

[1] https://pdf.abbyy.com/ (last accessed August 31, 2021).

or image captioning [230, 247] the research focus in the area of OCR shifted more and more to the usage of deep learning methods [112, 208, 242]. The usage of deep learning has clear benefits over the usage of traditional manually engineered feature approaches, such as stroke width transform (SWT) [75], or maximally stable extremal regions (MSERs) [179]. Methods based on deep learning can automatically learn which features need to be extracted to fulfill the task. Learning which features to extract is one factor why methods based on deep learning tend to generalize to unseen data. Another factor is the usage of large-scale training datasets that could easily contain millions of documents, which contrasts earlier training datasets that commonly contained not more than a few hundred or thousand examples for training.

The main challenge in using large-scale training datasets is the acquisition of such training data. Clean images that fit the task at hand need to be gathered and manually annotated. Manual annotation is a costly and time-consuming process, *e.g.*, the annotation of ImageNet [71] took ca. 3 years using 49 000 workers on Amazon mechanical turk.[2] Drawing from these observations, we argue that there is a great need for other methods to acquire training data and the necessary annotations. One solution is to stop relying on fully supervised machine learning and instead develop methods that can work with incomplete knowledge in weakly-supervised settings. The benefit of using weak supervision is that, *e.g.*, a system to classify and localize objects does not need annotations for classification and localization but only annotations for classification. Only requiring classification annotations is of great benefit because it is much cheaper to obtain classification labels than to obtain precise localization annotations. Another solution is to use and develop algorithms to generate synthetic data. The benefit of synthetic data is the possibility for the user to precisely control the content of the generated content.

In this thesis, we focus on methods to reduce the costs related to the creation of ground truth annotations for the training of deep neural networks. Specifically, we focus on two use cases from the area of optical character recognition systems. We propose a novel approach for weakly-supervised end-to-end scene text recognition and novel approaches using synthetic data to analyze historical documents in archives.

1.1.1 Weak Supervision for Scene Text Recognition

Scene text recognition deals with the task of identifying and recognizing text in images containing uncontrolled natural scenes. Such images could, for instance, be produced by an individual with a smartphone. The analysis

[2] https://www.cs.princeton.edu/courses/archive/spr18/cos598B/slides/cos598b_7feb18_imagenet.pdf slide 12 (last accessed August 31, 2021)

of scene images is challenging because of their wild nature, meaning that lighting conditions, camera movement, and image content can not be known beforehand. Thus, it is necessary to create robust recognition systems. Here, methods based on deep learning can show their potential because deep models can take various degrading conditions into account if they were trained with sufficiently diverse data. However, here the problem of annotation costs arises.

The field of scene text recognition began to benefit from synthetic data very early. Especially methods that provide solutions for the isolated task of text recognition could easily benefit from synthetic recognition data produced by text rendering and distortion algorithms [112]. Later, the first synthetic data approaches for synthesizing full-scene text images were proposed [90].

However, the synthesis of realistic scene text images is still a complex problem and might not be applicable to all kinds of text. Related work shows that integrating examples of real text into the training with synthetic data boosts the performance of proposed methods significantly [140]. In the field of end-to-end scene text recognition, methods exist that do not need information about the locations of text to localize and read text in a given image. However, these methods are not able to report the location of the read text. Thus, we argue that it is necessary to develop methods that use less supervision and thus require less annotation effort, are able to report localization and recognition results, and achieve results similar to results reported in related work, even if it is possible to synthesize large amounts of data.

1.1.2 Synthetic Data for Archive Analysis

Archives contain wisdom, knowledge, and information gathered over decades or centuries. Thanks to digitization, it is possible to preserve archival material and also, in the long run, perform automated analysis supporting historical research. To this end, besides preservation, digitization alone does not add much value to the data. Further analysis of the raw scanned documents is necessary to extract their content, thus enabling further computerized analysis capabilities.

Especially in the field of archive analysis, accurate and extensive ground truth annotations are of utmost importance. However, the acquisition of accurate ground truth annotations is very costly and might even be to expensive and laborous for many archives because required experts might not be available. Thus, machine learning methods in the field of archive analysis can highly benefit from the availability usage of synthetic training data.

Another crucial observation is that a model trained on a set of historical documents might not be applicable to other historical documents because historical documents are heterogeneous, *i.e.*, western European documents

from the 15th century are incredibly different from documents of the same time from the east Asian regions, this is also true when comparing documents from the early 19th and 20th century to documents from the 15th century. Thus, we argue that methods that can synthesize novel ground truth annotations could be a viable way to enable a fast and reliable analysis of archival data. Only recently new methods to synthesize large amounts of historical data emerged [51, 126, 231]. However, recent methods require a large amount of manual preparations, *e.g.*, design of realistic document templates [231]. We focus on the development of mechanisms to produce and use synthetic training data with as little manual intervention as possible to enable the application and adaption of our proposed methods to a multitude of archives.

The development of our technologies for the analysis of historical document data is done in conjunction with and support of the Wildenstein Plattner Institute (WPI).[3] The WPI is a non-profit foundation that is dedicated to supporting research in the field of art history by compiling digitial catalog raisonnés and archives. The WPI has an archive consisting of more than 10 million document pages that are now entirely digitized. To enable further research, ease the exploration, and search for information, we develop and apply methods to extract handwritten information from the archive. Handwritten information is of particular interest to the researchers of the WPI because handwriting might contain valuable information about works of art that has been noted down at auctions or during talks with artists themselves, *i.e.*, information about the provenance of a work of art.

1.2 Contributions and Publications

In this thesis, we address the problem of obtaining annotated training data for the application of OCR methods on use cases in the areas of scene text recognition and archive analysis. First, we address the research question of how we can lower the annotation cost by using only a fraction of the typically required ground truth annotations, by introducing a novel method for end-to-end scene text recognition. Second, we address the research question of how we can use synthetic data to reduce the need of manual annotation work, *e.g.*, in the area of document analysis for archival material. The major contributions of this thesis can be summarized as follows:

Scene Text Recognition Our first use case to use less annotations is rooted in the field of scene text recognition. Here, we propose a new weakly supervised end-to-end method for scene text localization and recognition. Our method only requires textual annotations for the recognition task to

[3] https://wpi.art/ (last accessed August 31, 2021).

simultaneously learn to detect and recognize text. Our method consists of a two-stage deep neural network. The first stage is a localization network that predicts a set of transformation parameters. These transformation parameters describe the location of the text, *i.e.*, words or characters. We use the transformation parameters in a differentiable sampler [113] to extract the words or characters and forward them through our second network, the recognition network. We perform experiments on a range of standard scene text detection and recognition benchmarks showing that our method can learn to localize text regions in a weakly supervised manner. Similar to related work [215, 243] our method can be applied on training images without location annotations but in contrast to related work our method does not only extract the textual content of words in the images but also the location of the words in the image. Furthermore, our model achieves competitive and state-of-the-art results on several scene text recognition benchmark datasets while showing exceptional performance on irregular scene text recognition benchmark datasets. (related publications: [22, 31, 32, 249])

Archive Analysis Our second use case is the synthesis and application of synthetic training data for the training of machine learning systems for document analysis systems. Here, we show examples from the field of archive analysis where we propose several data synthesis strategies and new methods to ease and enable the large-scale analysis of handwriting data in historical archives. First, we propose a handcrafted data synthesis strategy to synthesize patches of documents. We use these patches to train a classification model to determine whether patches of a document image contain handwriting. Using the prediction for each patch of a page, we can determine whether a full page contains any handwriting at all. Our system stands in contrast to related work, *e.g.*, [62] because we solely use synthetic data for the training of our classification model. In our experiments on data extracted from a diverse set of documents from the archive of the WPI, we are able to achieve an F1-Score of 0.98 showing that our method reliably detects the presence of handwriting on a given page. (related publication: [29])

Second, we propose a new analysis step for a document analysis pipeline. This analysis step is located close to the end of a document analysis pipeline and is used to categorize already cropped word images into classes such as date, number, or word based only on their visual appearance, which we denote as handwriting classification. Such a classification task can be of benefit in two ways. On the one hand, we can use the classification results to identify the best fitting recognition model. On the other hand, more importantly, we can reason about the content of a document without

the need to precisely read the content, which is especially helpful if no recognition model for the document's language or the writer, in the case of handwritten data, is available. To this end, we propose two methods; one method based on a simple, fixed softmax classifier and another more flexible method based on metric learning. We also propose well-suited data synthesis methods for the task of handwriting classification. In our experiments, we validate our assumptions and show the capabilities of our models. We find that the softmax classifier is the best performing model. However, we also validate that our approach based on metric learning is very flexible as it can perform zero-shot classification by correctly identifying samples of a class unknown at training time when confronted with them at test time. (related publication: [27])
Last, we propose a novel data synthesis method using the inner knowledge of a generative adversarial network (GAN) to synthesize a large-scale fully-annotated pixel-wise semantic segmentation dataset to segment historical documents. Here, we investigate the capabilities of StyleGAN [129, 130] and find that it is possible to not only synthesize RGB images but also corresponding segmentation annotation images at the same time. We devise an algorithm to extract the information from StyleGAN during the synthesis of data. Using our approach, it is possible to quickly synthesize millions of training samples for a pixel-wise semantic segmentation network. In contrast to related work [126, 231], our method does not need manually designed document templates for the generation of document-like data, our proposed approach directly learns the document structure from the available raw scanned images. While the approach is still under active research, we show in a qualitative evaluation that it is viable. The main benefit of our proposed approach is that it is possible to directly learn a model that is fitted to the raw data while no annotations for each image are necessary. However, our current approach still requires human annotation effort, although we could reduce the necessary effort to a minimum. (related publication: [28])

Earlier versions of several parts of this thesis and work that did not directly lead to this thesis but might be related to it have been published and presented at international scientific conferences and workshops. In addition, the proposed system for handwriting determination (see section 4.3) is to be integrated into the document analysis pipeline of the WPI (ongoing effort). The list of publications to this thesis includes the following:

Conferences

- Haojin Yang, Cheng Wang, Christian Bartz, and Christoph Meinel. "SceneTextReg: A Real-Time Video OCR System". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pages 698–700. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2973811
- Christian Bartz, Haojin Yang, and Christoph Meinel. "SEE: Towards Semi-Supervised End-to-End Scene Text Recognition". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. Apr. 2018
- Christian Bartz, Laurenz Seidel, Duy-Hung Nguyen, Joseph Bethge, Haojin Yang, and Christoph Meinel. "Synthetic Data for the Analysis of Archival Documents: Handwriting Determination". In: *2020 Digital Image Computing: Techniques and Applications (DICTA)*. Nov. 2020, pages 1–8. DOI: 10.1109/DICTA51227.2020.9363410

Workshops

- Christian Bartz, Haojin Yang, Joseph Bethge, and Christoph Meinel. "LoANs: Weakly Supervised Object Detection with Localizer Assessor Networks". In: *Computer Vision – ACCV 2018 Workshops*. Edited by Gustavo Carneiro and Shaodi You. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pages 341–356. ISBN: 978-3-030-21074-8. DOI: 10.1007/978-3-030-21074-8_29
- Christian Bartz, Hendrik Rätz, and Christoph Meinel. "Handwriting Classification for the Analysis of Art-Historical Documents". In: *Pattern Recognition. ICPR International Workshops and Challenges*. Edited by Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pages 562–576. ISBN: 978-3-030-68796-0. DOI: 10.1007/978-3-030-68796-0_40

Technical Reports

- Christian Bartz, Haojin Yang, and Christoph Meinel. "STN-OCR: A Single Neural Network for Text Detection and Text Recognition". In: *arXiv:1707.08831 [cs]* (July 2017). arXiv: 1707.08831 [cs]
- Christian Bartz, Joseph Bethge, Haojin Yang, and Christoph Meinel. "KISS: Keeping It Simple for Scene Text Recognition". In: *arXiv:1911. 08400 [cs]* (Nov. 2019). arXiv: 1911.08400
- Christian Bartz, Hendrik Rätz, Haojin Yang, Joseph Bethge, and Christoph Meinel. "Synthesis in Style: Semantic Segmentation of Historical Documents Using Synthetic Data". In: *arXiv:2107.06777 [cs]* (July 2021). arXiv: 2107.06777 [cs]

Besides the publications that directly led to this thesis, we also published further work. We provide the complete list of all publications in the appendix (chapter 7).

1.3 Outline of the Thesis

We organize this thesis in the following manner:

After the introduction, presented in this chapter, we provide an introduction into the field of OCR and deep learning concepts used in this work in chapter 2. Here, we first introduce the task of OCR. Following the OCR introduction, we provide an overview and explanation of the machine learning concepts used in this work in section 2.2.

In chapter 3 we show our solution for the use case of weakly supervised end-to-end scene text recognition. There, we first provide a review of related work in the field of scene text detection and recognition in section 3.3. Following the related work, we introduce the datasets we use and also our data synthesis strategy to enhance the existing datasets in section 3.4 and section 3.5. In the last two sections of this chapter, we introduce our proposed methodology (section 3.6) and show the results of our extensive experiments (section 3.7) to validate our proposed method.

In chapter 4, we present our solutions utilizing synthetic data for the analysis of historical documents. First, we review related work in the area of historical document analysis (see section 4.2). In section 4.3 we introduce our solution for determining whether a page contains handwriting and validate our proposed method in experiments on a diverse dataset containing real data sampled from the archive of the WPI. Following our solution on handwriting determination, we introduce our approach to handwriting classification in section 4.4. Here, we first introduce the data synthesis methods we propose for this task, followed by our proposed methods for handwriting classification. Our experiments show the suitability of our proposed models and discuss usage scenarios for each proposed model. In section 4.5 we show our data synthesis approach that directly operates on raw scanned images to produce pixel-wise semantic segmentation ground truth data for the training of off-the-shelf semantic segmentation networks.

Last, in chapter 5, we conclude and summarize the achievements presented in this thesis. Further, we provide a brief outlook into possible future work, followed by references.

2 Foundations of Optical Character Recognition

Text is ubiquitous in our modern world. We can find it nearly everywhere, *e.g.*, in newspapers, books, documents, on billboards, signs, or the packaging of products. It is not feasible to use humans to analyze the available amount of textual content in scans, photos, or videos because human labor is expensive and slow compared to the computing speed of computers. An excellent example of this is the analysis of the Jeremy Bentham Collection in the "Transcribe Bentham" project, where crowdsourcing is used to recognize the handwritten text found in 95 000 scanned pages. The project started in 2010. The researchers expect the earliest date of complete transcription to be in 2025 [56]. A well-defined and working algorithm running on powerful computing hardware could be able to transcribe all pages in less than two days.[1] However, such algorithms are not available yet. Thus, further development of automated approaches for OCR is necessary. In this chapter, we shortly introduce the task and history of OCR, including typical components of OCR systems and research frontiers. Following the introduction of OCR, we further introduce basic concepts of modern computer vision technologies that we use in our work.

2.1 Optical Character Recognition

In ancient times it was only possible to transfer knowledge via verbal communication. Using only verbal communication and relying on the recipient's memory has the disadvantage of not remembering all details correctly. Writing and reading is a cultural accomplishment that allows transferring knowledge without the need to fear losing information if the recipient does not correctly remember all details of the information. Nowadays, text is ubiquitous in every modern society. We read books, newspapers, or magazines. We can also find text on any product packaging or signs. Reading texts is a seemingly simple task for a human. However, for a machine, it is a difficult task that is researched for more than a century. One of the first devices designed to read printed texts

[1] We, for instance, performed print OCR in a massively parallel way on a 1000 core computing cluster for more than 700 000 pages from the archive of the WPI in less than two days.

Figure 2.1: Standardized font faces for early OCR of printed text. *(left)* OCR-A,[2] *(right)* OCR-B[2]

was the Optophone by Dr. Edmund Fournier d'Albe in 1913 [70]. The device used photosensors to detect black print and converted the found black parts into characteristic sounds for each character, thus enabling a blind person to read a text.

In 1928, Emanuel Goldberg filed a patent for a machine that helped users to find specific content in microfilmed documents [217]. With the development of digital computers and first scanning devices, it was possible to process more and more data. Especially finance and bank-oriented use cases called for methods to automatically recognize printed text from letters, cheques, or forms. To enable early optical character recognition, specific fonts have been standardized.[3] These fonts had a specific and straightforward design that allowed the programming of machines to recognize text automatically. We provide examples of such fonts in figure 2.1.

However, already in 1970, the Scan-Data Corporation produced one of the first OCR machines that were able to recognize texts set in various fonts [174]. Throughout time software and digital imaging superseded hardware-based machines for optical character recognition, which led to the release of mature

[2] "OCR-A S" and "OCR-B S" (https://commons.wikimedia.org/wiki/File:OCR-A_SP.svg, https://commons.wikimedia.org/wiki/File:OCR-B_SP.svg) by GJo. CC BY 3.0 (https://creativecommons.org/licenses/by-sa/3.0/legalcode)

[3] ISO 1073-1:1976 (https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/00/55/5567.html) and ISO 1073-2:1976 (https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/00/55/5568.html) (last accessed August 31, 2021)

products such as "ABBYY FineReader"[4] or Tesseract [214]. Such software performs very well when recognizing printed text from a cleanly scanned page, *i.e.*, a binarized page where each pixel corresponding to printed text is black, and all other pixels are white. However, such software does not perform well on text found in natural scenes (scene text), *i.e.*, captured with the camera of a mobile phone or handwritten texts. These limitations call for further research.

2.1.1 Components of OCR Systems

An OCR system consists of multiple analysis stages until the user receives the recognition result. The work presented in this thesis contributes to different stages of a typical OCR pipeline [187]. In the following, we present typical components of an OCR pipeline and highlight the contributions of this thesis.

Optical Scanning The first step of an OCR system is optical scanning. Here, the document is converted into a digital format by using optical document scanners or cameras. In the case of OCR for printed or handwritten text, documents are mostly digitized using optical document scanners. In the case of scene text recognition, mostly digital cameras such as digital single-lens reflex cameras (DSLRs), digital camcorders, or smartphones can be used to capture images that might contain text.

Binarization and Segmentation State-of-the-art print OCR tools, *i.e.*, Tesseract [214], require the document images to be supplied in binarized form. Here, binarized refers to a document where each pixel representing text is black, and any other pixel is white. Such a binarization is necessary for these print OCR tools to find individual words on the scanned page correctly. While it might be a simple task to binarize modern office documents, it is challenging to binarize historical documents, typically found in archives (see figure 4.19 and figure 4.20 for some examples of challenging historical documents). Documents contained in archives do not only consist of challenging multi-colored backgrounds; they might also contain handwritten information that can not be recognized by a print OCR tool and hence leads to wrong recognition results. Thus, it is crucial to remove any content that is not printed text.

Localization Following the binarization step, regions that contain text are extracted. In the case of a document analysis pipeline, this step is also known as page segmentation. In page segmentation, components that belong together are identified, and either individual lines or words of these components are

[4] `https://www.abbyy.com` (last accessed August 31, 2021).

extracted. In scene text recognition, text localization plays a pivotal role, as the text is embedded into natural scenes.

Preprocessing In the next step, the identified text areas are preprocessed. The aim of preprocessing is to remove noise and also to transform the input into a normalized form. Such a normalized form makes it simpler for the underlying recognition model to produce the correct output. Normalization, in this case, includes rotation of rotated text, slant removal, and resizing to a fixed input size.

Feature Extraction The prepared image containing a full line of text, a group of words, or only a single word is used as input to a feature extraction algorithm. This step can be seen as one of the most challenging steps in OCR. The aim of feature extraction in OCR is to capture the representative traits of characters. The robustness of the feature extraction step thereby determines the overall performance of the OCR system and is therefore crucial. Such feature extraction can be performed by handcrafted algorithms or data-driven algorithms, such as neural networks.

Recognition Following the feature extraction step, the actual recognition takes place. Here, the system assigns the extracted features to a task-specific class, *e.g.*, the most probable character class for each identified character in a word image.

Post Processing The last step aims at the improvement of recognition results. On the one hand, individually recognized characters could be grouped into words or lines. On the other hand, we can use post-processing to correct recognition errors by correcting the recognized words using a dictionary and a spell checker.

In this Thesis In this thesis, we show contributions to the following OCR components:

Binarization Here, we propose a novel segmentation method (see section 4.5) for documents that are directly fitted to the available data without the need for a large-scale annotated training dataset.

Localization We propose a novel weakly supervised algorithm for simultaneous localization and recognition of scene text in chapter 3.

Preprocessing In section 4.3 we propose a method to determine whether a given document page contains handwriting. If the page contains handwriting, we also provide a rough location of the handwriting. Such information

can be beneficial when choosing the correct analysis algorithms for a given document scan.

Recognition In section 4.4, we propose a novel analysis step before the actual text recognition step that classifies a given word image based on its visual structure. Such a classification allows choosing the best fitting recognition model and allows the analysis of the content of documents without the need for perfect recognition results. In section 3.6, we propose a novel scene text recognition model that achieves state-of-the-art results on a range of open benchmark datasets.

2.1.2 Going Beyond Print OCR Systems

The components introduced in the last section are typical building blocks of a system such as Tesseract [214]. While state-of-the-art OCR systems report recognition accuracies of more than 99 %[5] these accuracies can only be achieved on optimal scans of documents.

Tesseract, for instance, is optimized to recognize text from documents that consist of black text on a white background, use standard fonts, and adhere to a simple layout with one or two columns. Tesseract cannot handle any handwritten text as it does not contain a module to recognize or localize handwritten text. Most documents found in archives do not adhere to modern document layouts, as they are heterogeneous in layout and content, show different signs of degradation, or contain mixtures of handwritten and printed text. Furthermore, print OCR systems fail in the task of recognizing text contained in natural scenes [125]. Thus, there is a need to develop novel approaches for text recognition in historical documents and text in natural scenes (scene text); see also figure 2.2 for a comparison of the text types mentioned here.

2.2 Neural Networks

While traditional OCR systems relied on the extraction of handcrafted features, *e.g.*, SIFT [159], SURF [33], as well as machine learning methods such as support vector machines (SVMs) [66], nowadays approaches based on neural networks are the dominant approaches in the area of computer vision. Neural networks became increasingly more popular thanks to the increase in available parallel computation power of modern GPUs and the availability of large-scale annotated datasets, such as the ImageNet dataset [71].

[5] https://paperlessocr.com/resources/flexicapture/flexicapture-faqs (last accessed August 31, 2021).

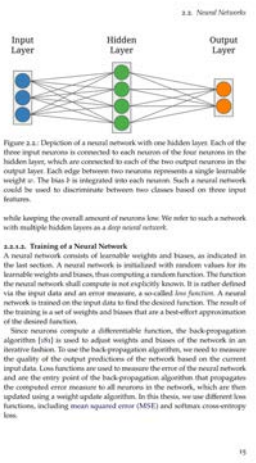


Figure 2.2: Different types of documents. The left image depicts a printed document that can easily be analyzed using standard OCR tools such as Tesseract [214]. The image in the middle displays a typical scene text image containing text as found in the real world, while the image on the right depicts an example archival image that contains a mixture of printed and handwritten text on a more challenging background.

The fact that neural networks became the dominant approach in the last few years is quite surprising because the computational model of a neural network is already known since the 1940s [168]. However, many complex problems led to several so-called AI winters. Neurons, for instance, are only able to compute linear functions. Furthermore, neural networks require a large amount of memory and computational resources. With the upcoming of general-purpose GPUs, the availability of large annotated datasets and the development of algorithms for the training of the first deep neural networks [34, 200] neural networks became an integral part of modern computer vision approaches.

In this section, we will shortly introduce the concept of neural networks, and we will specifically focus on the types of neural networks used throughout this thesis.

2.2.1 Basic Building Blocks of a Neural Network

Neural networks are loosely inspired by the inner workings of neurons in the mammalian brain [168]. A neural network consists of several building blocks, with the smallest unit being a single neuron. Multiple neurons that work together are called a neural network.

2.2.1.1 Neurons and Fully Connected Layers

As already stated above, the atomic unit of a neural network is a single neuron. A neural network consists of multiple neurons, where each neuron computes a weighted sum:

$$f(x, w, b) = w^T x + b. \tag{2.1}$$

Where $x \in \mathbb{R}^n$ is an input vector consisting of n inputs, $w \in \mathbb{R}^n$ is a vector of n learnable weights, and $b \in \mathbb{R}$ is a scalar bias that is added to the computed output of the weighted sum. Multiple neurons can be grouped into a single *layer*. If all neurons in a layer receive the same input, such a layer is called *fully connected layer*, and the weight vector w can be written as a weight matrix $w \in \mathbb{R}^{n \times m}$, where m denotes the number of neurons in the layer. The bias b is then similarly written as a vector with m dimensions as b^m.

Fully connected layers can further be stacked on top of each other and making it possible to compute a more complex function. Each layer of the network that is not directly involved in data input or output is called a hidden layer. Theoretically, an unlimited number of hidden layers may be used. A network consisting of at least one hidden layer is also called a *multilayer perceptron (MLP)*. Please refer to figure 2.3 for a structural overview of a MLP.

It was shown that MLPs with only one hidden layer are able to approximate any function mapping from one finite-dimensional space to another finite-dimensional space, with the desired accuracy if the number of hidden neurons is large enough [69, 101]. However, using only one hidden layer in a neural network makes it difficult to capture multiple inputs because the representations that can be learned with one hidden layer are only local. Thus, only a large amount of neurons, *i.e.*, a large number of dimensions, can capture all components. Hinton *et al*. [80] showed that stacking multiple hidden layers on top of each other enables the learning of more compact representations, allowing the network to learn representations of objects at different abstraction layers while keeping the overall amount of neurons low. We refer to such a network with multiple hidden layers as a *deep neural network*.

2.2.1.2 Training of a Neural Network

A neural network consists of learnable weights and biases, as indicated in the last section. A neural network is initialized with random values for its learnable weights and biases, thus computing a random function. The function the neural network shall compute is not explicitly known. It is rather defined via the input data and an error measure, a so-called *loss function*. A neural network is trained on the input data to find the desired function. The result of the training is a set of weights and biases that are a best-effort approximation of the desired function.

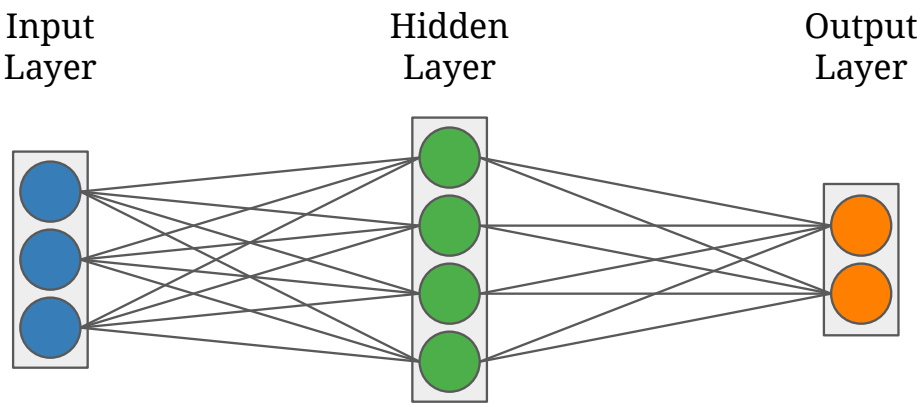


Figure 2.3: Depiction of a neural network with one hidden layer. Each of the three input neurons is connected to each neuron of the four neurons in the hidden layer, which are connected to each of the two output neurons in the output layer. Each edge between two neurons represents a single learnable weight w. The bias b is integrated into each neuron. Such a neural network could be used to discriminate between two classes based on three input features.

Since neurons compute a differentiable function, the back-propagation algorithm [198] is used to adjust weights and biases of the network in an iterative fashion. To use the back-propagation algorithm, we need to measure the quality of the output predictions of the network based on the current input data. Loss functions are used to measure the error of the neural network and are the entry point of the back-propagation algorithm that propagates the computed error measure to all neurons in the network, which are then updated using a weight update algorithm. In this thesis, we use different loss functions, including mean squared error (MSE) and softmax cross-entropy loss.

MSE is an error measure that measures the difference of a computed value to the expected value. It is mostly used for regression tasks, *i.e.*, prediction of text locations, where a specific value has to be computed. This loss function computes the mean of the squared difference of the predicted vector $y \in \mathbb{R}^n$ and the target vector $\hat{y} \in \mathbb{R}^n$:

$$\text{MSE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} (y - \hat{y})^2. \tag{2.2}$$

We use the softmax cross-entropy loss function for classification tasks, *i.e.*, recognizing characters in a word image. The softmax cross-entropy loss is based on the softmax function:

$$\text{softmax}(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{N} e^{y_j}}. \tag{2.3}$$

The softmax function transforms a vector $y \in \mathbb{R}^n$ into a vector $z \in \mathbb{R}^n$ that has the following properties: the value of each element of z lies in the interval $[0, 1]$, while the sum of all values of z is exactly 1. Thus, the output of the softmax function can be interpreted as a probability distribution ($P_{\text{model}}(y)$) over the components of the output vector y. The softmax cross-entropy loss ($\mathcal{L}$) is computed using the negative log probability of the desired output distribution $\hat{y}$:

$$\mathcal{L}(y, \hat{y}) = -\log P_{\text{model}}(\hat{y}|y). \tag{2.4}$$

After calculating the error measure, the derivative of the loss function with respect to the input is calculated, and the resulting gradient is propagated through the entire neural network using the back-propagation algorithm. The resulting gradients for weights and biases of each neuron denote the direction and strength of each desired parameter change to get a better approximation of the implicitly defined function the network shall learn. The update of each parameter is handled by the gradient descent algorithm [55]:

$$w_{n+1} = w_n - \gamma \nabla N(w_n). \tag{2.5}$$

With w being a specific weight of a neuron N at timestep n. γ denotes the global learning rate, *i.e.*, the rate at which the value of each weight is allowed to change per update step. Examples of the whole dataset have to be forwarded through the network to get a good measure of the update step. Following the forward step, gradients for each example must be computed before one update step can be performed. Since the datasets used for training a modern neural network contain many examples, it is computationally infeasible to compute each example's derivatives before taking one update step. Instead, examples are put through a network in batches. After each batch, gradients are computed, and the parameters are updated. This update algorithm is referred to as mini-batch stochastic gradient descent (SGD).

SGD is a widely used algorithm for weight updates. However, over the course of years multiple extensions to SGD have been proposed [132, 147, 251]. In this thesis we mainly make use of Adam [132] and its extension RAdam [147].

Adam is an extension of SGD, which combines the advantages of multiple extensions of SGD. On the one hand, Adam maintains a per-parameter learning rate. These per-parameter learning rates are adapted based on average

magnitudes of recently observed gradients. RAdam extends Adam by adding a default warmup phase where the learning rate of each parameter is increased until a specified amount of update steps is reached. Furthermore, RAdam introduces a term to rectify the variance of the adaptive per-parameter learning rates.

2.2.1.3 Types of Supervision

As stated above, a machine learning model is trained using an error measure, usually defined by a loss function. To obtain a meaningful error measure supervision is required. In this thesis, we make use of several kinds of supervision. We further mention different kinds of supervision for the training of deep neural models. Thus, we provide a common basis for such terms.

Full Supervision Deep learning owes its great success in recent years to three main factors. First, although the basic algorithms, *e.g.*, neural networks as such [168] or the backpropagation algorithm [198], used for deep learning have been available for many years, developments to train deeper networks [34, 200] enabled further research in this area. Second, the development and availability of general-purpose graphics processing units allowed the wide adoption of deep learning because of their extreme parallel and efficient computing power [136]. The last and most important factor is the availability of large-scale annotated datasets for the training of deep neural networks [71].

The great successes of deep learning are thus mainly based on the fully supervised training of deep networks in a fully supervised way. We refer to training under full supervision if each input sample has an associated ground truth annotation. One of the most pressing problems while training under full supervision is the costs in time and money of obtaining error-free annotations.

Unsupervised Training Other approaches for the training of machine learning models with varying amounts of supervision exist. First, we wish to mention approaches that do not rely on any human supervision at all. Such approaches are denoted as unsupervised learning [98]. Neural networks trained under an unsupervised learning regime are trained to identify and extract structures and underlying information from the available data. Thus, transforming the available raw data to another representation that might be simpler to analyze [98]. Close to unsupervised learning is the field of self-supervised learning. Here, a machine learning model is also entirely trained on the available data itself, *e.g.*, by predicting purposely hidden words in the area of Natural Language Processing [170, 186] or by solving computer vision tasks, such as structure estimation or image colorization [181, 254]. Here, the

task is not to find any structure in the available data but to directly solve specific tasks such as image colorization.

Weak Supervision The next type of supervision that we want to introduce is the large field of weak supervision. According to Zhou [257], weak supervision can be categorized into three types. The first type is *incomplete supervision* where only a subset of the available data is annotated, the rest of the available data is unannotated. The second type is *inexact supervision* where only coarse annotations are provided to the learning algorithm. The third type is *inaccurate supervision* where the given labels are not clean.

Incomplete supervision can then be further categorized into active learning [205] and semi-supervised learning [58]. Active learning describes a learning method where initially no or only very few annotated examples exist. Additionally, the learning algorithm is allowed to ask some form of an oracle, mostly a human annotator, for information on samples where it does not know how to classify them in the case of a classification algorithm. Semi-supervised learning describes a similar learning setup. However, the algorithm always has access to annotated data and a set of unannotated data, but there is no oracle to aid the learning algorithm.

Inexact supervision describes tasks where only some parts of the possible annotations are available. In contrast to incomplete supervision, every input has an annotation but the provided annotation might not be as fine-grained as desired. Consider, for instance, the problem of object detection. Here the task is to localize and recognize objects in a given image. Currently, the best methods rely on supervision of the locations of the objects and the class of each object [93, 233]. The task could be trained under inexact supervision if, during training, only the class information for each object in the image was available but not the location information. In our work that we present in chapter 3, we propose a novel model for end-to-end scene text recognition trained under the setting of inexact supervision.

Models trained under inaccurate supervision have to deal with noisy annotations. In contrast to the aforementioned types of supervision, here the model always has access to input with an associated full annotation. However, the annotation is not guaranteed to be correct, *e.g.*, because the given images were not correctly annotated.

2.2.1.4 Activation Functions

The weighted sum computed by a single neuron introduced in equation 2.1 is a linear function. If we were to stack multiple layers of neurons on top of each other, the resulting function that can be represented is still only a linear function. However, the world is not linear; thus, a method is required to

allow the computation of non-linear functions. The solution to this problem is simple, as a single non-linear function on top of a neuron can be used to transform the result of the linear function to a non-linear result. Depending on the use case, we use a multitude of activation functions.

An example of such an activation function is the *sigmoid* function:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{2.6}$$

The sigmoid function is a good choice as non-linearity in a neural network because the function's gradient is well-defined and straightforward. However, the maximum value of the gradient of the sigmoid function is 0.5. Using the sigmoid function in very deep neural networks leads to the vanishing gradient problem, where early layers of the network do not learn. The *tanh* function can be used to mitigate the vanishing gradient problem. However, current state-of-the-art methods mainly rely on rectified linear units (ReLUs) [176] as activation function. The function computed by a ReLU is a piecewise linear function with a stable gradient of 1 if the output of the neuron is > 0, else the gradient is 0:

$$\text{ReLU}(x) = \max(0, x). \tag{2.7}$$

Because the ReLU function has a stable gradient of 1, it is a perfect choice for the training of deep neural networks. Although ReLU is already an activation function that allows stable learning, several flavors of ReLU have been proposed. These flavours handle negative values differently. The *Leaky ReLU* function [163] for instance does not set all negative values to 0, instead values smaller than 0 are multiplied with the constant factor 0.01:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if} x > 0, \\ 0.01x & \text{otherwise.} \end{cases} \tag{2.8}$$

2.2.1.5 Normalization

Neural networks learn to approximate functions based on input data and a loss function. Optimizing the function approximation based on the loss function is a non-convex process. The optimization also has to overcome flat regions, or sharp minima [138], which makes optimization with algorithms based on SGD unstable, *e.g.*, due to the sensitivity to the choice of hyperparameters. Optimization is complicated in a deep neural network that consists of multiple non-linear functions, which depend on the output of earlier functions.

Batch normalization (BatchNorm) [109] is a method that allows training deep neural networks successfully. While it was initially believed that BatchNorm helps to reduce the changes of distributions of inputs to individual layers while a network learns (internal covariate shift), it was found that

BatchNorm helps to create a smoother optimization landscape [202]. Such a smoother optimization landscape mitigates flat regions, or sharp minima typically found in deep neural networks [138] and reduces the sensitivity of SGD based optimization methods to hyperparameters. We make frequent use of BatchNorm in nearly all deep neural networks trained by us in our work.

The idea behind BatchNorm is that the inputs are normalized by subtracting their mean and dividing by their standard deviation. Both are estimated based on the current batch. Next, a learnable scale coefficient and offset are applied to the normalized inputs:

$$\text{batchnorm}(x) = \gamma \odot \frac{x - \mu_B}{\delta_B} + \beta. \tag{2.9}$$

Here, x is the input batch, while μ_B, δ_B, γ, and β denote the mean of the batch, the standard deviation of the batch, the learnable scale coefficient, and the learnable offset, respectively.

While BatchNorm is an effective method to train deep neural networks, it suffers from its dependence on the size of a batch during training. BatchNorm is only effective if a network is trained with a large batch size. However, large neural networks and limited available memory render it challenging to use BatchNorm in every case. Group normalization (GroupNorm) [245] mitigates the dependence on the batch size while still allowing the same benefits of BatchNorm. GroupNorm not based on the size of the batch, but rather on the number of channels of a tensor. For normalization, GroupNorm divides the channels of a tensor into groups and computes the mean and variance for normalization within each group. In section 3.6.2 we make use of GroupNorm to train our models for the recognition of scene text.

Although BatchNorm and its variants perform very well, recent work argues that a reparameterization of the weights could already be enough to achieve the benefits of normalization layers for the training of very deep neural networks [47].

2.2.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are one of the most used kinds of deep neural networks in the area of computer vision. They have been proven to be very effective in a multitude of application scenarios; by showing impressive performance in the recognition of handwritten digits [68], image classification [94, 136, 252], object detection [93, 194, 195], and OCR [111, 112, 114, 141]. A CNN is a hierarchical, multi-layer neural network characterized by three essential factors: local connectivity, weight sharing, and pooling. In the following, we introduce each of the essential factors of CNNs. We provide a structural overview of a CNN in figure 2.4.

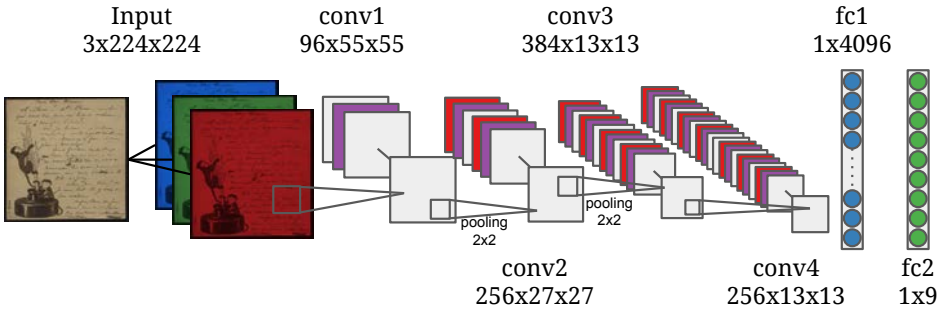


Figure 2.4: Illustration of a CNN, which is based on the AlexNet architecture [136]. It consumes an RGB image and runs the image and extracted features through multiple convolution and pooling operations followed by two fully connected layers. Each label denotes the name of the layer and the size of the feature map, *e.g.*, *number of channels* × *height* × *width*. The 9 neurons of the last layer can be interpreted as a classification result. In this example, the neural network could be used for the classification of documents.

Local Connectivity When dealing with high-dimensional inputs, such as images, a MLP would need to optimize millions of parameters, *e.g.*, a network that analyzes images of size 256×256, containing one hidden layer with 1000 neurons, and an output layer with two neurons, consists of $256 \times 256 \times 1000 + 1000 + 1000 \times 2 + 2 = 65,539,002$ parameters, which might be very difficult to optimize. In a CNN, each neuron is only connected to a small neighboring sub-region of the input, which reduces the number of parameters to optimize. Such a connection is accomplished by learning a small kernel of weights. The size of the kernel is also referred to as the *receptive field* and is mostly small, *e.g.*, 3×3 for convolutions in the famous ResNet [94] architecture. In a CNN the local neighborhood of an input is convolved with the learnable kernel. In the case of applying a CNN to a two-dimensional image at location x, y this can be written as:

$$\text{conv}(x, y, K, I) = (K * I)(x, y) = \sum_m \sum_n I(x - m, y - n) K(m, n). \tag{2.10}$$

Where $*$ denotes the convolution operation between the input image I and the kernel K; m and n denote the kernel width and height, respectively. When applied in a neural network the result of the convolution operation for each channel i of the input is summed and shifted by a learnable bias b, thus the resulting feature map o at location x, y of a convolutional layer for kernel j is defined as:

$$o(x, y)_j = \sum_i \text{conv}(x, y, K_{ji}, I_i) + b_j. \tag{2.11}$$

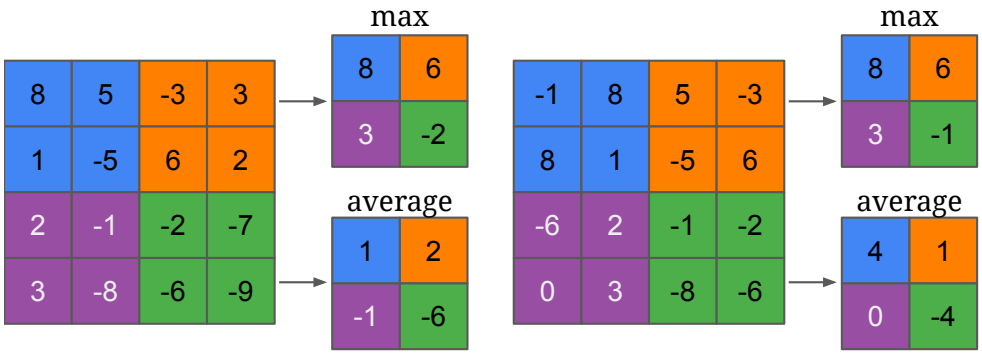


Figure 2.5: Pooling is an essential operation in CNNs. Here, we depict the result of max pooling and average pooling on two tensors, both using a 2×2 pooling width. On the one hand, we can observe that max-pooling only keeps the strongest responses of the feature map. Further, max-pooling makes the response translation-invariant to a certain extent, as we can observe in the feature map on the right side, a right-shifted version of the feature map on the left.

The learnable kernel K and the bias b form the learnable parameters of a CNN and are denoted as *filter*.

Weight Sharing Another feature of a CNN is that the weights of a learnable kernel are shared across the entire input. Weight sharing stands in contrast to MLPs where each weight is used only once. Each filter is replicated across all image subregions, forcing the network to learn generalized data representations. Furthermore, this allows a CNN to be position-independent.

Pooling The third factor is pooling, which may follow after a convolutional layer. Pooling is an operation that summarizes the output of several neighboring neurons. The benefits of using pooling are twofold: On the one hand, pooling reduces the dimensionality of feature maps, reducing the number of required operations in later layers and allows efficient use of memory. On the other hand, pooling helps to make learned representations invariant to small input translations. A widely used pooling operation in CNNs is max pooling, which we also showcase in figure 2.5. Max pooling only keeps the activation value of the neuron in the neighborhood with the maximum value. Another widely used pooling method is average pooling, where the operation's output is the average of the activations of all neurons in the neighborhood.

We can now see that all properties of a CNN help us to reduce the number of learnable parameters. Following our example from above we see that a CNN

with two layers would consist of $3\times3\times1000+1000+3\times3\times1000\times2+2 = 28,002$ parameters, which are much less parameters compared to a similar MLP.

2.2.3 Spatial Transformer

In the following, we introduce other advanced concepts used in deep neural networks that we use throughout our work. In this section, we introduce the concept of spatial transformers [113], a method we rely on in our work presented in chapter 3.

A spatial transformer is a differentiable image transformation method. It can be used for rectification or the modeling of attention. A spatial transformer is a module of a deep neural network and should not be confused with a Transformer that we introduce in section 2.2.5. A spatial transformer is a combination of three parts. First, a deep neural network, denoted here as localization network, predicts parameters of a spatial transformation that is to be applied to the input features. The second part uses these parameters to create a regular grid of sampling coordinates. Last, the predicted sampling coordinates are used by a differentiable sampling method to rectify the input features based on the predicted sampling coordinates. In the following, we introduce each of these components.

2.2.3.1 Localization Network

The first component takes the input features $I \in \mathbb{R}^{C\times H\times W}$, with C channels, height H, and width W. It regresses the parameters θ of a transformation that is to be applied to the input features. The localization network, denoted as f_{loc}, may be an arbitrary neural network. The amount of parameters in θ depend on the type of the transformation to be applied to the input features I. One example of a transformation could be an affine transformation. An affine transformation consists of six parameters, thus the localization network regresses the 6 parameters of an affine transformation matrix A_θ:

$$A_\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \end{bmatrix}. \tag{2.12}$$

The transformations may have any form as long as the applied transformation is differentiable with respect to its parameters. Calculating the gradients with respect to the transformation parameters allows gradients to flow to the localization network, enabling the network to learn to predict the correct transformation parameters.

2.2.3.2 Grid Generator

A successful transformation of the input features requires each pixel of the feature to be mapped to a location in the output feature. A parameterized

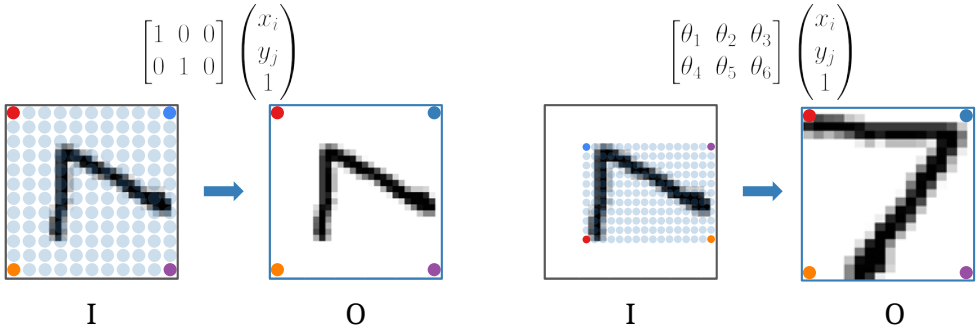


Figure 2.6: Two example applications of sampling grids to an input image I and the resulting output O. The sampling grid on the left represents an identity transform, while the sampling grid on the right represents a warping operation with an affine transformation. The solid colored dots represent matching corners. Image reproduced and adapted from [113].

sampling grid is created in the grid generator to map the input to the output. Here, we apply the predicted transformation of the last step to a regularly spaced grid $G \in \mathbb{R}^{H_o \times W_o}$. H_o and W_o denote the number of sampling locations in height and width, respectively. All in all, the grid contains H_o and W_o evenly spaced numbers over the interval $[-1, 1]$. The regular sampling grid is now multiplied with the predicted transformation matrix resulting in the new sampling grid G_{sample}, which is a set of two-dimensional sampling locations (u_i, v_j) with $i \in [0, \ldots, W_o]$ and $j \in [0, \ldots, H_o]$. Let us assume that our localization network predicts an affine transformation matrix A_θ as shown in equation 2.12. Let us further assume that the components of G are denoted as (x_i, y_j), then we can determine our sampling locations u_i, v_j as:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = A_\theta \begin{pmatrix} x_i \\ y_j \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 \\ \theta_4 & \theta_5 & \theta_6 \end{bmatrix} \begin{pmatrix} x_i \\ y_j \\ 1 \end{pmatrix}. \tag{2.13}$$

We display the effect of the described transformation in figure 2.6.

2.2.3.3 Image Sampler

The last component of a spatial transformer is the image sampler. The image sampler takes a set of sampling coordinates as described above and applies a differentiable sampling mechanism on the input features I to generate the output features O.

To obtain the output feature, we now apply a generic sampling kernel function k, *e.g.*, bilinear sampling, at each location (u_i, v_j) of the input feature map I. Please note that this sampling is applied identically to all channels to

preserve spatial consistency between channels. If we assume that we use bilinear interpolation as our kernel function k, our differentiable image sampling procedure to determine the pixels of the output features $O \in \mathbb{R}^{C \times H_o \times W_o}$ can be defined as:

$$O_{ij} = \sum_{h}^{H} \sum_{w}^{W} I_{hw} \max(0, 1 - |u_i - h|) \max(0, 1 - |v_j - w|). \tag{2.14}$$

This formulation of sampling is differentiable with respect to the sampling coordinates u_i and v_j, as well as the input features I; hence it is possible to use spatial transformers at arbitrary locations of a deep neural network.

2.2.4 Recurrent Neural Networks

The next concept we wish to introduce are recurrent neural networks (RNNs). RNNs can be thought of as neural networks with a loop. Naturally, they are very well suited for the modeling of sequences of arbitrary length. In our work, we make use of RNNs for the modeling of input and output sequences. RNNs are beneficial for the task of text recognition. In the following, we are going to introduce the main ideas and concepts of RNNs and the specific kind of RNN that we use in our experiments.

2.2.4.1 Vanilla RNNs

As already stated above, RNNs are neural networks with a loop. This loop makes it possible to run a function with the same parameters an arbitrary amount of times. A recurrent layer consists of an input i_t at time step t, a connection from its own computed output to itself (the hidden state) h_t and an output o_t. The output o_t at time step t directly depends on the output of the last output, *i.e.*, the hidden state h_t, the input i_t and the parameters of the RNN:

$$o_t = \sigma(W_{\text{output}} h_t + b_{\text{output}}), \text{with} \tag{2.15}$$

$$h_t = \sigma(\text{rnn}(i_t, h_{t-1})), \text{and} \tag{2.16}$$

$$\text{rnn}(i, h) = W_{\text{hidden}} h + W_{\text{input}} i + b_{\text{hidden}}. \tag{2.17}$$

The parameters of the RNN consist of three learnable weight matrices ($W_{\text{output}}, W_{\text{hidden}}, W_{\text{input}}$) and two bias terms ($b_{\text{output}}, b_{\text{hidden}}$). σ denotes a non-linear activation function, *e.g.*, tanh.

We can apply this function an arbitrary amount of times on the same or different inputs x. However, applying the function an arbitrary amount of time leads to a problem with the backpropagation algorithm. Backpropagation requires the computational graph created by the application of individual

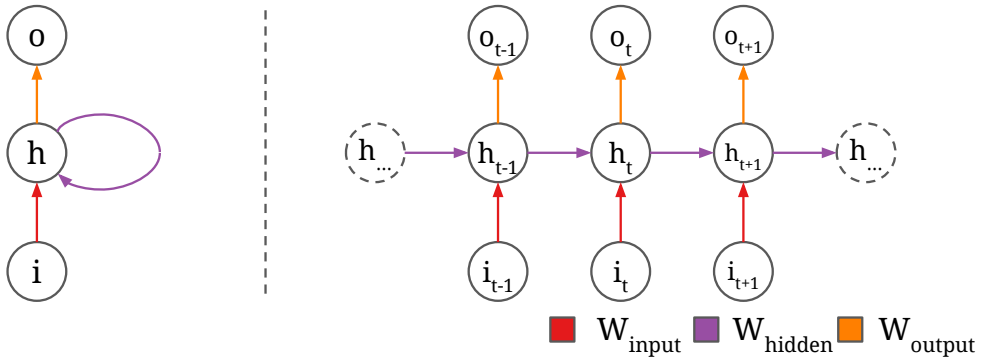


Figure 2.7: The computational graph of a RNN is a directed cyclic graph (left side). The loop needs to be unrolled (right side) to enable usage of the backpropagation algorithm. The unrolled graph also clearly shows that the involved learnable weights w_{input}, w_{hidden} and w_{output} are reused at each time step of the RNN. The colors correspond to the weight matrices, as indicated in the legend.

functions to be directed and acyclic. A directed acyclic graph is required because otherwise the calculation of gradients would be ambiguous, *i.e.*, the output of a function would not be well defined because it would not be clear how often we ran through the loop. The computational graph created by a RNN is cyclic; thus, a trick is necessary to use an RNN when training with backpropagation. The trick is to unfold the cyclic graph into an acyclic graph. We can unfold the computational graph since we can directly quantify the number of times we looped through a part of the network. We provide a graphical representation of this process in figure 2.7.

2.2.4.2 Long Short-Term Memory

RNNs are powerful models for learning to process sequences of data. However, when modeling long sequences, they heavily suffer from the vanishing or exploding gradient problem [100]. In a vanilla RNN gradients vanish or explode because elements of the weight matrices are reused and multiplied with the hidden state repeatedly. If a value of the weight matrix is lower than $|1.0|$, the gradients will start to vanish as they get smaller with every time step. If a value is larger than $|1.0|$, the gradients will explode as their magnitude increases with each time step.

Possible mitigation strategies for the vanishing gradient problem can be to add shortcut connections to earlier time steps [146]. Another approach is to use a different RNN design that allows free flow of the gradients from one time step to another. The long short-term memory (LSTM) model [99] enables such a free flow of gradients. In a LSTM, an extra hidden state is added. This extra state, called the cell state c, is not multiplied with any learnable

and reused weights. Thus, the gradients can not vanish anymore. A LSTM further consists of extra learnable connections, the so called *gates*. The gates control whether parts of the internal state of the LSTM shall be forgotten, the influence of the current input and hidden state on the new cell state, and how much information of the cell state is used for the new hidden state, which is also the output of a LSTM. We provide a schematic overview of a LSTM unit in figure 2.8.

The first gate, the forget gate, takes the concatenation of hidden state and input vector as input and puts it through a fully connected layer with sigmoid (f_t)activation. This gate aims to determine the parts of the cell state that are not necessary for further processing and can thus be forgotten. The second gate, the input gate, consists of two parts. Here the concatenation of input and hidden state is routed through two fully connected layers, activated with the sigmoid (i_t) and tanh ($\tilde{c}_t$) functions, respectively. The sigmoid activated layer is used to decide which parts of the input vector are to be stored in the cell state, while the tanh activated layer shifts the input to the domain of the cell state. The result of both layers is multiplied elementwise and added to the cell state. The last gate, the output gate, is used to filter the content of the current cell state. Here, we use a fully connected layer with sigmoid (o_t) activation to decide what parts of the cell state are used as the output of our LSTM at the current time step. More formally, we can write:

$$x_t = \text{concatenate}(h_{t-1}, x_t), \tag{2.18}$$
$$f_t = \sigma(W_f \cdot x_t + b_f), \tag{2.19}$$
$$i_t = \sigma(W_i \cdot x_t + b_i), \tag{2.20}$$
$$\tilde{c}_t = \tanh(W_c \cdot x_t + b_c), \tag{2.21}$$
$$o_t = \sigma(W_o \cdot x_t + b_o), \tag{2.22}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{2.23}$$
$$h_t = o_t \odot \tanh(c_t). \tag{2.24}$$

Here, *concatenate* denotes the concatenation operation of the input vector x at timestep t with the hidden state h at time step $t-1$. c denotes the cell state at the corresponding time steps, σ denotes the sigmoid activation function, and $\odot$ denotes an elementwise multiplication.

We can see that the cell state is only updated through addition and elementwise multiplication with the outputs of fully connected layers. Thus, no learnable weights are involved in updating the cell state, which allows gradient information to flow with only minor changes down to the first timestep.

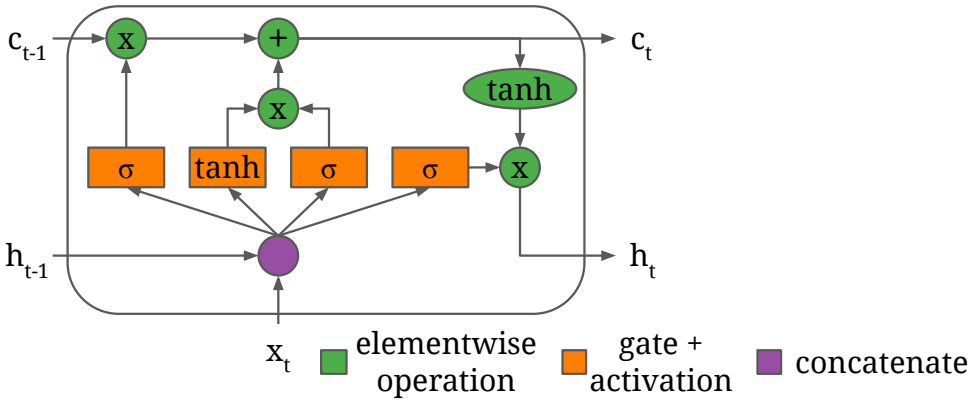


Figure 2.8: A LSTM unit consists of multiple gates. Each gate controls changes to the inner cell state c and the hidden state h, which is also the output of a LSTM. The layer on the left is the forget gate that controls which parts of the cell state shall be reset. The subsequent two gates form the input gate, determining what information will be stored in the cell state. The last gate, the output gate, determines the output of the LSTM. It is visible that the cell state c is not directly involved in any neural computations. Thus, the vanishing gradient problem is mitigated.

2.2.5 Transformer

RNNs and especially improvements of the original RNN model, *e.g.*, LSTMs are well established models for a range of approaches in sequence modeling, such as machine translation [219], speech recognition [86], or image captioning [230]. Although RNNs are very successful, certain design aspects of a RNN lead to degraded performance. On the one hand, it is not possible to parallelize the execution of a RNN because each state h_t depends on the last state h_{t-1}. Thus, it is not possible to compute the state h_{t+1} at the same time as computing the state h_t. The missing ability to parallelize execution leads to slow execution of RNNs for long sequences, where the available memory limits the usage of these models. The sequential nature of RNNs also makes it difficult for the model to learn learn dependencies between distant input positions [133].

A solution to these problems is the transformer model [229]. A transformer models sequences in parallel and is entirely based on attention mechanisms. In the following, we are going to investigate the transformer architecture in more detail.

The transformer is an encoder-decoder model where the encoder maps a sequence of input representations $x \in \mathbb{R}^n$ to a sequence of intermediate representations $z \in \mathbb{R}^n$. Based on z, the decoder produces a sequence of

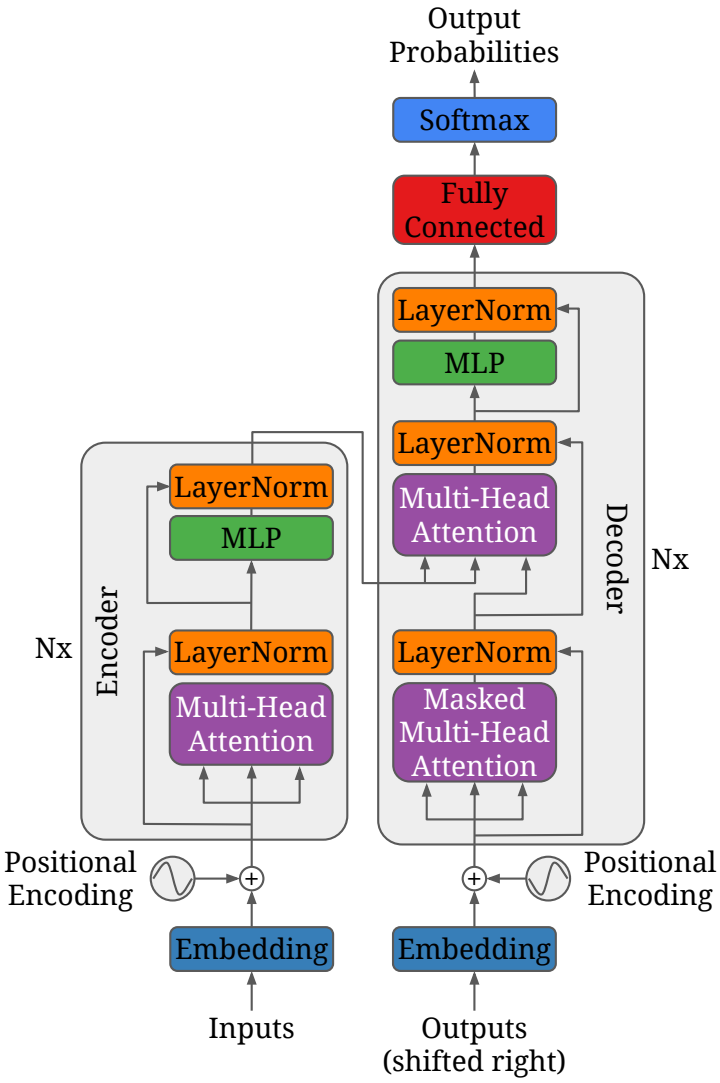


Figure 2.9: The architecture of a transformer model. The encoder consists of N encoding blocks (left side). Each encoding block consists of a multi-head self-attention layer and a MLP on top. The decoder also consists of N decoding blocks (right side). Each decoding block consists of a masked multi-head self-attention layer that only takes past timesteps into account. The masked self-attention layer is followed by a multi-head attention layer that takes the output of the encoder as input. The decoder guides the attention of the multi-head attention layer. Last, a MLP produces the output of the decoder, and this output is then transformed into a probability distribution using a fully connected layer with softmax activation. Adapted from [229].

outputs $y \in \mathbb{R}^m$. Encoder and decoder may contain multiple stacks of encoder or decoder blocks, as shown in figure 2.9.

Encoder and decoder model dependencies between distant input positions by employing self attention. Self attention allows the neural network to decide which parts of the input depend on each other. Thus, the transformer can easily access information from previous and future parts of the sequence without any barriers.

Attention in a transformer is modeled as multi-head scaled dot-product attention. Scaled dot-product attention refers to an attention model that computes an attention map, *i.e.*, a matrix where the value of each element indicates the importance of the feature map. The attention map is computed by multiplying a query Q, *i.e.*, the volitional cue with a set of keys K, *i.e.*, non-volitional cues, followed by a scaling operation and application of a softmax activation. The computed attention map is applied on the feature map, by performing a weighted sum of the elements of the attention map with the elements of the feature map:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{2.25}$$

Here, V denotes the feature map we want to attend to, and $\sqrt{d_k}$ is the scaling factor, with d_k denoting the dimensionality of queries and keys. Both encoder and decoder use scaled-dot product attention. However, the self-attention layer in the decoder uses a masked version of this attention. The masked version of scaled dot-product attention prevents the decoder from attending to future elements of the sequence.

Multi-head attention refers to applying multiple different attention heads on the same attention inputs (Q, K, V). The authors of [229] found it beneficial to project the inputs of the attention layer with multiple independent linear layers and perform scaled dot-product attention on each of these projected versions. The results of each application of attention are concatenated and projected to a resulting feature map. The usage of multi-head attention allows the model to attend to multiple representations of the same input simultaneously.

Following multi-head attention, layer normalization [15], another form of normalization, which is similar to batch normalization (section 2.2.1.5) and a residual connection [94], enabling faster and more stable learning are employed. At the end of each block, the transformer employs a fully connected linear projection of the attended feature maps.

The inputs to a transformer need to be embedded. Inputs could be embedded using a CNN or a learned word/character embedding, such as GloVe embeddings [185]. Additionally to the embedding of each input of the sequence, the position of each sequence item needs to be known. To achieve this,

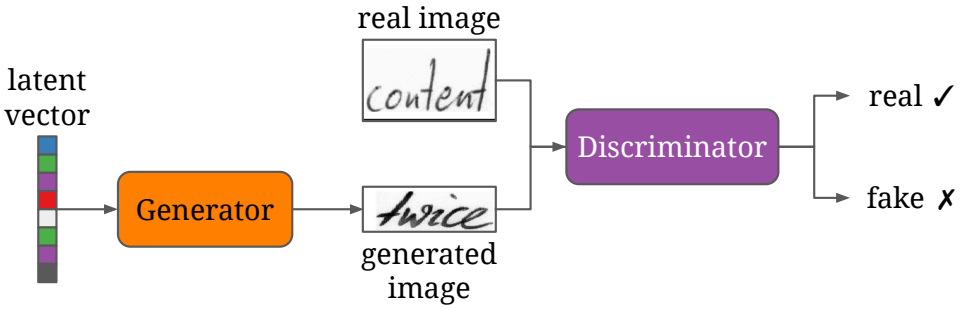


Figure 2.10: Depiction of the general structure of a GAN. A generator generates an image based on a randomized latent input vector. The generator is trained by a discriminator, who tries to determine whether a given input image is a real image or if the generator generated it.

a positional encoding (PE) based on sine and cosine functions of different frequencies is employed:

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right), \tag{2.26}$$

$$\mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos\left(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\right). \tag{2.27}$$

Here, pos denotes the position of an item in the sequence, and i denotes the dimension of the embedded input. This embedding function was chosen in [229] because it might allow the model to learn to attend by relative positions and because it might allow the model to extrapolate to sequence lengths not encountered during training.

In our work, we make use of the transformer model for the recognition of text in chapter 3. There, we also show that a transformer can be trained to propose regions of interest in an image in a weakly supervised way.

2.2.6 Generative Adversarial Networks

In the previous sections, we introduced a range of building blocks and concepts of deep neural networks. These concepts are mainly used in a discriminative manner, *i.e.*, mapping an input to a class label. In the following, we will introduce a different kind of neural network model: generative adversarial networks (GANs). GANs are a class of deep generative models whose objective is not the discrimination of values but rather the synthesis of new data. Learning to synthesize new data could be an excellent alternative to reduce the manual labeling burden while creating new training datasets for supervised machine learning methods. Hence we make heavy use of GANs in section 4.4 and section 4.5.

GANs were introduced by Goodfellow *et al*. [84] and describe a deep generative model based on two deep neural networks that are trained using the backpropagation algorithm. The two neural networks involved in a GAN behave as adversaries. The first model, the generator, tries to synthesize data as close as possible to real data. More formally, we could say that the generator tries to learn the distribution p_{data} over the real data items x. The generator models the distribution $p_{\text{generator}}$ by training a deep neural network G to perform a mapping from a vector of latent noise z to the space of the real data. The latent noise vector z is typically drawn from a well-defined probability distribution p_z, *e.g*., a normal distribution with mean 0 and fixed variance for each model. The objective of the generator is to synthesize samples whose distribution $p_{\text{generator}}$ is as close as possible to the real distribution p_{data}. The second model, the discriminator, tries to discriminate whether a given sample belongs to the real data distribution p_{data}, or whether it was generated by the trained generator G. The discriminator is also defined as a deep neural network D that produces a single scalar value. During training, both models constantly compete against each other. While the discriminator improves in discriminating where a given sample belongs, the generator uses the knowledge of the discriminator to fool the discriminator, which forces the discriminator to improve again. We could also say that D and G play an adversarial *minimax* game with the following value function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log(D(x))] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]. \quad (2.28)$$

In the end, the training objective of the discriminator is to produce a maximum output if the input belongs to the data distribution p_{data} and to produce a minimal output if the input belongs to the learned distribution $p_{\text{generator}}$ of the generator. At the same time, the generator tries to generate samples that fool the discriminator, *i.e*., maximize the output of the discriminator. Under the assumption that the discriminator D is a perfect discriminator, the generator will eventually converge to the real data distribution, so that $p_{\text{generator}} = p_{\text{data}}$. Then the discriminator will not be able to discriminate between the output of the generator and the real samples, *i.e*., $D(x) = \frac{1}{2}$. The neural networks involved in this adversarial game can be of arbitrary type; we provide a structural overview of a GAN in figure 2.10.

This formulation of an adversarial training shows great potential for image synthesis. Based on this inital formulation a wealth of other work has been proposed [11, 46, 110, 129, 171]. However, GANs suffer from a range of problems. Training of a GAN can be highly unstable. The gradient obtained when using the backpropagation algorithm on the formulation in equation 2.28 leads to unstable training [12], which can be mitigated by using different objective functions [11] that provide meaningful gradient information everywhere.

Another problem is the problem of mode collapse. If a generator experiences mode collapse, the samples generated by a GAN only fall into a single or a set of a few possible modes, *i.e.*, the generated samples are not diverse. Possible mitigation strategies are the addition of extra guidance networks that take the same input as the discriminator does and provide guidance on a manifold level [21], by making the generation of images class conditional and adding a classifier [201]. Furthermore, utilizing normalization methods found in the area of neural style transfer [74, 129] help to mitigate the mode collapse problem.

The initial formulation of GANs only allowed the generation of low-resolution images. With advances solving the problems mentioned above, new training mechanisms and network architectures emerged that allow the generation of high-resolution images of high perceptual quality [46, 127, 129].

3 Weak Supervision for Scene Text Detection and Recognition

In this chapter, we examine a possible strategy to lower the annotation cost for machine learning by using only a fraction of the typically required ground truth annotations while examing a use case of weak supervision and data synthesis in the field of scene text detection and recognition. In this chaper, we introduce our approach for data synthesis for the training of high-accuracy scene text recognition models. We also introduce a weakly supervised approach for end-to-end scene text recognition. Our approach simultaneously detects and recognizes scene text and is based on inexact weakly supervised learning because only textual labels are supplied but no annotations for the location of text. Parts of this chapter have been elaborated in the following publications: [22, 31, 32, 249].

3.1 Motivation

The term scene text describes text that we can find on, *e.g.*, photos, or videos of natural scenes. Thanks to the widespread adoption of smartphones, the usage of 360-degree cameras, or the mapping of the world in projects such as Google Streetview,[1] nowadays billions of images with the potential to contain scene text are available. The vast amount of available images forbids manual examination by humans. On the one hand, a manual examination would be very costly, while on the other hand, it would take a significant amount of time. Hence, automated methods to analyze text in scene images are necessary. The automated analysis of scene text allows for a range of applications, *e.g.*, support of navigational systems, content-based image retrieval, image-based machine translation, or as support for visually impaired people.

The development of accurate and robust scene text detection and recognition systems has been a widely researched challenge over several years [114, 178, 179, 206, 209, 211]. However, the development of accurate and robust localization and recognition of scene text is still a challenging task.

Looking at examples of scene text shown in figure 3.1, we can see why accurate and robust localization and recognition of scene text is a challenging

[1] https://www.google.com/streetview/ (last accessed August 31, 2021).

Figure 3.1: Depiction of typical examples of scene text found in real-world images. The text is embedded into the natural scene, inhibiting challenging properties such as challenging scene layouts (*left* and *right*), as well as curved text instances (*middle*).

task. Scene text appears in various forms. Scene text can be written in a multitude of fonts and languages. Scene text can be heavily distorted; such distortions include geometric distortions due to a manifold of view angles, blurred text due to camera movements, or prolonged exposure times, and reflections due to uncontrollable lighting conditions. Furthermore, as scene text occurs in natural scenes, backgrounds can be challenging, making it difficult to localize and recognize text. The challenging properties of scene text forbid the use of methods for the analysis of printed text as already discussed in section 2.1.2. Hence, novel approaches need to be developed. While in the "early" days methods based on handcrafted features, such as stroke width transforms (SWTs) [75], or histogram of oriented gradients (HOGs) together with support vector machine (SVM) classifiers [172, 250] were dominant, nowadays methods based on deep learning dominate the field of scene text detection and localization. Especially advances in the field of attention modelling [17], semantic segmentation [93] and the synthesis of training data [112] pushed the state-of-the-art significantly. Based on early results in the synthesis of training data, we developed a data synthesis pipeline for the training of our first scene text recognition models (see section 3.5) Nowadays, the usage of synthetic training data for the recognition of scene text is standard practice. However, although synthetic data sets for the localization of scene text are available [90], challenging datasets without localization annotations, but annotations for the recognition of textual content are available, *e.g.*, [215]. In this chapter, we introduce an end-to-end scene text recognition method that provides the textual content of a given scene image and the location of the text in the image simultaneously. So far, related work focused on either solving one or both scene text recognition tasks using fully annotated training data. Our method stands in contrast to related work because it is able to deliver not only the recognition result but also the location of individual words or characters without the need for any annotations of word locations.

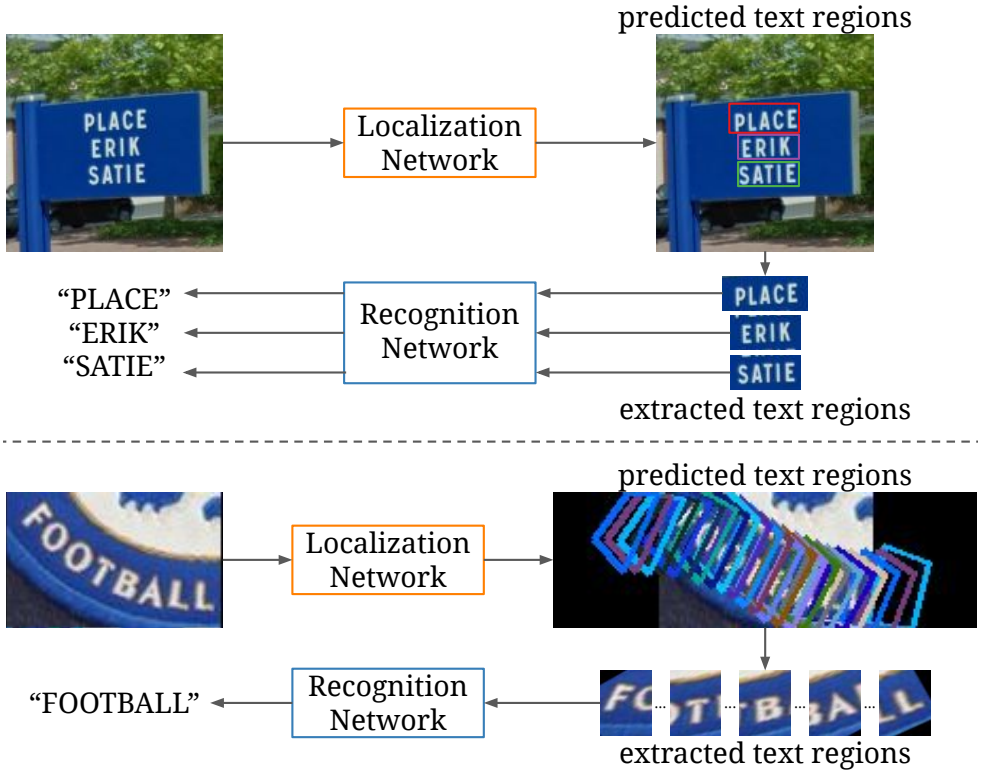


Figure 3.2: Our proposed pipeline can be used for two scene text related tasks. On the one hand, we can use our proposed pipeline for end-to-end scene text recognition (*top*). On the other hand, we can use our proposed pipeline for the task of scene text recognition on cropped word images (*bottom*). No changes in the network architecture are necessary for the change from one task to another.

Thus, our proposed method allows to reduce the amount of manual work required to annotate the training data for scene text recognition models. Please refer to figure 3.2 for a brief visual introduction of our proposed method.

Our model performs both tasks simultaneously and is trained solely under the supervision of the recognition part. The localization part is trained in a weakly supervised fashion, enabling us to learn to localize text without the need for localization annotations. Our proposed model can also be used for a multitude of scene text-related tasks, not only for end-to-end scene text detection and recognition. The model also shows highly accurate and robust results when used only for scene text recognition.

Figure 3.3: Depiction of examples of scene text found in real-world images. *(Top)* Three examples of full images containing scene text. Such images are used as input to a scene text detection network. The example images contain a wide variety of scene text. The image on the right contains an example of axis-aligned scene text. The image in the middle contains an example of curved scene text that is annotated using a polygon. The image on the left side contains a wide variety of challenging scene text examples. *(Bottom)* Examples of cropped word images that are used as input for the task of scene text recognition.

3.2 Scene Text Detection vs. Scene Text Recognition

Scene text detection describes the task of detecting or localizing text in a given image, while scene text recognition only describes the task of recognizing the textual content of already localized text regions, see figure 3.3 for a visual comparison. Localization is essential because it is necessary first to determine whether an image contains any text before further costly analysis steps are performed. Localized text regions can be represented in a multitude of ways. In its most simple form, a text region is represented as a 4-tuple, a bounding box, stating the x and y coordinates of the left upper corner and the width w and height h of a box that fully encloses the text: (x, y, w, h). Such an axis-aligned box can be used to represent text regions in a given image. However, as soon as the text itself is not axis aligned anymore, *e.g.*, rotated or sheared, an axis-aligned bounding box is not the best representational choice anymore because the text region might contain too much background, which distracts the text recognition algorithm. Here, a bounding box could be represented by a set of four 2-tuples, each 2-tuple representing the x and y coordinate of a corner of the box, *e.g.*, $[(x_{\text{left}}, y_{\text{top}}), (x_{\text{right}}, y_{\text{top}}), (x_{\text{right}}, y_{\text{bottom}}), (x_{\text{left}}, y_{\text{bottom}})]$. If the text is written in a format that can not directly be approximated by a bounding box, pixel-wise segmentation can be used to get a good approximation of the texts' shape. Such a pixel-wise segmentation may be represented as a pixel-wise

mask of boolean values, where *True* indicates a text region and *False* indicates background.

Scene text recognition directly operates on the image content of the localized text regions. Therefore, scene text recognition depends on the task of scene text detection. However, both tasks can be handled individually in evaluation and development since they do not share many properties. While scene text detection deals with the localization of text regions in a large image, scene text recognition deals only with recognizing textual content. No further localization is necessary. Even our proposed method for end-to-end scene text detection and recognition relies on solving both tasks individually. However, we use the knowledge and requirements of the recognition stage of our model to train the localization and recognition stage jointly.

3.3 Related Work

The field of scene text detection and recognition has been researched for many years [160]. During this time, many approaches were introduced and advanced the state-of-the-art further and further. While, in the beginning, classical computer vision methods based on handcrafted features dominated [75, 235], nowadays methods based on deep neural networks dominate the research in the areas of scene text detection and recognition [141, 253]. In this section, we introduce and discuss related work in scene text detection and scene text recognition. While we introduce early approaches using handcrafted features, our primary focus lies on related work using deep neural networks as we use in our work. First, we examine related work in the area of scene text detection, then we examine the area of scene text recognition, and last, we provide an overview of related end-to-end systems.

3.3.1 Scene Text Detection

Early scene text detection methods based the overall approach of text localization on well-known steps of existing approaches employed for printed text localization and recognition. In [59], for instance, Chen and Yuille used a set of derivative features and an ensemble of classifiers to localize possible regions of text. These regions were then binarized and fed to an off-the-shelf optical character recognition (OCR) tool for the recognition of printed text. Other methods extract connected components, which represent individual characters, using methods such as maximally stable extremal regions (MSERs) [179], or stroke width transforms [75] to predict a range of text region candidates. In a subsequent step, these text region candidates are filtered by classifiers and grouped into words. Further methods rely on multi-scale sliding window

classifiers that identify possible text regions [234]. Early methods rely on the researchers' observations and manual fine-tuning of a range of hyperparameters for each involved benchmark dataset. The required fine-tuning made it challenging to adapt the approaches to other datasets, and manual feature extraction became more complicated.

With the breakthrough of Krizhevsky *et al*. [136] in Imagenet classification, research in the area of scene text adopted deep learning techniques more and more.

Jaderberg *et al*. [114] build on the ideas of [239] by using a deeper convolutional neural network (CNN) with maxout [85] activations for localization of individual characters. They form lines using thresholding on an image of the detector response. Regions are connected by keeping all regions where characters are close to each other. Individual words are formed by using thresholding on the extracted text line images. Connected components that are close enough to each other are grouped into words. In [111] Jaderberg *et al*. propose to use region proposal methods, as used in object detection [259]. The proposed regions are filtered, and then a bounding box regression CNN is used to adjust the proposed bounding boxes for individual words. Although the described methods make use of deep learning, their overall procedure is still quite complex and makes heavy use of carefully tuned approaches.

This changed with [90], where Gupta *et al*. introduced the first large-scale synthetic dataset for the training of scene text detection models. To show the suitability of their proposed dataset, they proposed a novel scene text detection model based on the YOLO [194] object detector that directly predicts bounding boxes of individual words and can be trained end-to-end on input images with the corresponding word bounding boxes. From then on further approaches consisting of only a single neural network for prediction of text regions emerged [118, 224].

Following this, attention of research shifted to examining more complicated forms of scene text, mainly oriented scene text, where words are not axis aligned anymore, but rather rotated in arbitrary ways [142, 144, 151, 152, 153, 162, 206, 256]. Here, the basic idea of the approaches stayed the same: using a single neural network to predict text regions directly. However, these approaches predict oriented bounding boxes, as indicated in section 3.2 instead of axis-aligned bounding boxes. The used network architectures are based on state-of-the-art object detection networks [149, 194, 195] and contain text specific enhancements that compensate the differences between arbitrary object detection and text detection.

The most current approaches for scene text detection focus on curved text examples. Such curved text regions can not be approximated by a single

rectangle, thus the approaches shifted to the prediction of polygons [141, 240] and semantic segmentation maps [16, 61].

All of the current approaches are trained in a fully supervised fashion. Groundtruth for axis-aligned bounding boxes and oriented bounding boxes is obtained by human annotations or synthetic data [90]. However, the ground truth for approaches based on polygons and semantic segmentation maps is always annotated by humans, making these approaches very costly. The objective of our work is to find a way to circumvent the necessity of human-annotated text locations in images. Although our proposed approach is not directly comparable and able to compete with state-of-the-art text detection methods directly, we show that approaches similar to ours are viable and might alleviate the need for human annotations in the future for further applications on data not represented by the available data.

3.3.2 Scene Text Recognition

The field of scene text recognition followed a similar development path as the field of scene text detection. Early approaches based on the extraction of handcrafted features proposed a system for scene text detection and recognition most of the time. With the rise of deep learning, researchers began to concentrate on each task individually. Early approaches recognized the textual content of found words by recognizing each character individually. In [235] Wang *et al*. propose a method that uses HOG features for localization and recognition of individual characters. Neuman and Matas [179] localize single characters using MSERs. Characters are recognized using a SVM on directional features. Mishra *et al*. [172] recognize individual characters using sliding window character detectors and a recognition model based on a conditional random field (CRF) and dictionary post-processing. Wang *et al*. [238] use a sliding window detector together with a random fern classifier. Yao *et al*. [250] propose to learn to extract strokes of individual characters. The extracted strokes are then used as features for character classification with 62 individual SVMs. These approaches use handcrafted features and complex pre- and post-processing techniques to recognize individual characters. Furthermore, many parameters have to be tuned by hand, and these proposed methods tend not to generalize well.

Bissacco *et al*. [41] propose one of a set of scene text recognition approaches that lie on the edge of using handcrafted features and deep features. They make use of handcrafted features, *e.g.*, HOG features, to segment identified text lines into single characters. They then recognize each character individually using a multilayer perceptron (MLP). Wang *et al*. [238] use a CNN for the recognition of single centered characters, which were found before by using sliding a

CNN character detector over the entire image. Jaderberg *et al*. [114] propose to use a deeper CNN for character recognition. These approaches only work on individual characters and highly depend on the accurate localization of a single character because they are not robust enough to correctly deal with misaligned character predictions. In our experiments on scene text recognition, our model also tries to extract single characters. However, these extracted characters do not need to be centered, making our model very robust.

Later Goodfellow *et al*. [83] proposed a CNN that takes a cropped house number image as input and directly predicts the textual content without further segmentation into individual characters. Their proposed CNN consists of five classifiers, where the first four classifiers predict single digits and the last classifier predicts the number of digits in the input image. Jaderberg *et al*. [112] extended this to unconstrained text recognition, by using 23 individual classifiers. Furthermore, Jaderberg *et al*. proposed a network with a classifier with more than 90 000 classes, *i.e*., all words found in a dictionary, that directly predicts the word in the image. Based on these ideas further approaches by He *et al*. [95] and Shi *et al*. [207] propose to use a recurrent neural network on top of the CNN to predict the textual content of the word image. In our work, we also use a CNN with a sequence model to predict the textual content of a word image. However, we propose to use a transformer instead of recurrent models such as long short-term memorys (LSTMs).

The overall idea of a CNN based feature extractor with a sequence model for recognition has since then been extended by various other methods [140, 141, 143, 150, 208, 209, 237, 253]. Further developments mainly concentrate on integrating attention mechanisms or mechanisms to rectify the given word crops to enable better recognition accuracy.

In [150] Liu *et al*. propose a network architecture that includes a spatial transformer, which is used to find and rectify individual characters. This work is similar to ours. However, we do not use the spatial transformer only for rectification, but rather in various ways, *i.e*., text detection or character cropping. Further methods utilize a spatial transformer network for rectification [208, 209, 253]. All of these methods first rectify the word image using a predicted thin-plate spline transformation. In [253], Zhan *et al*. apply multiple thin-plate spline transformations before the rectified result image is passed to the actual recognition part of the network. Shi *et al*. [208, 209] further utilize an attention guided recurrent neural network for the prediction of character classes. Our work is inspired by [208]. However, we go a step further and utilize the spatial transformer in multiple ways for multiple tasks with a single and simple network architecture. Another approach that tries to rectify the word image was introduced by Luo *et al*. [161]. They do not use a spatial transformer for rectification. Instead, they predict an offset map applied to each pixel, thus

rectifying the word image. Cheng *et al*. [60] propose a network architecture that deals with arbitrary rotated text lines. Another approach using attention without explicit rectification was proposed by Liao *et al*. [141]. They propose to use an attention-guided recurrent neural network on two-dimensional feature maps.

Other approaches require information about the location of each character in the word image during training. Wang *et al*. [237] propose to use to use a convolutional LSTM [211] with attention as recurrent network for character prediction. The proposed attention mechanism is guided by location predictions of the center of each character, which are learned in a supervised way. Lia *et al*. [143] cast the problem of text recognition as a semantic segmentation problem. They predict the location and class of each character. Both approaches require annotated character bounding boxes during training. Annotations for each character are simple to obtain when using synthetic data but very costly when training on real data or adapting the approach to real data.

The work that is closest to ours in the area of pure scene text recognition is the work of Wang *et al*. [236]. They use a single convolutional feature extractor and the decoder part of a transformer (see section 2.2.5) to predict the textual content of a word image. We extend this idea and produce better-focused inputs to the transformer using a spatial transformer (see section 2.2.3).

3.3.3 End-to-End Systems

Until now, we presented systems for scene text recognition that solve each of the two tasks (scene text detection and scene text recognition) using two distinct approaches. Here, we focus on systems that consist of a single neural network to perform end-to-end scene text recognition. Smith *et al*. [215] propose a single neural network for the recognition of text on street signs. They use a convolutional feature extractor together with a stack of LSTMs for the extraction of textual content from individual text lines. Wojna *et al*. [244] propose another single neural network for the recognition of text on street signs. Their network utilizes a convolutional feature extractor and produces character outputs using a spatial attention mechanism. Both approaches can transcribe the textual content of a given scene text image. However, they can only produce the textual content, not the location of the text in the image.

A system for the prediction of textual content and the location of words was introduced by Li *et al*. [139]. The system first extracts features using a convolutional feature extractor. Based on the features, text proposals are generated. The text proposals are used by a text detection network that also refines the text proposals, which are then pooled and used as input to

the text recognition sub-network. The whole system is trained end-to-end using annotations for the textual content and bounding boxes of the text. A similar system was proposed by Liu *et al*. [151]. Their system also consists of a single neural network that predicts the location and content of words in an input image. The network consists of a convolutional feature extractor whose extracted features are first fed to a text detection branch to predict text locations. Second, the predicted location information is used to crop and rectify identified text locations, which are then recognized using a text recognition network. This system is also trained using annotations of text locations and textual content, which stands in contrast to our proposed method, where we only use textual annotations to train our model to predict locations and content of words.

3.4 Datasets for Scene Text Recognition

Annotated data is of utmost importance for research in the area of scene text detection and recognition. As annotated data for training is a scarce resource, most researchers use synthetic data to train their proposed systems. At the same time, proposed systems are evaluated on annotated real-world benchmark datasets. In this section, we briefly introduce commonly used training datasets and all benchmark datasets that we evaluate our proposed model.

3.4.1 Training Datasets

The first large-scale synthetic training dataset is called *MJSynth* and was introduced by Jaderberg *et al*. [112]. The dataset consists of 9 000 000 images containing 90 000 words drawn from a dictionary. The images are rendered using 1400 different fonts. Furthermore, they are placed on different natural background images and distorted. All images have a fixed height of 32 pixels but variable width to account for varying aspect ratios. We also follow the approach of Jaderberg *et al*. and synthesize another 8 000 000 samples using our data synthesis tool described in section 3.5 and use this data for some of our experiments. We provide some examples from these datasets in figure 3.4.

Another synthetic dataset that we use for the training of our proposed method is the *SynthText* dataset by Gupta *et al*. [90]. This dataset consists of 800 000 images containing 8 000 000 words. The words are placed in plausible locations with visually correct orientations and matching illumination properties on real scene images. Following related work, we extend the *SynthText* dataset with 1 600 000 images from the *SynthAdd* dataset [140], which were

Figure 3.4: Examples taken from the synthetic MJSynth [112] and the SynthText dataset [90]. The top three rows show examples from the MJSynth dataset. The MJSynth dataset contains only grayscale images rendered on several backgrounds. The two bottom-most rows contain samples from the SynthText dataset. The images are colorful and more challenging than the MJSynth samples. However, only using all available training datasets leads to state-of-the-art performance.

generated with the same generation tool as the images of the *SynthText* dataset. We provide an example from the *SynthText* dataset in figure 3.4.

For our experiments on end-to-end scene text recognition, we utilize two different datasets. On the one hand, we experiment on synthetic toy datasets to illustrate the feasibility of our proposed approach. Our toy datasets are based on the StreetView House Number (SVHN) [177]. Here, we create two datasets. On the one hand, we create a regular dataset, where we place four digits in a regular grid (see figure 3.5). On the other hand, we randomly place individual house number images on a background (see figure 3.5). We synthesize 110 000 images and split them into 100 000 images for training and 10 000 images for evaluation for each dataset, respectively.

Besides running experiments on synthetic toy datasets, we also train and evaluate our model on the French Street Name Signs (FSNS) [215]. The FSNS dataset contains images of street name signs from France. They were captured from Google StreetView images. One sample contains up to four different views of the same street name sign (see figure 3.6).

Each image is only annotated with the textual content of the street name sign, namely the name of the street depicted in the image. The location of each word is not annotated, making this dataset a perfect dataset for our proposed method. All in all, the dataset consists of 1 081 422 images, which are divided into 1 044 868 images for training, 16 150 images for validation and 20 404 images for testing.

Figure 3.5: Examples taken from our toy datasets. The first two images are taken from the regular grid dataset and show how we placed house numbers in a regular grid. The last two images are taken from the randomly placed dataset. Here, we randomly placed house numbers on a single image with a plain background.

Figure 3.6: Example images taken from the FSNS dataset. Each input image consists of up to four views of the street sign. If four different views are not available, the remaining views are filled with random noise (*top-right image*). Many images contain blurred text (*middle-left and bottom-left image*), distractors (*middle-right image*), or even street name signs displaying a different name (*bottom-right image*).

3.4.2 Benchmark Datasets

In the following, we shortly introduce all benchmark datasets we use throughout the experiments presented in this chapter.

For evaluation of our models for end-to-end scene text recognition, we use the aforementioned FSNS dataset. For evaluation of our scene text recognition models, we use a range of standard benchmark datasets. These benchmark datasets were introduced throughout the years for several scene text recognition challenges.

The first dataset, we evaluate our proposed method on is the ICDAR 2013 (IC13) dataset [125]. The IC13 dataset consists of 1095 cropped word images

of focused scene text. The images were captured using an ordinary digital camera. For a fair comparison with other approaches, we remove all images with non-alphanumeric characters, which leaves us with 1015 images. We also use other datasets containing focused scene text for the evaluation of our models. On the one hand, we use the IIIT5K-Words (IIIT5K) dataset [172], which contains 3000 cropped word images that mostly contain horizontal words images. However, some images also contain challenging curved text examples. On the other hand, we use the Street View Text (SVT) dataset [239] that consists of 647 cropped word images. These images contain horizontal text lines. However, the quality of many images is severely degraded.

We also evaluate our models on more challenging datasets. The ICDAR 2015 (IC15) dataset [124] is a dataset that contains 2077 cropped word images of incidentally captured scene text. Here, the text was captured using Google Glass[2] without the intent to capture text, hence the notion of incidental. Most images are severely distorted or blurred. For a fair comparison, we evaluate on the entire dataset, but also on the smaller ICDAR 2015-1811 (IC15-1811) subset that only contains words with alphanumeric characters.

The last two datasets contain mostly perspectively distorted or curved text samples. The Steet View Text Perspective (SVTP) dataset [191] contains 645 cropped word images. The images were collected from Google StreetView and mostly contain word images with a high rate of distortions. The CUTE80 (CUTE) dataset [196] contains 288 cropped word images. The images are of high visual quality. However, the dataset contains a large number of curved text instances.

In figure 3.7 we provide examples for typical samples from each of the introduced benchmark datasets.

3.5 Synthetic Data for Scene Text Detection and Recognition

The training of a deep neural network requires a large amount of training data since deep neural networks learn the function that they are supposed to compute based on the available data. For image classification algorithms, a general rule of thumb is that a supervised deep-learning algorithm will achieve acceptable performance when trained from scratch with around 1000 labeled samples per category and even human-level performance with a dataset having at least 100 000 labeled examples per category[3] If we were to consider the task of scene text recognition as an image classification task

[2] https://www.google.com/glass/start/ (last accessed August 31, 2021).

[3] https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/ (last accessed August 31, 2021).

Figure 3.7: Each row depicts typical samples from one of our benchmark datasets for scene text recognition. The ordering of the datasets is as follows: CUTE, ICDAR 2013, ICDAR 2015, IIIT5k Words, Street View Text, and Street View Text Perspective.

with 95 categories, including the Latin alphabet in upper and lower case, numbers, as well as a set of special characters, we would end with a total required number of at least $95 \times 5000 = 475000$ samples to achieve acceptable performance.

Most of the benchmark datasets for text recognition that we introduced in section 3.4 include a set of images that can be used for the training of a model. However, the size of these sets is limited, *i.e.*, 229 training images in the IC13 dataset, or 2000 training images in the IIIT5K dataset. The manual annotation of additional images would be very costly. Therefore, Jaderberg *et al*. [112] introduced the idea of an image synthesis pipeline to generate massive amounts of synthetic training data. Although the dataset by Jaderberg *et al*. already contains 9 000 000 images, the dataset does not include samples with blurred text, reflections, and lighting artifacts, or special characters, such as

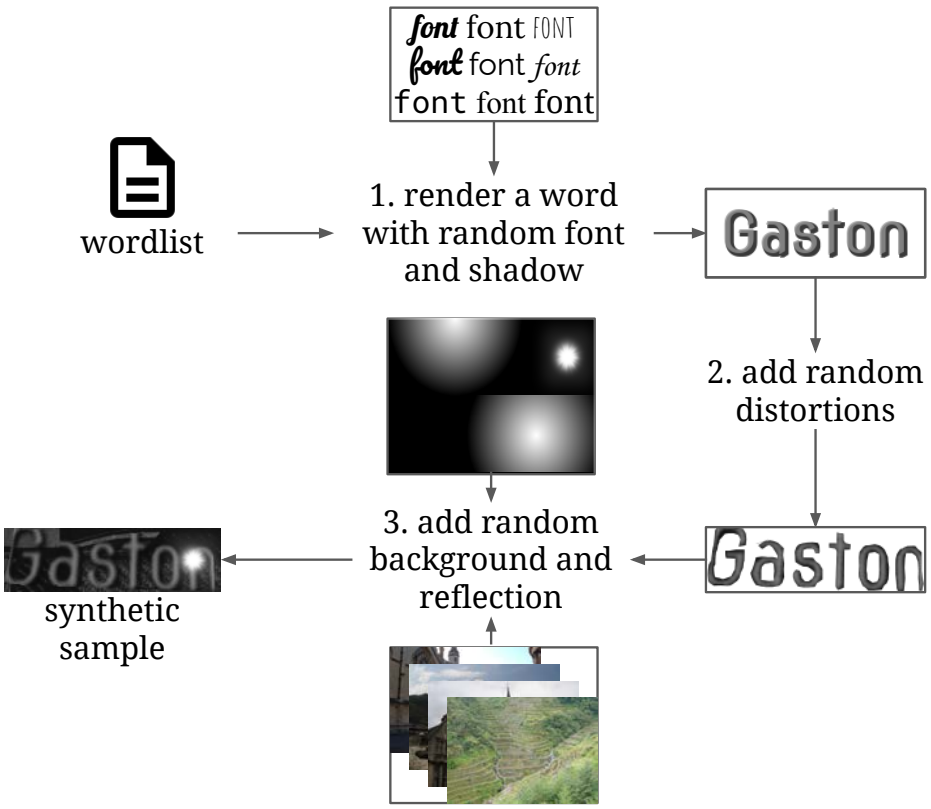


Figure 3.8: Our proposed data synthesis pipeline consists of multiple steps. First, we use a wordlist to determine the word to render. We render the selected word using a randomly sampled font. We also randomly add a shadow to the rendered word. We apply random distortions, such as gaussian blur, perspective transformations, or rotations to the word image following the rendering process. Last, we blend a random excerpt of a random background image and randomly add reflection effects. In the end, we save the generated sample.

exclamation marks, colons, *etc*. This is why we decided to follow the approach of Jaderberg *et al*. and implement a data synthesis tool that can produce samples, which include the missing properties of the samples provided by Jaderberg *et al*. In the following, we will introduce our data generation pipeline in more detail.

Our proposed data synthesis tool is implemented as a Python program. It can produce samples using an arbitrary amount of fonts, an arbitrary amount of backgrounds ranging from single color backgrounds to natural image backgrounds, and various forms of image distortions or blur effects. The generation process of a single image incorporates a set of operations. Please see figure 3.8 for a graphical overview of our data synthesis pipeline.

Before any sample is generated, the user specifies all fonts that shall be used for the generation. Google[4] offers a large set of different fonts that are well suited for the generation of artificial scene text samples. The user may specify a wordlist that is used to draw words to be created from. Our data generator randomly adds special characters to a specified fraction of words to allow special characters to appear in the generated images.

After generating a word, selecting the font, font size, and background of the current sample, the generator selects a text color. The text color is constrained to have a minimum contrast compared to the background color to ensure the readability of the text. Next, the text is rendered using the specified font and font size. Randomly, we also render the text a second time with a different color to add an outline to the text or add a shadow effect, which can also be found on real-world images.

Following the rendering of the text, we apply a set of image transformations to each sample. We mimic fast camera movements or unsteady photographs by applying a random gaussian blur to the rendered text and background images. We also distort the rendered text using random perspective transformations. These perspective transformations skew the text, thus providing it with visual properties of incidental scene text. Following perspective transformations, we also apply random rotations. Besides randomly rotated text, we also render a certain amount of text images not as straight lines but rather as curved text, as found on many billboards. For rendering curved text, we place the characters on a *sin* or *cos* curve of differing lengths. Text found in scene images often contains reflections of sunlight or other artificial light sources. We also add such reflections to our samples using two different approaches. On the one hand, we use predefined reflection overlays to blend with the generated image to simulate a white glare on the image. On the other hand, we make use of metaballs [42] to model small reflections on images. The rendered text image is blended with the background image in the last step, thus producing our final sample.

In figure 3.9 we provide some examples generated by our pipeline and compare them to samples taken from the dataset of Jaderberg *et al*. We argue that our samples look similar to real-world samples and the samples of Jaderberg *et al*. However, our samples also include blurred content, a set of distortions, and lighting effects missing in the samples of Jaderberg *et al*.

[4] `https://fonts.google.com/` (last accessed August 31, 2021).

Figure 3.9: Comparison of samples generated by the data synthesis pipeline of Jaderberg *et al*. [114] and samples generated by our proposed data synthesis pipeline. The top three rows show samples from the MJSynth dataset of Jaderberg *et al*. The bottom three rows show samples generated using our proposed data synthesis tool. The data synthesized by our pipeline looks more realistic and challenging. Thus, we are confident that our data is a good enhancement of already existing training data.

3.6 A Weakly Supervised Neural Network for End-to-End Scene Text Recognition

When reading a text, a human does so in a sequential manner. The first action is to put attention on a line of text. The second action is to read each character of this line of text sequentially and then attend to the following line of text. Most proposed systems for end-to-end scene text recognition do not behave in this way. Previous work instead solves the problem by analyzing all available information at once. Analyzing all available information at once might be more efficient but is also more difficult. Following the example of a human reader, we see that the task of reading text on an image is decomposed into several subtasks. First, the text is localized line by line or word by word, and then each word is read character by character. Although related work divides between detection and recognition, scene text detection and scene text recognition are often handled by individually designed solutions.

In this section, we propose a novel method that can perform scene text recognition and localize and recognize text in an end-to-end fashion. To switch between both tasks, our proposed method does not need to be adapted, and it can be trained using only the annotated textual content of the image. No annotations for the locations of single characters or words are necessary.

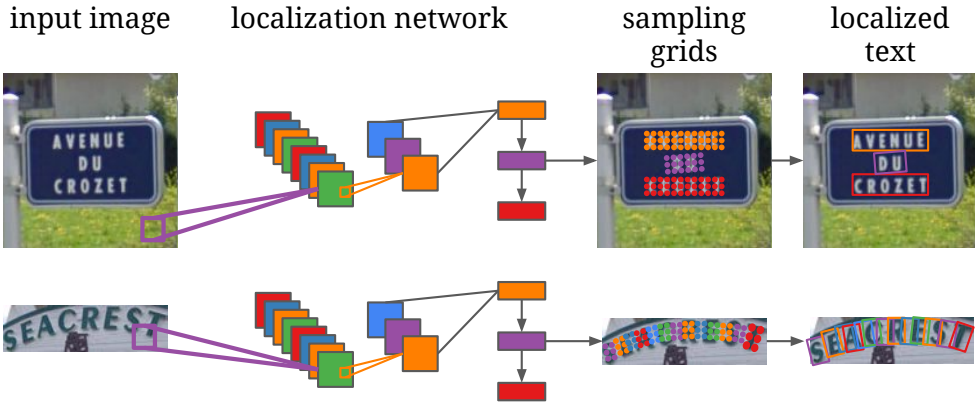


Figure 3.10: Our localization network consumes the input image with text and produces a set of sampling grids later used to crop individual text regions. We train the localization network to predict regions containing entire words for our experiments on end-to-end scene text recognition. For our experiments on scene text recognition, we train the network to predict regions that contain characters or parts of characters.

Our network is designed so that it can only correctly recognize the textual content if it can also correctly localize the text.

Our proposed network consists of two parts. We call the first part the *localization network* and the second part the *recognition network*. Both networks utilize a ResNet [94] based feature extractor. The localization network produces regions of interest using a recurrent spatial transformer network, while the recognition network recognizes the textual content of all extracted regions of interest using a transformer [229] (see also section 2.2.5).

3.6.1 Recurrent Spatial Transformers for Text Detection

Our localization network has the task of, depending on the task, predicting regions that either contain an entire word or a single character. The localization network itself is trained in an unsupervised fashion by the recognition network. Our network consists of a convolutional feature extractor and a recurrent spatial transformer. We provide a structural overview of our proposed network architecture in figure 3.10.

In the following, we describe the convolutional feature extractor and the concept of a recurrent spatial transformer.

Feature Extractor In our network, we use a deep convolutional feature extractor based on the ResNet [94] architecture. We chose to use ResNet over other well-known network architectures because ResNet does not suffer as

much from the vanishing gradient problem as other network architectures, such as VGG [213] or Inception [220] do. The feature extractor computes a function $f_{\text{loc}}^{\text{conv}}$ and takes an input image Image $\in \mathbb{R}^{C \times H \times W}$ with C channels, height H and width W as input and predicts a feature vector V. This feature vector is used as input to a recurrent network, which is the first part of our recurrent spatial transformer.

Recurrent Spatial Transformer Following the convolutional feature extractor, we propose to use a recurrent spatial transformer. A recurrent spatial transformer is an extension of the spatial transformer [113] introduced by Jaderberg *et al*. (see also section 2.2.3). Our recurrent spatial transformer utilizes a recurrent neural network (RNN) to predict a set of N parameters θ that are used to crop regions of interest from the input image. Thus, we extend the original spatial transformer to apply multiple transformations at the same time. The RNN itself computes a function $f_{\text{loc}}^{\text{rnn}}$, which takes the feature vector V of the convolutional feature extractor and the hidden state of the last time step as input. The output of the function is a set of affine transformation parameters θ^n with $n \in \{0, \ldots, N-1\}$ for each time step h the RNN is run. Thus, the recurrent spatial transformer computes the following function:

$$V = f_{\text{loc}}^{\text{conv}}(\text{Image}), \tag{3.1}$$

$$A_\theta^n = f_{\text{loc}}^{\text{rnn}}(V, h_{n-1}). \tag{3.2}$$

With A_θ^n being an affine transformation matrix, as shown in equation 2.12.

The following building blocks of a spatial transformer, as introduced in section 2.2.3 remain the same, with the exception that we consistently produce N outputs for each of the remaining steps.

3.6.2 Text Recognition Network

Following the localization network, we employ a text recognition network. The image sampler of the recurrent spatial transformer in the localization network produces a set of N image crops. These image crops are used as independent inputs to the recognition network. Depending on the task at hand, each extracted image crop contains textual data at a different level of granularity. If we train our model for end-to-end scene text recognition, each image crop is supposed to contain an entire word, which is then recognized by the recognition network. If we train our model only on scene text recognition, each image crop should contain at most one full character. The recognition network then combines the recognition results of all characters.

The recognition network extracts features using a deep convolutional feature extractor. We also chose to use a ResNet-based feature extractor because a

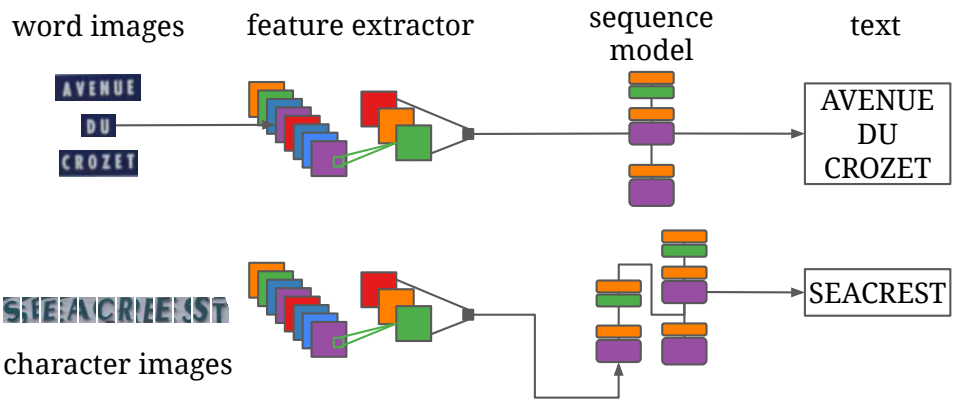


Figure 3.11: Our recognition network consumes text regions and predicts the textual content of each consumed text region. Our model consumes an entire word image for end-to-end scene text recognition and predicts all characters in each word image using a LSTM or bidirectional long short-term memory (BLSTM). We provide our model with cropped characters of parts of characters for our experiments on plain text recognition. We extract features and put the extracted features into the encoder of a transformer. Then, the decoder of the transformer predicts the textual content of the full sequence of character images.

residual network maintains a strong gradient to the first layer of a neural network. We argue that maintaining a stable gradient is even more critical in the recognition network. The gradient also has to be strong and informative when flowing through the recurrent spatial transformer into the localization network, which is solely trained by the gradient information obtained by the recognition network.

Following the convolutional feature extractor, we utilize a set of sequence recognition networks. On the one hand, we utilize a recurrent neural network similar to [207]. On the other hand, we propose to use a sequence generation network employing a transformer [229] (see also section 2.2.5).

Depending on the task at hand, the recognition network either produces a set of probability distributions for each character of each identified word region (end-to-end recognition) or a probability distribution for each identified text region that represents at most one character (text recognition only). In the end, the text recognition network predicts a probability distribution y for each character. The distribution y is defined over the possible label space L_ϵ, where $L_\epsilon = L \cup \{\epsilon\}$, with L being the set of characters to recognize and ϵ representing the blank label. We provide a structural overview of the recognition network in figure 3.11

Figure 3.12: Rotation dropout helps the network to predict reasonable rotation angles, helping the network to converge faster and better. The image on the left-hand side shows the localization result of a network trained without rotation dropout. On the right-hand side, we provide the result of a network trained with rotation dropout. In both cases, the recognition network can produce the correct text transcription. However, the recognition network performs worse in the model without rotation dropout.

3.6.3 Model Training

Our training set consists of a set of images and their corresponding text labels. Please note that we do not use any annotations for the location of words or characters different from related approaches. Both networks, localization and recognition, are trained using the recognition loss solely. We use softmax cross-entropy loss (see equation 2.4) between the predicted probability distribution y and the label distribution $\hat{y}$ for each character in the input image.

However, we observed that our proposed model does not converge if we do not add additional regularization terms to the training objective. On the one hand, we found that we need to constrain the prediction of rotation angles (see figure 3.12 for a graphical explanation).

On the other hand, we found it crucial to add further localization-specific regularization terms that directly operate on the predicted affine transformation matrices. In the following, we will explain all of our added regularization methods in more detail.

Rotation Dropout As mentioned earlier and visualized in figure 3.12, we found that the localization network tends to predict transformation parameters that include excessive rotations early on. Once the localization network predicts such excessive rotations, it never returns to predicting transformation matrices without excessive rotations. We argue that the network is stuck in a local minimum that it can not leave anymore. To mitigate such a behavior, we propose a mechanism to encourage less excessive rotations inspired by dropout [216]. We call our proposed method rotation dropout. Rotation dropout randomly drops the parameters of the affine transformation matrix, which are responsible for rotation, *i.e.*, θ_2 and θ_4. Randomly dropping

these parameters prevents the localization network from predicting excessive rotations while still allowing the network to learn to predict rotations.

Preventing Mirrored Sampling Grids Since we allow the prediction of arbitrary affine transformation matrices, the predicted sampling locations might sample the image at the correct location, *i.e.*, a location that contains text, but the sampled pixels might contain content mirrored on one or both coordinate axes. Such behavior is not desired for our use-case and hinders the convergence of our network. Thus, we propose to add a regularization term to the coordinates of the sampling grid. Given a sampling grid with sampling locations $((u_0, \dots, u_{W_o}), (v_0, \dots, v_{H_o}))$ with $u, v \in [-1, 1]$, we first extract the x-positions of the top-side of the sampling grid, *i.e.*, u_0 and u_{W_o}, as well as the y-positions of the left-side of the sampling grid, *i.e.*, v_0 and v_{H_o}. We then calculate regularization terms for sampling grids mirrored along the y-axis and x-axis respectively as:

$$\mathcal{L}_{\text{up\_down}}(V) = \max(v_0 - v_{H_o}, 0), \tag{3.3}$$

$$\mathcal{L}_{\text{left\_right}}(U) = \max(u_0 - u_{W_o}, 0), \tag{3.4}$$

$$\mathcal{L}_{\text{mirror}}(G_{\text{sample}}) = \mathcal{L}_{\text{up\_down}}(V) + \mathcal{L}_{\text{left\_right}}(U). \tag{3.5}$$

Here, U, V denote the vector of sampling locations on the x-axis and y-axis, respectively. The resulting regularizer penalizes grids based on their mirrored size. Thus, the larger the mirrored grid, the larger the regularization term.

Preventing Sampling Outside of the Input Image Related work [208, 209, 253] proposes to initialize the fully connected parameter predictor for the affine transformation matrices A_θ^n so that all weights are initialized to 0, and the bias is used to predict transformation parameters that lead to a sampling grid that spans most of the input image. In [208] Shi *et al.* argue that random initialization of the parameter predictor leads to failure of convergence during training. We, on the contrary, argue that random initialization of the parameter predictor does not lead to failure of convergence if extra regularization is employed. We even found that random initialization leads to faster convergence. To allow the network to converge when using random initialization, we propose to add an extra regularization term that penalizes sampling grids that would sample image locations outside of the input image. It is vital to prevent the network from sampling outside image locations because locations outside the image contain only zeros. Thus, the gradient obtained from sampling outside the image is zero, keeping the network from converging. To encourage the network not to predict transformation parameters that lead to sampling at locations outside of the input image, we propose an *out-of-image* regularizer, which we directly apply on the predicted sampling grid.

We know that each element u, v of the sampling grid has to be in the interval $[-1, 1]$ to be inside of the image. Thus we penalize any value of the sampling grid that is outside of this interval using the following regularizer:

$$\mathcal{L}_{\text{out\_of\_image}}(c) = |\min(c + 1, 0)| + \max(c - 1, 0). \tag{3.6}$$

With c being any value u or v from the sampling grid.

Learning Objective Using the described loss function and the introduced regularization terms, we can formulate our entire learning objective. The full objective consists of the cross-entropy loss $\mathcal{L}$ and all regularization terms can then be defined as:

$$\mathcal{L}_{\text{network}} = \mathcal{L} + \mathcal{L}_{\text{regularizers}}, \text{with} \tag{3.7}$$

$$\mathcal{L}_{\text{regularizers}} = \mathcal{L}_{\text{mirror}}(G_{\text{sample}}) + \sum_i \mathcal{L}_{\text{out\_of\_image}}(u_i) + \sum_j \mathcal{L}_{\text{out\_of\_image}}(v_j). \tag{3.8}$$

In figure 3.13 we show our full proposed pipeline that we use in our experiments.

3.7 Experiments

In this section, we evaluate our presented network architecture on a range of different tasks. We show that our proposed model is able to reach competitive and state-of-the-art performance on a range of standard benchmark datasets. Further, and most importantly, we show that our proposed network architecture can achieve excellent results when used for different tasks without any changes in the overall network structure. Our experiments are organized in the following way. First, we provide experimental results answering the question, whether our proposed concept is indeed able to learn to detect text in a weakly supervised manner. Second, we apply our proposed model for end-to-end scene text recognition on the FSNS dataset [215] (see also section 3.4.1). Last, we apply our model for the task of scene text recognition on all standard benchmark datasets introduced in section 3.4.2.

We begin this section by first introducing our experimental setup used throughout our experiments; we then show and explain the results of each of our experiments in detail.

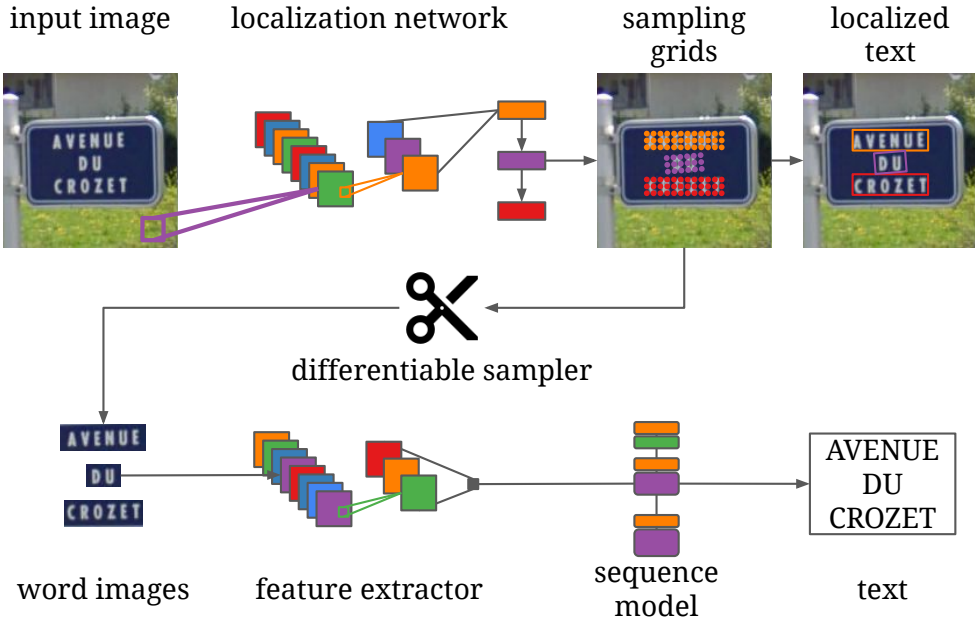


Figure 3.13: We train both parts of our proposed model together. We combine both models using the differentiable image sampling mechanism of spatial transformers. Although our model learns to extract the location of found text regions, we do not use any specific supervision for this part of our network. The character annotations of each image entirely supervise our network. Although we only show the case of end-to-end scene text recognition, the process is the same for plain text recognition.

3.7.1 Experimental Setup

We now describe our experimental setup, which includes used software, chosen network architectures for our experiments, and optimizers and hyper-parameters.

Localization Network The localization network we use in every experiment is based on the ResNet architecture by He *et al.* [94]. The input to the localization network is the image where either single words or single characters shall be localized. The input image may be input as an RGB image or a grayscale image.

For our experiments on the SVHN-like datasets and the FSNS dataset, we use the following network structure: We set the input size to 200×200 or 150×150 for our experiments on the SVHN toy datasets and the FSNS dataset, respectively. Before the first residual block, we use a 3×3 convolution, followed by a 2×2 average pooling layer with stride 2. After these initial layers, three residual blocks with two 3×3 convolutions, each followed

by batch normalization (BatchNorm) [109] (see also section 2.2.1.5), are used. The number of convolutional filters is 32, 48 and 48, respectively. Each convolutional layer is activated using the rectified linear unit (ReLU) activation function. We use 2×2 max-pooling with stride 1 following the second residual block. The last residual block is followed by a 5×5 average pooling layer and a BLSTM with 256 hidden units. After the BLSTM a fully connected layer with 6 neurons produces the affine transformation matrix that is used to generate the sampling grid for the following bilinear interpolation.

For our experiments on pure scene text recognition, we use a slightly different network. However, the overall architecture is the same. In this case, the input to the localization network is an image with a size of 200×64 pixels. As network architecture for our feature extractor, we use a standard ResNet-18 network, but we use group normalization (GroupNorm) [245] (see also section 2.2.1.5) instead of BatchNorm as normalization layer. Here, we also use a BLSTM with a 6-neuron fully connected layer to produce the affine transformation matrices to crop each character individually.

Following the prediction of the affine transformation matrices, we, in both cases, use rotation dropout and our localization-specific regularizers as described in section 3.6.3.

Recognition Network The inputs to the recognition network are N crops from the original input image that represent the text regions found by the localization network. The crops have different sizes, depending on the task at hand. During our SVHN toy dataset experiments, we set the input size of the recognition network to 50×50 pixels. For our FSNS experiments, we set the input size to 75×50 pixels. We set the size to 50×64 pixels for our experiments on pure text recognition. Here, the overall network structure is similar to the structure of the localization network. The network used in our experiments on the SVHN toy datasets has the same structure as the localization network. However, the number of learnable convolutional filters is higher. The number of convolutional filters if 32, 64 and 128, respectively. In our experiments on the FSNS dataset and our pure text recognition experiments, we use ResNet-18 as a feature extractor. In our experiments on the SVHN toy datasets, we follow Jaderberg *et al*. [112] and use an ensemble of N independent softmax classifiers to generate the character predictions. During our FSNS and text recognition experiments, we utilize a transformer to generate predictions, as described in section 3.6.2.

Alignment of Groundtruth Since we do not have any information about the locations of text regions, we assume that all given ground truth labels are sorted according to the western reading direction. Thus, we require the

ground truth texts to be annotated in the following order: (1) from top to bottom, and (2) from left to right. We want to point out that it is vital to have this consistent ordering of ground truth labels. If the labels are in random order, the network would not learn to find text regions sequentially.

Implementation We implement all of our experiments using the open source deep learning framework Chainer [225] and provide our code and models to the community, for further experimentation[5] Our experiments can be run on an NVIDIA GPU with at least 8GB of available video memory. We mainly used NVIDIA 1080Ti GPUs.

Optimizer and Hyperparameters For the optimization of our networks, we use different optimizers. For our experiments on the SVHN toy datasets, we use stochastic gradient descent (SGD) with momentum, set the learning rate to 0.00001, and multiply the learning rate by 0.1 every 5 epochs. We train our model for 10 epochs. For our experiments on the FSNS and text recognition datasets, we use RAdam [147]. Here, we set the learning rate to 0.0001, employ gradient clipping in the localizer and shift the learning rate by 0.1 every epoch. For our experiments on the FSNS dataset, we also employ curriculum learning [35], starting the training with simple samples containing at max 1 word and gradually increase the difficulty once the model converges. For our experiments on pure scene text recognition, we train the model to convergence. The model converges after roughly 3 epochs. However, this depends on the used training dataset.

We set the batch size to 128, 128 and 32 for our experiments on SVHN, FSNS and text recognition, respectively.

Data Augmentation During training, we utilize data augmentation and augment 40 % of the input images by randomly resizing them, applying blur, or extra perspective distortions.

3.7.2 Experiments on SVHN

The first experiments we perform are meant to show that the overall idea of our network architecture is feasible. To show this, we experiment on two synthetic datasets. Both datasets are based on the SVHN dataset. On the

[5]Our source code can be found in the following locations:

- Code for SVHN and FSNS experiments: `https://github.com/Bartzi/see` (last accessed August 31, 2021).
- Code for our text recognition experiments: `https://github.com/Bartzi/kiss` (last accessed August 31, 2021).

one hand, we created a dataset where four house numbers are arranged in a regular grid (as shown in figure 3.5 (left)). On the other hand, we randomly placed a maximum of two house numbers on a plain background (as shown in figure 3.5(right)).

First, we experiment on the dataset with regularly placed house numbers. In these experiments, we find that our idea indeed works. In the end, we are able to reach recognition accuracies of 99.5 % on our validation dataset. We note that the dataset itself is simple, which is why such accuracy is possible at all. However, during training, we make an important observation relevant to the following experiments. We observe that our model stops to improve the accuracy of the localization predictions after some time, while the recognition accuracy of the network is already good. We argue that this is because the recognition network has already converged and does not provide strong gradient information for the localization network to improve. We provide the result of training on our grid dataset showing this behavior in figure 3.14. We found a simple solution to this problem. Since we hypothesize that the recognition network already converged because the localization predictions are good enough, we restarted the model's training. During the restarted train run, we initialize the localization network using the weights of the previous run, while we initialize the weights of the recognition network randomly. Using this strategy (see figure 3.14 for the result), the predictions of the localization network improve, and thus the overall accuracy of the network improves significantly.

Following our experiments on the grid dataset, we experiment on the dataset with randomly placed house numbers. Here, we quickly observe that it is not possible to directly train a model to localize two house numbers simultaneously from scratch. Hence, we resort to the application of a curriculum learning strategy [35]. Our curriculum learning strategy is as follows: First, we train our model on images containing only a single randomly placed house number. Once our model converged, we use the pre-trained model to initialize the model we train on the dataset with two house numbers. Using this approach and our previously described restarting approach, we can train models that are able to localize and recognize randomly placed house numbers with a validation accuracy of 68.3 %. We do not use any annotations for the location of the house numbers. The network is forced to learn to localize the house numbers to succeed in recognizing them correctly. In figure 3.15, we provide visualizations of the train progress. We note that the necessity to train the model step by step is a weakness of our approach. However, we further note that our model learns to localize text in a weakly supervised way. Hence, we expect it to be challenging to train such models. Nevertheless, we show that using a clever training strategy it is still possible to train such a model.

Figure 3.14: Our experiments showed that restarting the training with a pre-trained localization network but a randomly initialized recognition network is essential for good localization results. The top image shows the localization results of our network when trained completely from scratch. The left-most patch shows the entire input image, including the predicted bounding boxes of text locations. The four images beside the left-most patch show the content of each of the extracted text regions. Here, we can see that the text is not accurately localized. The image on the bottom shows the same views, but this time produced by a model trained using our restart mechanism; the localizations are more accurate, and the network's overall accuracy is better.

Figure 3.15: We found that the localization of multiple randomly placed SVHN house numbers is only possible if the model is trained under a curriculum learning strategy. First, we train with all images where we only placed a single house number on a random location (*left*). Second, we add images with two house numbers and train on them (*middle*). We can repeat this step until we trained our model with the required number of house numbers per image. In the end, we can use the restart approach again and improve the quality of the localization predictions (*right*).

3.7.3 Experiments on the French Street Name Dataset

Following our experiments on the dataset with randomly placed house numbers on a plain background, we experiment with a real-world dataset. The real-world dataset we experiment with, is the FSNS dataset [215] (see also section 3.4.1). The FSNS dataset consists of more than 1 million images of street name signs extracted from Google Streetview. The images are only annotated with their textual content, making this dataset perfect for our proposed model. The dataset is challenging because it contains multiple lines of text with up to 6 words per image. To recognize each possible character, the alphabet of our text recognizer contains 133 different character classes and one class representing a "blank character." In contrast to our previous experiments, each street name sign is embedded in natural scenes, including distracting backgrounds. Furthermore, the dataset contains many images where the text is occluded, incorrect, or nearly unreadable for humans.

We found that our model does not converge when trained directly on the supplied ground truth during our experiments with the dataset. The annotations of the FSNS dataset are provided as single lines of text. Our proposed model is not able to learn to localize entire lines of text. Instead, we extracted single words from the annotations and trained our model to localize and recognize individual words. During the training of our model, we encounter similar convergence problems as described before. Thus, we also resort to a curriculum learning approach and train our full model in multiple stages. First, we train the model only on images with one word. Once this model converges, we reuse the trained model and train using all images with less than three words. We repeat these steps until all images are included in the training.

In table 3.1 we provide the experimental results of our model compared to other models. The evaluation metric used for the evaluation of the FSNS dataset is the sequence accuracy. Sequence accuracy measures the number of full-text lines correctly transcribed in relation to the overall amount of text lines.

The results show that our model can achieve competitive performance. We are not able to supersede the results by Wojna *et al*. However, our model is competitive, and our model can provide each word's location, which is a property none of the existing works have. In figure 3.16 we show some qualitative results of our FSNS model. Our qualitative results show that our model can reliably localize words and can handle distracting views.

Table 3.1: Recognition accuracies on the FSNS benchmark dataset in comparison to other state-of-the-art methods. The columns *localization* and *recognition* denote whether the approach delivers results for the localization task or the recognition task, respectively. Our model is, in contrast to related work, able to produce results for the task of scene text localization and scene text recognition.

Method	Accuracy (%)	Localization	Recognition
Smith *et al*. 2016 [215]	72.5	×	✓
Wojna *et al*. 2017 [244]	84.2	×	✓
Ours 2018	78	✓	✓

Figure 3.16: The samples taken from the FSNS dataset are diverse and challenging. However, our proposed approach can correctly recognize most of the given images, even if the input images contain a wealth of distractors. Our results show that our model can learn to localize text without any available localization information at training time. In the bottom-right image, we show a failure case of our model. Here, our model mixes the information of one view containing an incorrect street name sign into the information extracted from the other views containing the correct street name sign.

3.7.4 Experiments on Scene Text Recognition Benchmarks

In our last series of experiments, we apply our proposed model without any changes on the task of scene text recognition. Here, the task is to recognize the text in an already cropped word image directly. As indicated in section 3.3.2, early work on scene text recognition relied on first localizing single characters and then formulating the full word prediction based on the predictions of each character. Such an approach is a good choice because the task is broken down into several more straightforward sub-tasks to solve. We hypothesize that our proposed model can be used to recognize single characters, which are then grouped into single words. We are confident that our model achieves excellent performance because the model itself supervises the localization of individual characters. There is no need for us to formulate any post-processing methods for character proposals as done in related work [114, 238]. Furthermore, we hypothesize that our model does not need to learn to perfectly localize each character since we are using a sequence to sequence model at the end of the recognition model that allows us to relax the constraint that each extracted text region needs to contain a perfectly centered character.

Other than in our previously described experiments, we can train a single model directly from scratch without the need to use our retraining or curriculum learning approaches. In the following, we provide results of our model compared to state-of-the-art models and provide a detailed ablation study showing the influence of a range of design decisions on the performance of our model.

However, before we investigate results on specific datasets, we want to discuss the exciting learning process of our model. In figure 3.17, we show the learning process of our proposed model by providing a graphical representation of the model at several training iterations. In this image, we show that our model learns to localize characters/character regions without the need for further annotations. We can also see that both models can learn simultaneously. Another interesting observation is that our model learns to read the text from the back to the front.

3.7.4.1 Comparison to State-of-the-art Models

We compare our best model with several other methods on the datasets introduced in section 3.4.2. We train our best-performing model using the networks described in section 3.7.1. We set the dropout rate for rotation dropout to 0.95. We use an alphabet consisting of 95 different characters for the training of our model. This alphabet includes digits, case-sensitive letters from a to z, 32 symbols, and a blank character. For a fair comparison, we only train our model on publicly available synthetic scene text datasets. We also

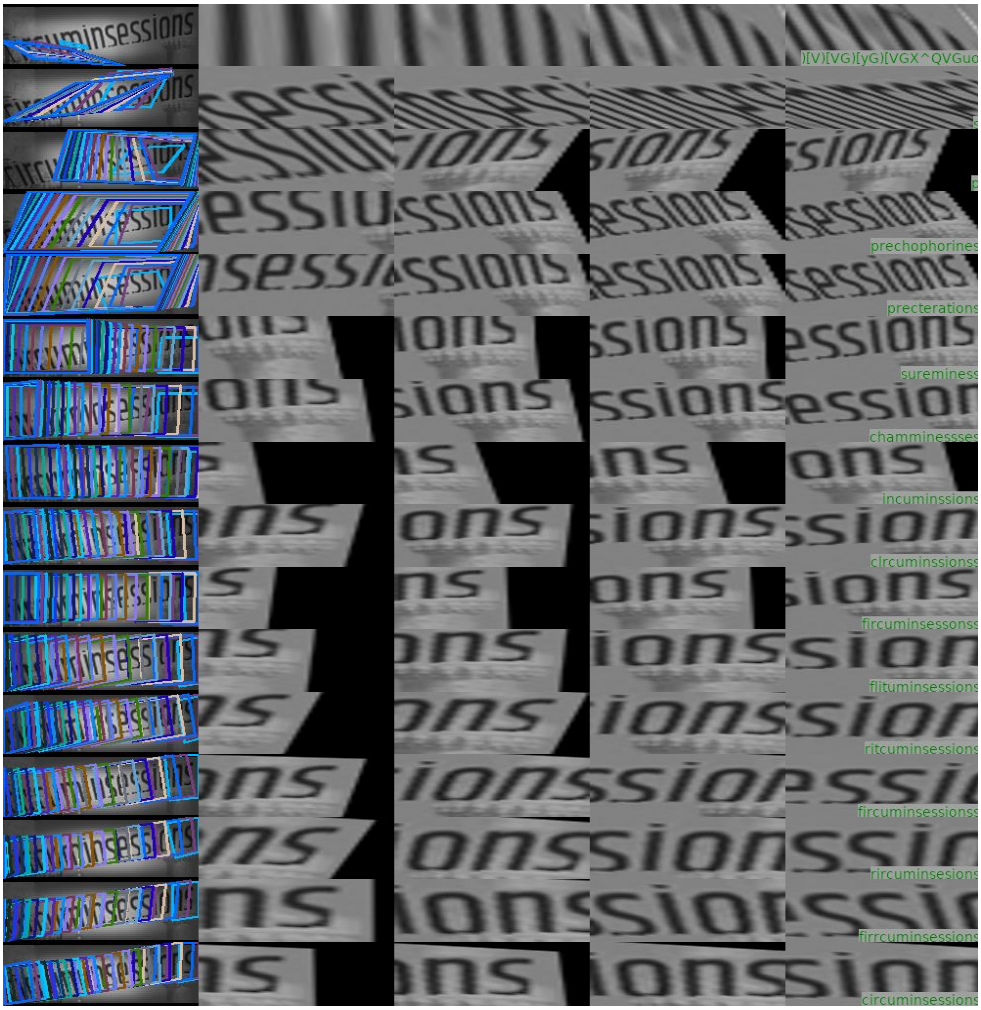


Figure 3.17: The learning process of our proposed text recognition model. Here, we show how our model learns to simultaneously localize and recognize text when trained on text recognition. The first row shows the predictions of the model at iteration zero. We show the input image with all predicted text regions (colorful boxes) on the left. In the following columns, we show the content of the first four predicted text regions. In the bottom right corner, we show the textual prediction of the network. The subsequent rows show the predictions of the network at the last step +1000 iterations. We can see that our model quickly learns to find the text. An interesting observation is that the model reads the text from back to front! We think this happens because reading from the back to the front makes it easier to decode the transformer's predictions in the recognition model. Best viewed digitally and in color.

Figure 3.18: The qualitative evaluation of our proposed scene text recognition approach shows that our model learned to follow the slope of words, even if they are curved. The top two rows show some correct results of our proposed model. The last row shows three typical failure cases of our model. We are certain that these failure cases can be alleviated by using more training samples with curved and severely degraded text instances.

only report those results of related work that were obtained when training solely on synthetic data. During evaluation, we follow [236] and perform data augmentation. We rotate each input image by ±5 degrees if the width is 1.3 times larger than the height; otherwise, we rotate the image by ±90 degrees. We then feed all rotated images and the original image to the trained network and use the prediction with the highest average score for each character as the output of our network.

We provide our results in table 3.2. Our quantitative results show that our model sets new state-of-the-art results, especially on irregular datasets. We note that our model is not able to correctly identify about 1.8 % of the word images in the IC15 dataset because these words contain characters not present in our alphabet. In figure 3.18, we show qualitative results of our approach. These results confirm that our model works well on distorted and heavily blurred text. The failure cases show that our model is not able to handle excessively curved text and exotic fonts. Both failure cases are most likely rooted in the lack of training data for such kinds of textual data.

Table 3.2: Results of our text recognition model compared to other models. Numbers in **bold font** and *italic font* indicate the best and second-best performance, respectively. We only report results that are obtained without any lexicon. It is possible to see that our model outperforms every other model on irregular text datasets, showing the general robustness to distortions of our model.

Method	IC13	IC15	IC15-1811	IIIT5K	SVT	SVTP	CUTE
Jaderberg *et al.* 2014 [112]	90.8	-	-	-	80.7	-	-
Shi *et al.* 2016 [207]	89.6	-	-	81.2	82.7	-	-
Shi *et al.* 2016 [208]	88.6	-	-	81.9	81.9	-	-
Liu *et al.* 2018 [150]	91.1	*74.2*	-	92.0	85.5	78.9	-
Cheng *et al.* 2018 [60]	-	68.2	-	87.0	82.8	73.0	76.8
Bai *et al.* 2018 [18]	*94.4*	-	73.9	88.3	85.9	-	-
Shi *et al.* 2019 [209]	91.8	-	76.1	93.4	89.5	78.5	79.5
Wang *et al.* 2019 [237]	-	-	-	90.5	82.2	-	83.3
Li *et al.* 2019 [140]	91.0	69.2	-	91.5	84.5	76.4	83.3
Luo *et al.* 2019 [161]	92.4	68.8	-	91.2	88.3	76.1	77.4
Liao *et al.* 2019 [143]	91.5	-	-	91.9	86.4	-	79.9
Wang *et al.* 2019 [236]	92.0	**74.8**	*79.1*	*94.2*	89.0	81.7	83.7
Zhan & Lu 2019 [253]	91.3	-	76.9	93.3	*90.2*	79.6	83.3
Liao *et al.* 2019 [141]	**95.3**	-	77.3	93.9	**90.6**	*82.2*	*87.8*
Ours 2019 (best configuration)	93.1	*74.2*	**80.3**	**94.6**	89.2	**83.1**	**89.6**

3.7.4.2 Ablation Study

To find the parts that add the most value to our proposed text recognition pipeline, we perform an ablation study, where we change several parts of our pipeline. We only change a single aspect for each experiment, which could either be a building block of the pipeline or changes to the training mechanism we apply. We show the quantitative results of our ablation experiments in table 3.3. All in all, we performed the following ablation experiments:

Changes to the Training Data In the first three experiments, we investigate the influence of the used training data on the performance of our model. In the first experiment, we used only the SynthText Dataset by Gupta *et al*. [90], which accounts for 41 % of the overall training images. The result (*SynthText Only*) shows that reducing the number of training images and also the variety of available images decreases the performance of our model by a large margin. A fascinating observation is that the performance decreases most on datasets containing mostly irregular scene text examples, *i.e*., the CUTE or SVTP datasets. This result indicates that the SynthText dataset includes mainly data that is well suited to recognize regular text.

Our second experiment investigates whether the result of the SynthText only experiment is only rooted in the lower number of available training data. Thus, we shrink and balance each dataset and keep a maximum of 200 000 samples per word length in each dataset. In the end, we are left with about 60 % of the overall training images. The results for this experiment (*Balanced Dataset*) indicate that using a lesser number of samples does not drastically decrease the performance of the model. Thus, we conclude that a well-balanced range of examples showing many kinds of scene text are essential for a deep model to be applied to a wide range of text recognition tasks.

In our third experiment, we complement the training data with data generated by our data generator. The results of this experiment (*All Data + Our Data*) clearly show that extra training data is beneficial for the performance of the network. Nevertheless, we also see that we can increase performance only on some datasets, not on all. The reason for these results is because our data synthesis tool only synthesizes data that is close to the data found in the IC13, IC15, and IIIT5K datasets, meaning that our synthetic data mainly contains regular words that are also highly distorted and blurred. Our synthetic training data misses rotated and curved samples. We are sure that additional rotated and curved samples would enable us to increase the performance of our model even further.

Changes to the Network Architecture In the next series of experiments, we determine the impact of several design decisions on the proposed method's

Table 3.3: Results of our ablation experiments. **Bold font** indicates the best performance for the corresponding dataset.

Variation	IC13	IC15	IC15-1811	IIIT5K	SVT	SVTP	CUTE
SynthText Only	90.4	65.8	71.1	89.2	82.4	73.3	74.7
Balanced Dataset	91.8	72.5	77.6	93.1	86.7	77.3	89.2
All Datasets + Our Dataset	**93.4**	**76.1**	79.5	93.7	89.2	81.1	84.7
Transformer Localizer	91.0	70.6	76.3	92.5	87.9	78.0	85.8
Softmax Recognizer	86.0	54.7	59.4	82.2	74.6	60.1	68.4
Transformer[2]	93.0	72.8	75.3	90.8	88.3	79.8	85.4
Recognition Network Only	90.9	67.5	73.6	91.5	85.0	74.6	78.1
Without Augmentation	92.7	72.9	75.4	90.8	88.3	79.2	87.2
Recognizer Optimizer	92.5	73.4	77.0	94.5	**90.0**	79.8	87.2
Best Configuration	93.1	74.2	**80.3**	**94.6**	89.2	**83.1**	**89.6**

overall performance. First, we exchange the LSTM that we use in the localization network with a transformer. This experiment is meant to answer two questions: (1) Is it possible to train a transformer without providing labels in the forward pass? (2) Does the usage of a localization network based on a transformer increase the performance of our proposed model? The results (*Transformer Localizer*) show that it is indeed possible to train a transformer without the need to provide labels in the forward pass. However, the training takes significantly more time because the sequence has to be rolled out repeatedly until the network predicted all locations. Furthermore, we can see that using a transformer in the localization network does not improve the overall results of our network.

In our second experiment of this series, we use a softmax classifier, as used in [112], instead of a transformer in the recognition network. The result (*Softmax Recognizer*) shows that using a sequence to sequence model, in our case a transformer, is one of the most crucial building blocks of our proposed model. We think that the softmax recognizer does not work very well because it highly depends on the accurate localization of individual characters. At the same time, our sequence to sequence model does not require such accurate localization because it can accumulate feature information over several time steps and can attend to all input features regardless of the character position.

In the next experiment, we increase the number of times we stack transformer layers in the recognition network. According to related work [229], a stack of multiple transformer models leads to the best performance on Natural Language Processing tasks. However, our results (transformer^2) show that using multiple transformer layers (we used two) does not increase the performance for our task. Hence, we conclude that it is sufficient to stay with the most straightforward transformer architecture, which also needs fewer computations and has fewer parameters to optimize.

In our last experiment, we propose not to use the localization network at all. Instead, we use a regular sliding window and only use the recognition network on the regularly applied sliding window crops. The results (*Recognition Network Only*) indicate that our proposed localization network is crucial for the correct recognition of curved and irregular text instances because we can observe that the performance on all irregular datasets drops by a large margin. This further confirms observations made in related work [207, 209, 253] where the authors argue that rectification is of high benefit for the task of scene text recognition.

Changes to the Training Method In our last series of experiments, we experiment with different changes to the overall training method. First, we train a model without using any data augmentation. Using no data augmentation (experiment *Without Agumentation*) leads to overall degraded performance.

While the performance decreases on some datasets by a large margin (IC15-1811, SVTP), it does not decrease as much on other datasets (CUTE, IC13). These results show that using data augmentation is vital on several datasets, hinting that the available datasets do not cover all necessary modes.

In the last experiment, we added another optimizer to the recognition network that is only trained on regular crops obtained by using a sliding window approach (similar to *Recognition Network Only*). The rationale behind this experiment is that the extra training might help the recognition network perform better, as it sees more diverse input data and also can learn faster since the input contains well-cropped regions from the beginning of the training. The results (*Recognizer Optimizer*) show that using this extra optimizer does not improve the results. However, using the extra optimizer also does not decrease them by a large margin, which indicates that such an extra optimizer could be beneficial if the input obtained from both input sources, *i.e.*, predicted crops of the localization network and crops obtained from the sliding window, matches well.

3.8 Discussion

The results of our experiments show that our proposed approach is a valuable addition to the landscape of scene text recognition approaches. We hypothesize that building on our proposed approach could lead to end-to-end scene text recognition systems that only need textual annotations instead of location annotations and annotations representing textual content at the same time. However, our experiments also showed that creating such a system on more challenging scene text datasets might not be that simple. In section 3.7.2 we already discovered that the system does not converge if we directly train on multiple words. A system applied end-to-end on more complex scene text datasets, *e.g.*, the SynthText dataset [90], needs to be trained under a sophisticated training curriculum.

In further work [30], we show that a similar architecture consisting of a localization network and an evaluation network can reliably detect objects in an image when trained under weak supervision. However, further experiments with the introduced approach for scene text detection proved to fail. We hypothesize that the guidance obtained from the recognition or evaluation network is not strong enough to be propagated through the 6 output parameters of the localization network.

All in all, we are sure that further research in this area should prove fruitful if it is possible to overcome the problem of missing guidance while backpropagating errors through the differentiable sampling mechanism.

3.9 Summary

In this chapter, we examined the question how we can lower the annotation cost by using only a fraction of the typically required ground truth annotations, by showing an example from the area of scene text recognition systems. We presented a novel weakly-supervised method, which can localize and recognize textual content from scene images. In contrast to existing previous work, our method is trained only using textual annotations; we do not require any location annotations for scene text localization. Not requiring ground truth annotations of text locations should make it possible to use such systems for new applications that do not have much annotated data available since annotating only the textual content is cheaper than annotating text locations. To this end, we propose to build a system consisting of a localization network with a recurrent spatial transformer and a recognition network using either individual softmax classifiers or a transformer for the generation of textual content. Both networks are jointly optimized. The localization network receives its gradient information from the recognition network. Our experimental results on end-to-end scene text recognition and explicit scene text recognition show that our model is applicable to multiple areas of scene text recognition without adaptation. Our results also show competitive and state-of-the-art performance on several public benchmark datasets. Our method is also the first method that is able to deliver information about the location of texts without being explicitly trained or supervised to do so. However, although our proposed method is promising and shows good results on real-world data, we argue that scaling to more complex scene text datasets is challenging. In the future, a thorough examination of the learning challenges and design of a clever curriculum learning strategy should be performed to reach our goal of learning to detect and recognize text without having to specify the locations of the text.

4 Synthetic Data for Handwriting Analysis in Archives

In this chapter, we examine further strategies to lower the cost and burden of obtaining ground truth annotations and show solutions for the use case of handwriting analysis in historical archives. Here, we examine the value and applicability of synthetic data to develop computer vision methods for pre-processing methods of an optical character recognition pipeline. To this end, we propose novel data synthesis mechanisms. We further introduce novel challenging analysis tasks that have, to best of our knowledge, not yet been in the focus of research. In particular, we propose solutions to the following analysis tasks: (1) A data synthesis tool to use for the training of a classification model to determine whether a scanned page or region of a scanned page contains any handwriting at all. (2) A novel approach to classify what kind of content a cropped handwritten or printed word contains, *i.e.*, is it a number, a date, or an alpha-numeric string. (3) A novel data synthesis system to synthesize training data for the training of pixel-wise semantic segmentation systems for document analysis. Parts of this chapter have been elaborated in the following publications: [27, 28, 29].

4.1 Motivation

Archives contain the knowledge and wisdom of generations of scholars. Most of the information in an archive is contained in documents in written form, either written by hand or printed. Although measures are taken to prevent the quality of documents from degrading, the digitization of archives is necessary to preserve the content for generations to come. However, digitization does not only allow the conservation of documents; it also opens the door for computerized analysis. Modern computer vision algorithms enable us to automatically read and understand the content of millions scanned pages [40, 214, 242]. Following the extraction of the textual content of scanned pages, further analysis steps that help to find connections between documents, such as named entity recognition [116] or knowledge graph construction [115] can be employed.

Together, the extraction of textual content from raw document scans and further analysis of the extracted texts form a document analysis pipeline where

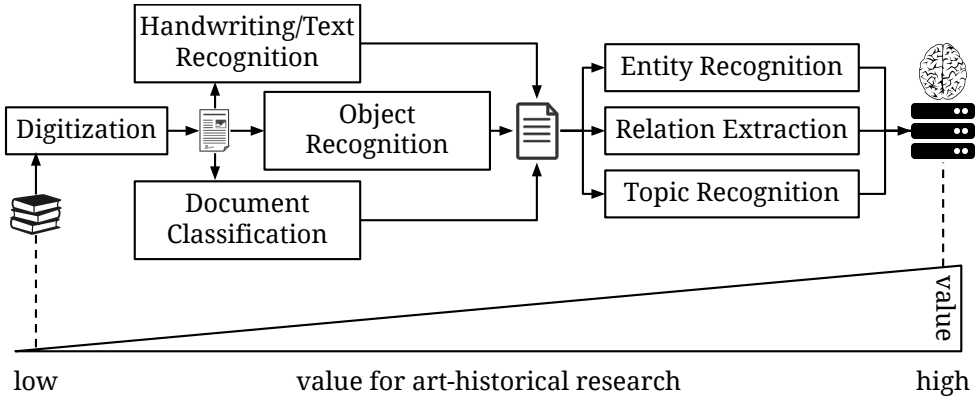


Figure 4.1: Schematic overview of a document analysis pipeline including several analysis steps that increase the value of the data for art-historical research. In this thesis, we are mainly concerned with the tasks of text and handwriting recognition. All analysis steps that follow directly after the digitization are analysis steps that apply computer vision on scanned pages. Following analysis steps use the semantic metadata extracted via computer vision methods for further information retrieval.

each step adds more value to the research of historians. See also figure 4.1 for a quick overview of possible analysis steps. The value for the research of historians is added by allowing researchers to query and find the information they are searching for effortlessly. Tools for automated analysis might even help to highlight interesting information that leads to further discoveries.
Such an entire archive analysis pipeline can be arbitrarily complex. The focus of our work lies in the identification and extraction of textual content from raw scans of documents. Here, we are focusing on the development of analysis approaches of the extensive digitized art-historical archive of the Wildenstein Plattner Institute (WPI).[1] The WPI provided excerpts of their digitized art-historical archive to us to extract textual data from the scanned images. At a first glance the task seems to be simple to perform because there is much related work in the area of handwriting analysis, *e.g.*, [44, 131, 164, 175, 182, 190, 212, 242], as well as mature solutions for the analysis of printed text, see section 2.1. However, most of the existing approaches are not directly applicable to the raw data extracted from a large archive such as the archive of the WPI due to the following reasons: (1) Most of the proposed approaches are only focused on solving a task where the input data adheres to well-defined constraints, such as documents containing solely handwriting, documents that only contain images, *etc*. The data found in a large archive that spans over centuries is very diverse. Thus, pre-processing methods need to be developed that can

[1] `https://wpi.art/` (last accessed August 31, 2021).

filter the data based on the requirements of existing approaches, making it possible to apply existing solutions. (2) Recent proposed solutions, such as [44, 175, 182, 242], use deep learning for the analysis of cultural heritage data. However, using deep learning requires the availability of annotated training data. Most archives do not contain any annotated data. Hence, it is impossible to directly apply previously developed solutions based on deep learning without the means to obtain annotated data. One possible solution to obtain a large amount of the required annotated training data is to use manual annotation, possibly supported by the crowd. One example of this approach is the transcription of the Jeremy Bentham Collection, where manual transcription on 95 000 scanned pages is performed. Starting in 2010 the project is expected to finish earliest in 2025 [56]. Looking at millions of pages alone in the archive of the WPI, it is not possible to perform such a full transcription by hand. Even a transcription of only parts is not feasible because it would be very costly. Automated synthesis of training data or the development of unsupervised/weakly-supervised approaches would be of great benefit for the analysis of such archives. If we were able to develop an analysis method that directly learns on the available unannotated data of one archive, we could adapt the method to other archives and perform analysis there without the need to perform costly and time-consuming manual annotations!

Thus, the work presented in this thesis focuses on developing pre-processing methods that help transform or filter the available data into the input formats required by already existing related work. While developing such pre-processing methods, we use synthetic data in each step, as we do not have access to any annotated data for the training of our deep neural networks.

The data we want to analyze consists of a mixture of documents containing only printed text, only handwritten text, mixed printed and handwritten text, and images. See figure 4.19 and figure 4.20 for some typical examples of the documents we want to analyze. To identify the correct analysis method, we first develop an approach for determining whether a provided scanned page contains any handwriting at all. We base our approach on a deep neural network solely trained using artificial data. See section 4.3 for a full description of our developed system. Since there are no handwriting recognition tools available that can be applied to handwritten data of several writers, we further focus on developing tools to aid handwriting recognition. On the one hand, we propose a novel pre-processing step that operates directly on extracted handwritten words. Based on the visual structure of the image containing handwriting, we classify whether the image contains a date, a number, an alpha-numeric string, or a word. Such a pre-processing step can help to identify the best fitting specialized handwriting recognition model. Furthermore, such

a classification could already be of use for researchers interested in finding all pages containing a specific kind of content. Thus, our proposed approach could be seen as a vision-based named entity recognition (NER) approach. We provide more details on our proposed approach in section 4.4.

Last, we investigate the field of semantic segmentation of document images. Specifically, we are interested in developing a method to distinguish between pixels that belong to printed text and pixels that belong to handwritten text. To this end, we propose a novel data synthesis method that directly operates on the available but unannotated data and is able to synthesize training data for the training of an off-the-shelf segmentation model. More details on our proposed data synthesis method for segmentation can be found in section 4.5. Before we explain our contributions to several parts of an analysis pipeline for historical documents, we introduce related work in the field.

4.2 Related Work

The first approaches for optical character recognition (OCR) were developed for the analysis of post or office documents, as indicated in section 2.1. It was until the early years of 2000 that OCR for analysis of digitized historical documents, *e.g.*, found in libraries and archives [1], came into the focus of research. With the digitization of archives all over the world, the automated analysis of historical documents got more and more critical. Research in the field of historical document analysis focuses on all parts of a traditional document analysis pipeline (see section 2.1.1). First, documents are digitized and roughly categorized. Then, preprocessing is performed, the textual content is recognized, and finally post-processing is applied. While research mainly focused on recognizing printed and handwritten text in the early days, research now also incorporates the automated analysis of the age or direct classification of the document class, *i.e.*, letters, book pages, *etc*. In the following, we examine related work found in each part of a historical image analysis pipeline that we contribute to.

4.2.1 Categorization

After digitization, documents are stored in a digital archive. To the computer, a raw digitized document is a collection of numbers without any meaning. Semantic analysis is necessary to index and use the digitized data in a proper way. Before a document can be analyzed, some information about that document should be obtained. Such information allows to assign the necessary analysis steps and helps to process the various possible document types efficiently. Here, research focuses on the identification of writers [5, 62, 119],

the automatic prediction of manuscript dates [45, 52, 91, 92, 96, 97], or the classification of the page content [4, 6, 7, 13, 19, 20, 57, 67].

While early work uses handcrafted features [5, 45, 52], all later approaches make use of deep neural networks. Here, various approaches are used and applied, ranging from deep neural networks initially designed for image classification [6], composite systems including object detection and image classification [62], to systems applying OCR and image classification [13] (although in this case, we might not be able to refer to the method as a preprocessing step anymore). One common property of all of the proposed models is that they are trained using manually annotated training data, which is costly to obtain. Furthermore, each proposed approach targets a specific kind of documents, *i.e.*, Bengali documents in [5], medieval scripts in [92], or contemporary documents in [6].

While there is a wealth of work for the categorization of documents available, the task of determining whether a scanned page contains handwriting has, to the best of our knowledge, not been tackled yet. Besides introducing a new preprocessing step for a document analysis pipeline, we also investigate the possibilities of using synthetic data. Here, we draw inspiration from related work in the field of scene text detection and recognition [90, 112] and propose a synthetic data pipeline for our task. We are confident that similar data synthesis approaches can be applied to related work at this stage of a document analysis pipeline to improve the scalability and adaptability of proposed approaches to novel and unseen data.

4.2.2 Segmentation

Another part of a document analysis pipeline that we contribute to is layout analysis and segmentation. Layout analysis deals with the localization and classification of the organization of a given document. Layout analysis includes the segmentation of full pages [218] localization and recognition of tables [64], the segmentation of individual text lines [44, 131, 182], or baseline detection [53, 76, 89, 169].

All recent work is based on the application of deep neural networks to the task at hand. For the segmentation of entire pages, Stewart and Barret [218] introduce a fully convolutional neural network for the segmentation of marriage certificates and birth records. They train their model on a single manually annotated training image to segment handwriting, stamps, or printed text from the background. Their proposed model shows promising performance on homogeneous data such as marriage certificates from a defined region and time. However, such an approach does not scale to an archive consisting of millions of heterogeneous documents.

Oliveira *et al*. [182] introduce a layout analysis system based on a ResNet-50 [94] feature extractor that borrows elements from the popular U-Net [197] architecture. Their system can be applied for page detection, page segmentation, and text line/baseline extraction. Kiessling [131] introduces a pixel-wise segmentation network for the extraction of text lines. Another recent work by Boillet *et al*. [44] introduces a simpler segmentation network, compared to the network introduced in [182], which they pre-train on multiple datasets, followed by fine-tuning for the dataset the model is to be applied on. Different approaches apply layout analysis and pixel-level segmentation on images of historical Vietnamese steles [204] or using siamese networks on historical Arabic manuscripts [8]. These approaches return a pixel-level segmentation of the entire input image. The pixel-level segmentation can then be used for further analysis steps, such as word and line segmentation. Further approaches that are closely related to pixel-level segmentation approaches, are approaches binarizing challenging documents [123, 134, 173]. Here, the task is to distinguish text pixels from background pixels. Approaches for pixel-level segmentation, binarization or text line segmentation, use variants of the U-Net [197] architecture. The U-Net architecture is mostly adapted by using different network backbones [182], adding dilated convolutions [44], or cascades of U-Net networks [123].

However, all these approaches rely on manually annotated data, which is costly to obtain. Especially for segmentation, it is crucial to have enough annotated training data available because the network needs to be adjusted to the data distribution of the data it is to be applied on. Our contribution in this field is a novel method for synthesizing training data, which can then be used to train already proposed document segmentation or binarization methods.

4.2.3 Recognition

The main objective of many research endeavors in historical document analysis is the recognition of the textual content of a document. After the segmentation of a document, including the identification of text regions, the next step in a document analysis pipeline is the recognition of the text. To this end, we distinguish between the recognition of printed text and handwritten text. One could say that recognizing printed text is solved, at least for documents with printed text from the last centuries. Modern print OCR solutions such as Tesseract [214] can read a wealth of documents written in a multitude of fonts. However, the recognition of older printed text still poses a challenge [63]. In our work, we are mainly interested in the analysis of handwriting. Thus, we further focus on the analysis of handwriting.

Over the years, a wealth of systems for the recognition of handwriting have been proposed [72, 73, 137, 166, 175, 190, 242, 246]. While early methods used handcrafted features [165, 166], later systems quickly adopted multi-dimensional recurrent neural networks [72, 73, 87], which are now mostly superseded by the usage of convolutional neural networks with sequence to sequence encoders and decoders [190, 246]. Nowadays, offline handwriting recognition systems [246] reach word error rates of as low as 8.6 %, or 7.5 % on the standard offline handwriting recognition benchmark datasets IAM DB [165] and RIMES [14], respectively.

Although handwriting recognition systems already achieve promising performance, several challenges relating to analyzing historical documents and documents in archives remain. Especially the recognition of old scripts holds its challenges. Historical handwritten Japanese characters, for instance, pose a challenge to offline handwriting recognition systems. On the one hand, it is challenging to segment individual words or characters from text lines. Thus, approaches based on object detection try to cast the text recognition problem as an object detection problem [222]. On the other hand, not much annotated training data for the recognition of old scripts is available, which makes it necessary to use artificial methods to synthesize annotated training data [137]. The Transkribus project [175] provides a platform for the creation and usage of handwriting recognition models for several kinds of handwriting data from different centuries. However, the Transkribus project relies on manual annotations, proposing to obtain annotations of at least 5000 words before a handwriting recognition model with reasonable accuracy can be trained. Access to methods that automatically generate annotations based on the available unannotated data would be of high benefit for projects, such as the Transkribus project.

Another problem that has not yet been addressed is how to handle various kinds of handwritten text, *e.g.*, numbers, words, dates, or price labelings. A further problem that has not yet been addressed is the analysis of a corpus consisting of documents in multiple languages. As long as not enough training data for a specific language is available, it would be good to reason about the possible content of a word image without the need to recognize it entirely. Few related works dealt with directly classifying a word image solely based on its visual structure. Mandal *et al*. [164] proposed a system to identify and locate dates in documents. While they focus on finding date tokens, they first recognize the text of individual text lines before the identification of dates. Ingle *et al*. [108] propose to use a deep neural network to directly predict the style, *i.e.*, handwriting or printed text and script, *i.e.*, Latin of a given word or text line image. So far, no approach to directly propose the class of a word has been proposed.

4.2.4 Synthesis of Documents

The availability of annotated data is a massive problem for the analysis of documents, especially historical documents. When using modern data-driven approaches, the only chance to create a well-performing model is to use a large amount of training data. However, as already stated before, the acquisition of annotated training data is costly, especially if the data has to be annotated by hand, especially if the data can only be annotated by domain experts. If we look into other areas of text analysis, especially the area of scene text detection and recognition [90, 114], we can see that a viable solution to the training data problem is the synthesis of training data.

The usage of synthetic data is also a reoccurring pattern in the area of document analysis. Tools for semi-automatic creation of annotated training data are available [120]. However, such tools require a large amount of manual work to synthesize a large-scale training dataset. Other approaches utilize data augmentation techniques to augment already existing data with new content [126], or image degradation techniques [79, 137]. However, a vast archive containing documents from multiple centuries contains a diverse set of document types. Even if we were to annotate documents of each type, the whole process would still be very costly and might not be applicable to documents of other archives. Hence, we strive to find data synthesis methods applicable to any data without much human intervention. The application of image degradation techniques also relies on the capabilities of the researchers to correctly identify degradations in real-world data. Following such an approach is similar to the handcrafting of features, which the application of deep learning has superseded because methods on deep learning can better capture the subtle structures and distributions of the underlying data than their handcrafted counterparts.

Recently, novel methods based on the application of generative adversarial networks (GANs) for data synthesis have been proposed. The proposed methods can be classified based on their granularity level. On the granularity level of entire pages, a range of approaches are utilizing the image-to-image translation capabilities of CycleGAN [258]. The goal of these approaches is to synthesize realistically looking training data based on simple computer-generated documents. Bui *et al*. [51] utilize CycleGAN and MUNIT [105], another image-to-image translation GAN, for the transformation of computer generated receipt images to more realistic looking receipt images. Another work by Tensmeyer *et al*. [223] uses CycleGAN for the synthesis of document patches with their corresponding binarization ground truth. Tensmeyer *et al*. also first synthesized the ground truth using manual methods, then they use CycleGAN to refine the synthetic image and create a more convincing and

domain-adapted version of the training image. Pondenkandath *et al*. [189] go even a step further and use CycleGAN to transform entire documents from the domain of modern printed documents to historical handwritten documents. Their method first requires synthesizing printed documents with the correct layout, *e.g*., using LaTeX. Then, they use CycleGAN to transform the printed documents into handwritten documents. These approaches first require the production of synthetic documents using handcrafted algorithms, followed by a refinement or adaption step to transform the synthetic data to the target data domain. Vögtlin *et al*. [231] improve on the idea by Pondenkandath *et al*. by adding text recognition losses to encourage the model to produce content closer to the original data domain and keep the semantic content of the synthesized document. We also utilize handcrafted algorithms in our work. However, we propose a novel approach that does not require any handcrafted algorithms to obtain the ground truth; we directly extract the ground truth from the trained network. Hence, our proposed system is able to work directly on the target data domain without the need for domain transformation.

The usage of CycleGAN also plays a role in the granularity level of single words or text lines. Rusakov *et al*. [199] use CycleGAN for the synthesis of more realistic cruneiform signs compared to simple rendered cruneiform signs. They also follow the approach of first rendering the plain cruneiform signs and then transferring them to the target domain using CycleGAN. Alonso *et al*. [9] utilize a conditional GAN for the synthesis of handwritten words. They condition their GAN on the word to be written. Besides a discriminator to encourage the trained model to synthesize realistic images, they propose to use a text recognition network to guide the network in the synthesis of readable handwriting. Building on the handwriting synthesis system by Alonso *et al*., Kang *et al*. [121] propose a handwriting synthesis network that can produce even more realistic handwriting. The model of Kang *et al*. is furthermore versatile compared to the model of Alonso *et al*. because it can additionally be conditioned on a particular style of handwriting. In our work on handwriting classification, we use the model by Kang *et al*. to synthesize realistic handwriting.

4.3 Determining Pages that Contain Handwriting

Handwriting is a ubiquitous entity in historical documents. Thus, methods to analyze handwritten content need to be developed. However, as stated before, it is costly and complicated to obtain the necessary annotations to use state-of-the-art methods. In the following, we examine our contributions to historical document analysis pipelines with a particular focus on the usage of artificial data. In this section, we introduce our approach for the pre-processing task

of determining whether a page contains handwriting. The work presented in this section has also been evaluated in [29].

4.3.1 Motivation

In our work, we collaborate with art-historians from the WPI. The WPI possesses a large digitized archive containing more than 10 million scanned pages. The scanned documents found in the archive of the WPI include scans of auction catalogs, correspondences of art dealers, *etc*. Handwriting is of particular interest to the researchers of the WPI because it might contain valuable information about the provenance of works of art, prices at auctions, or hints about the author of a specific work of art. The automatic extraction of handwritten information from the archive would help the researchers in their daily research work.

As a pre-processing step, we propose an analysis step that has not been used in related work before. We propose to add a new step to a document analysis pipeline that examines each page whether it contains handwriting. The benefits of adding such an analysis step are manifold. First, performing such an analysis on the entire archive provides researchers with information about possibly interesting pages. Second, whether a document contains handwriting or not can be used to determine which parts of the analysis pipeline need to be run; this should help conserve processing time and keep false analysis results to a minimum. Third, we found that using our developed handwriting determination approach is helpful for the development of approaches to solve problems further up in the pipeline. We use our developed handwriting determination approach, for instance, to balance our training data for our semantic segmentation approach presented in section 4.5.

To this end, we propose a data synthesis pipeline that synthesizes patches of documents. Then, we use the synthetic data to train a classification model that is applied on patches of the original documents. The classification model classifies whether a given patch contains handwriting. In our experiments on a diverse dataset containing real data from the WPI, we found that our model generalizes well to real data with a F1-Score of 0.97. All code and models that were used to achieve the results presented for this approach can be found online.[2]

4.3.2 Determining Handwritten Pages with Synthetic Data

Raw digitized archival data does not necessarily contain any information about the content of a document. However, the organization of the data in a datastore

[2] https://github.com/Bartzi/handwriting-determination (last accessed August 31, 2021).

might contain information about the category or topic of the document. Names of directories or files could be used to gather coarse information about the document at hand, but this high-level information is not sufficient to find the required data in a fast and straightforward way. This is why the extraction of semantic metadata, preferably using automated methods, is necessary. Now, we introduce our first contribution to aid the automated analysis of historical documents. We propose a novel analysis step to determine whether a given page of a document contains handwriting. We train a deep neural network to classify whether an extracted patch of a document contains handwriting. We utilize synthetic data that we synthesize using our proposed data synthesis tool to train our deep neural network.

We train our proposed model on patches of scanned pages. The decision to use patches of documents instead of full documents is rooted in two observations. First, we found that it is not easy to automatically synthesize full document images that are diverse, with a clear visual structure. Second, since the goal of digitization is the preservation of historical material, the digitized material is saved with the highest possible quality and resolution. We would soon run out of memory if we were to use deep neural networks on such high-resolution data without any pre-processing. Hence, it is necessary to reduce the resolution of the input images. Resizing a high-resolution image to an image with a considerably smaller size could lead to considerable detail and information loss. Tiny handwritten annotations might not be visible anymore. Therefore, we decided to split each input image into patches. Splitting an image into patches allows us to keep the original resolution of the input image in each patch; it also allows us to provide the user with a rough location of handwriting regions. First, we introduce our data synthesis method. Second, we explain the deep neural model for classifying whether a scanned page contains handwriting.

4.3.2.1 Data Synthesis

At the heart of our approach is a data synthesis tool. Following related work (see section 4.2.4), we create a data synthesis tool that is based on observations on the structure of the data. Our tool synthesizes data similar to real data and provides us with a training image with the corresponding annotation to train our classifier.

It is challenging to synthesize an entire document page since it is nearly impossible to anticipate all possible layout combinations, create a convincing intra-document hierarchy, and keep the synthesized document consistent, especially when working with documents from archives containing documents from several centuries. Thus, instead of whole pages, we synthesize patches. Synthesizing patches is more straightforward because we do not need to

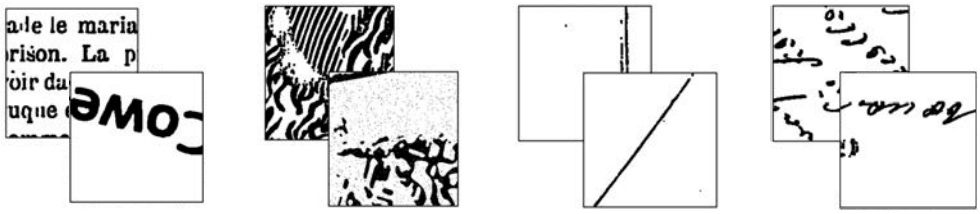


Figure 4.2: Examples of properties rendered on patches. A final patch consists of multiple properties rendered on a patch. Here, we show each property isolated to make the differences between them visible.

account for the composition and layout of multiple components of a single document while still synthesizing a diverse set of layout combinations, as they occur in real data. In short, we can say that we chose patches because we are confident that patches allow us to model the data distribution more simply. Each of the patches we synthesize contains several properties we identified as integral parts of real scanned pages. Our data synthesis result is a binarized image that contains one or more of our identified properties. Synthesizing binarized images allows us to assemble document parts from multiple sources into a single image without adjusting the background to receive uniform-looking samples. In the following, we describe the properties we identified and how we synthesize our patches.

Properties of Patches We found that pages of scanned documents exhibit one or more of the following properties: (1) Printed text in the form of multiple columns, with many words, as well as single printed words. (2) Graphics, *i.e.*, reproductions of works of art, or sketches. (3) Scanning artifacts, *i.e.*, a black border around the image, scanning noise, *etc.* (4) Handwritten texts or single words. In the following, we perform a closer examination of these properties showing how we model them with our data synthesis tool. In figure 4.2, we depict some examples of the properties found in our synthetic dataset.

Printed Text Over time, a wealth of printed text has been produced by many people in various countries around the earth. These printed texts appear in various layouts, using different fonts and writing styles. To account for a large variety and cover a wide range of eras, we included texts from the German Bundestag[3] and the archive of the US government.[4] The archives contain documents from the years 1949 to 2021 and 1793 to 2021, respectively. Besides extracting printed text from these documents, we also add printed

[3] https://dip.bundestag.de/ (last accessed August 31, 2021).
[4] https://www.govinfo.gov/app/search (last accessed August 31, 2021).

words to our synthetic data. We add printed words because isolated words might appear on postage stamps or labels contained in a scanned document. We synthesize printed words using a tool for the synthesis of training data for the recognition of printed text.[5] Before we render printed text on a patch, we binarize the used text portion using Otsu thresholding [183] to mitigate problems with different background colors when assembling our synthetic patches.

Graphics Documents from an archive might contain documents with graphical elements such as pictures or sketches. Graphics contain organic-looking structures, especially after binarization, which might confuse our model because these structures are very similar to handwriting. To mitigate false-positive predictions of graphical elements, we add photos from the validation dataset of the COCO 2017 Object Detection validation dataset [145]. On the one hand, we transform them to grayscale and add them directly to the patches. On the other hand, we binarize the images using adaptive gaussian thresholding.

Scanning Artifacts The scanning of a document produces a range of visual artifacts. A common artifact is the area around the actual document produced by the used scanning device and can be found in many scans. We include such artifacts by adding binarized excerpts of scanning borders to the produced patches. We also simulate other artifacts such as noise by applying pepper noise to some of our synthesized patches.

Handwriting The last property is handwritten information. An archive might contain documents with different amounts of handwriting. An archive might either contain documents entirely written by hand, *e.g.*, letters, documents with a mixture of printed and handwritten text, *e.g.*, handwritten notes in auction catalogs, or a mixture of handwritten text and graphics. In the case of entirely handwritten documents, the document's structure is similar to a document containing only printed text, as they often contain multiple lines and blocks of text. To mimic such data, we included parts of real handwritten documents. Here, we used images from the Folger Shakespeare Library.[6] Besides handwritten passages, archival documents might also include short annotations. To mimic such data, we include data from the IAM offline handwriting recognition dataset [167], which contains a wide variety of handwriting styles and words. We binarize all handwriting using Otsu thresholding before rendering it onto a patch.

[5] https://github.com/Belval/TextRecognitionDataGenerator (last accessed August 31, 2021).

[6] https://luna.folger.edu/luna/servlet/FOLGERCM1~6~6 (last accessed August 31, 2021).

Synthesis Process During data synthesis, we combine the properties described above and produce our synthetic training samples. Our synthesis process includes the following steps:

Property Selection In the first step, we decide which properties shall be used in the patch to synthesize. We randomly select 0 to n properties, with n being the number of all properties minus handwriting. If a patch should be a positive patch, *i.e.*, contain handwriting, we also either select a handwritten document or a handwritten word from the IAM dataset to be included in the patch.

Fragment Synthesis After selecting properties, we create an empty image serving as a canvas. As already stated above, we produce binarized images to avoid problems with different stages of background degradation, which would destroy the uniform look of our synthesized data and provide unwanted visual cues for the classification network. First, we select each described property and binarize the selected property using the binarization approach described in the description of each property. Following binarization, we apply a range of 0 to m random transformations, such as scaling, rotating, dilating, or eroding the property. Here, m denotes the number of available transformations. Applying these transformations allows us to add more variety to our data and model the data distribution more closely. Finally, each property image is added to the canvas and passed to the next step.

Fragment Evaluation After synthesizing a fragment, we perform a quality assurance step if the patch is supposed to contain handwriting. In this quality assurance step, we count the number of black pixels containing handwriting visible on the synthesized patch. Counting the number of black pixels that belong to handwriting is simple because we know where we pasted handwriting and can use this information to count all black pixels of our binarized image. This check is necessary because the documents we use as handwriting sources contain regions with little or no handwriting. Our quality assurance step allows us to discard all fragments that do not contain "enough" handwriting. During synthesis, we require that at least 5 % of the pixels of each patch with handwriting contain black handwriting pixels.

In figure 4.3, we provide a schematic overview of the proposed data synthesis pipeline, including examples of synthesized patches.

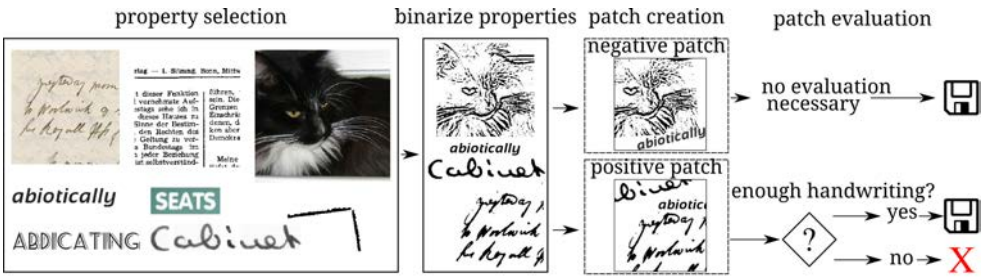


Figure 4.3: Illustration of our data synthesis pipeline, which consists of several steps. First, we select several properties from our pool of available properties. Then, we binarize the image of each selected property. Following the binarization, we assemble multiple properties into a negative or positive patch. A positive patch contains handwriting, and a negative patch does not. Following patch creation, we perform a patch evaluation step. The evaluation step is only necessary if we create a positive patch because we want to ensure that the fragment contains enough handwriting. This evaluation step helps us to provide only reasonable input to the classifier during training. In the end, we save the patch.

4.3.2.2 System design

After synthesizing our training data, we use the data to train a deep neural network on handwriting determination. We propose to use a deep convolutional feature extractor with a two-class classifier. For the extraction of features, we resort to a feature extractor based on the ResNet-18 [94] architecture, where we use group normalization (GroupNorm) [245] instead of batch normalization (BatchNorm) [109] (see also section 2.2.1.5). Thanks to the large amount of data we synthesize, we can train our model directly from scratch without the need to fine-tune a model initialized on the ImageNet dataset [71]. We train our model using stochastic gradient descent and utilize softmax cross-entropy as the loss function.

The input to our network is a single patch with fixed input size. The output is the classification result, whether the analyzed patch contains handwriting. During test time, patches are created by splitting the scanned page into patches of the same size, with the least possible overlap. We decide whether a scanned page contains handwriting by examining the classification result of each patch. If any patch is classified to contain handwriting, we decide that the entire page contains handwriting. This approach opens up the possibility of a higher number of false positives if the classifier is noisy. We think that such problems could be mitigated by allowing a higher overlap between fragments and performing local voting involving the classification result of multiple patches in the neighborhood. We leave the exploration of such post-processing for

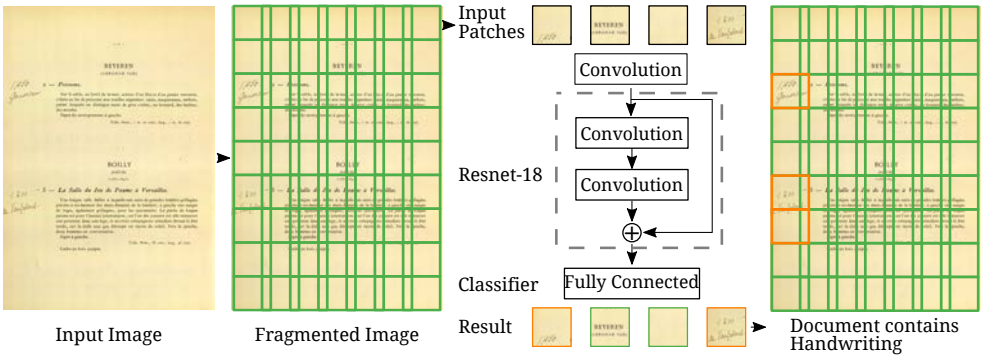


Figure 4.4: A depiction of our full handwriting determination system. We fragment the input image into several patches. Each patch is put into a ResNet-18 based binary classifier. The classifier categorizes each sample whether it contains handwriting or not. An orange border indicates patches containing handwriting, whereas a green border indicates patches without handwriting. Based on the classification result of all patches, we decide whether the document contains handwriting. We deem a document to contain handwriting if there is at least one patch classified as a patch containing handwriting. The division of the input image into several patches also allows getting the rough location of handwriting on each page.

future work, as our system already achieves promising performance, as we show in section 4.3.3. We provide a visual explanation and overview of our proposed analysis pipeline in figure 4.4

4.3.3 Experiments

Following the description of our proposed system, we now present the evaluation results on a real-world dataset. We gathered a real-world dataset from the archive of the art-historical archive of the WPI and manually labeled the data for evaluation. In the following, we introduce our experimental setup, present our achieved results and discuss them.

4.3.3.1 Experimental Setup

We implement our proposed system using the deep learning framework Chainer [225]. We set the input size of our deep neural network to 224×224 pixels and train the network only using synthetic data generated by our data synthesis tool. For training, we synthesized 1 million patches containing handwriting and another 1 million patches without handwriting, resulting in two million samples. We split the dataset in a train set with 1.98 million samples and a validation set with 20 000 samples. For optimization, we use Adam [132] as optimizer. We set the initial learning rate to 10^{-4}, train our

network for 200 epochs, with a batch size of 64, and shift the learning rate by a factor of 0.1 every 10 epochs. We perform all of our experiments on a machine equipped with an Nvidia RTX 2080Ti GPU.

Our manually labeled evaluation set consists of 1773 scanned document pages from the art-historical archive of the WPI. The scans in our evaluation dataset are very diverse. The dataset includes scans of book pages, auction catalogs, pages containing images, book covers, letters, and scans of modern documents with and without handwritten annotations. After annotation, we ended up with 1396 (roughly 78 %) pages containing handwriting and 377 (roughly 22 %) pages that do not contain handwriting. Before creating patches of size 224×224, we resize the largest side of the document scan to a size of 2000 pixels. We choose 2000 pixels as maximum side length to make sure that we can fit all patches into the memory of one of our GPUs at the same time. We note that sticking to such a constraint is not required, as it is also possible to examine each patch individually, using much less memory on the GPU. After we resize the input image, we create the patches in a sliding window fashion, as shown in figure 4.3. Each patch is used as input to the classification network. Based on the classification result of each patch, we decide whether a page contains handwriting, as described in section 4.3.2.2.

4.3.3.2 Experimental Results

To validate that our data synthesis approach is viable and that models trained with our synthetic data are applicable to real-world data, we perform three experiments on our test set. The experiments differ in the pre-processing method used for the input images. On the one hand, we evaluate how the model behaves when supplied with input binarized using two different binarization strategies. On the other hand, we evaluate the model's performance when using grayscale versions of the scanned pages as input. We assume that the model performs best when supplied with binarized images. We argue that our assumption should hold because the model has been trained on binarized images only. Hence, it should perform worse on grayscale input because the data distribution is different. For binarization, we make use of Otsu thresholding [183] and also adaptive gaussian thresholding. In table 4.1 we report precision, recall, F1-Score, as well as, false negative and false positive rate for each experiment.

The experimental results verify that our data synthesis process is effective for the application to heterogeneous document data. However, we can also see that our model is more likely to produce false-positive than false-negative predictions. This behavior is not ideal, but we argue that a model that is more likely to produce a false-positive result is better than a model that might miss an important document. The impact of missing a document is much higher

Table 4.1: Experimental results of our three experiments on the dataset with data from the art-historical database of the WPI. Results in **bold** represent the best result. *FNR* and *FPR* dennote *false negative rate* and *false positive rate*, respectively.

Metric	Otsu Threshold	Adaptive Gaussian Threshold	Grayscale
Precision	0.96	0.96	**0.97**
Recall	0.97	0.97	**0.98**
F1-Score	0.97	0.97	**0.98**
FNR	0.03	**0.02**	**0.02**
FPR	0.14	0.13	**0.10**

than the impact of a document wrongly predicted to be positive because it is easier to dismiss a false positive than to find a false negative. However, as already stated above, we might be able to mitigate this by improving the post-processing and decision-making of our approach, which is an idea that should be explored in future versions of our proposed approach. Another interesting observation from our experimental results is that our model performs best on grayscale input images. The result contradicts our assumption that the model should work better on binarized images, as we trained only on binarized images. We argue that our model shows this behavior because the binarization of complex structures, such as images, introduces artifacts that we can not model perfectly with our synthetic data. Using grayscale images as input does not produce such isolated artifacts. Hence, the model is not confused when presented with such data.

Besides the quantitative results in table 4.1, we also provide qualitative results of our model on patches of the data from the WPI in figure 4.5. In this figure, we present patches that have correctly been classified as containing handwriting, correctly classified as not containing handwriting, and patches that have falsely been classified as containing handwriting. We provide 4 patches for each category. We can observe that our model can handle complicated backgrounds and various kinds of handwriting well, even if handwritten words are not completely visible in a patch. Furthermore, our model can correctly classify empty patches, although we never synthesize any fragments without any text or image properties. A further examination of the false-positive predictions reveals that our model cannot handle complex backgrounds without any handwriting, *e.g.*, in the upper-right image. The model might show this behavior because we did not correctly mimic organic structures of such complicated backgrounds in our synthetic training data. In

Figure 4.5: Qualitative results of our model on several real-world fragments taken from documents of the archive of the WPI. We show *true positive*, *false positive*, and *true negative* results. We can see that our model successfully operates on grayscale images, although it was only trained on binarized images. However, our model struggles with complex structure and artifacts that exhibit properties of handwriting, *i.e.*, curved lines, as shown in the *false positive* column.

the future, we think that we might be able to improve the results by, on the one hand, adding more such structures to the data synthesis tool. On the other hand, we think that we can use the samples synthesized by the generative model we use in section 4.5 to synthesize a training set for our handwriting determination model.

Besides the experiments to validate the applicability of our proposed pipeline, we also used the proposed model for handwriting determination to balance the dataset for the training of our generative model (see section 4.5.5). Thanks to the balancing of the training set, we were able to train a generative model that produces more diverse samples containing both printed and handwritten text than the model trained on the unbalanced dataset. If we were to use the data synthesized by our generative model whose training data we balanced using the model introduced in this section, we would literally pull ourselves up by our bootstraps to train a better model for handwriting determination. We think this is an exciting observation that shows the possibilities and hints at possible problems using synthetic data in deep learning. One of the most severe problems could be that the models adjust only to the data distribution and domain of the synthetic data and not to the domain of the real data, which would render all of our efforts useless. On the other side, the possibilities are endless and would enable us to apply deep learning to more and more problems that were out of reach before.

4.3.4 Summary

In this section, we introduced a method for automatically determining whether a page's scan contains handwriting. Our method is designed to be a pre-processing step for document analysis pipelines in the domain of historical and archival data. Determining whether a page contains handwriting is especially helpful when analyzing diverse documents in content and might not contain handwriting on each page of the document. Thus, determining whether a page contains handwriting is a meaningful method to determine which further analysis steps for a given document need to be performed. A huge problem when dealing with historical or archival data is the availability of annotated training data. We overcome this problem by introducing a data synthesis method that can be used to train a deep neural network for handwriting determination successfully. In our experiments, we validate that our proposed data synthesis algorithm is viable. We show that a model trained on our synthetic data can successfully determine whether the scan of a page from the archive of the WPI contains handwriting. Thanks to our patch-based approach for handwriting determination, our model can also provide the rough handwriting location on a page. Furthermore, we found that our proposed model can also balance the training data for the training of other models further up in a document analysis pipeline.

Following the determination of whether a page contains handwriting, further analysis steps need to be employed. In the following sections, we present two further approaches that we developed. Both approaches are also based on the use of synthetic data.

4.4 Handwriting Classification

Following our first contribution to a historical document analysis pipeline, we now tackle another, later, part of the pipeline. In this section, we introduce a novel pre-processing step before the actual handwriting recognition step. We propose classifying a handwritten word based on its visual appearance into categories such as date, number, alpha-numeric, *etc*. The work presented in this section has also been discussed in [27] and was produced in collaboration with Hendrik Rätz, see chapter 6 for more information about the collaboration.

4.4.1 Introduction

A pipeline for OCR consists of multiple pre-processing steps before the actual text recognition stage is performed (see section 2.1.1). These pre-processing steps are necessary because the actual automatic recognition of textual content

is a complex problem. To further simplify text recognition, especially for handwriting, we propose another pre-processing step in this section. Our novel pre-processing step is located right before the recognition step. Based on an image containing a cropped word, we extract semantic information about the word without reading it letter by letter. Instead, we train a deep neural network to classify the content of the word image. Possible categories include, but are not limited to: dates, numbers, or alphanumeric strings. Classifying words based on their visual appearance has several advantages. First, it enables us to select the best recognition algorithm for each word instance, possibly increasing recognition accuracies. Second, the classification of handwriting allows us to provide researchers who search for specific semantic entities, *e.g.*, dates, with results, although we did not read the text in its entirety. Having such an option can be of high benefit because we can also provide semantic information found in documents written in languages where not much training data is available, but the visual structure of particular entities is similar. Thus, documents already analyzed with handwriting classification can help researchers to identify and filter documents and pages quickly by specifying the type of information they are looking for.

In this section, we introduce and evaluate several possible approaches for handwriting classification (see section 4.4.3). On the one hand, we propose to train a softmax classifier with a fixed number of classes. On the other hand, we propose to learn a more flexible model that uses metric learning to embed images of handwritten words. The objective of the model trained using metric learning is to embed images that have a similar structure close to each other while positioning images with a differing structure far from each other. Here, we also use methods for the synthesis of training data because already available datasets, such as the IAM database [167] do not contain many instances of handwritten numbers and no dates at all (see section 4.4.2). In our experiments, we show that our proposed models are well suited for handwriting classification. We also verify that our approach based on metric learning is flexible and even allows to distinguish classes it has never seen during training (see section 4.4.4). The contributions we present in this section can be summarized as follows: (1) We introduce methods for the successful synthesis of handwritten dates, numbers, *etc*. (2) We perform an in-depth analysis of the capabilities and drawbacks of several deep learning architectures for handwriting classification. Code and models are freely available for the community[7] to facilitate further experimentation.

[7] https://github.com/hendraet/handwriting-classification (last accessed August 31, 2021).

4.4.2 Synthesizing Realistic Handwriting

Existing handwriting databases, *e.g.*, the IAM Handwriting Database (IAMDB) [167], IAM Historical Handwriting Database (IAM-HistDB) [77, 78] or the RIMES dataset [14], contain a large amount of handwritten text but hardly any numbers, dates, or alphanumeric strings. The largest of the IAM databases (IAMDB), for instance, contains 115 320 word images but only 454 of those word images contain numbers. For the classification of handwritten word images, we need annotated datasets that also contain a fair amount of numbers for the training of our models. Besides numbers, we might also need words following a specific structure if we want to be able to, *e.g.*, categorize alphanumeric identifiers or price tags. Datasets that are entirely synthetic exist, *e.g.*, the IIIT-HWS dataset [135]. However, even these datasets do not contain many samples that are not words. Thus, we resort to the synthesis of training data for our use case.

To overcome the problem of missing handwritten numbers and dates, we use two data synthesis methods. First, we use the stroke information taken from the IAM Online Handwriting Database (IAMONDB) [154] to synthesize the first naive samples containing numbers that we use to balance the training data for the second synthesis method. Second, we adapt the *GANWriting* model proposed by Kang *et al*. [121] to synthesize a large number of realistic handwriting samples.

4.4.2.1 Synthesizing Crude Data using Recorded Strokes

As already stated above, we do not have access to a sufficiently large amount of handwritten numbers and dates. To overcome this problem, we make use of online handwriting data. Online handwriting data contains the information on how the pen moved while a test subject wrote words. Such data is captured using specific hardware for the capture of pen movements during writing. The IAMONDB [154] is a dataset that contains online handwriting information of 86 272 words obtained from 221 writers. Out of these samples, we identify all words containing numbers and manually extract the strokes necessary to create each digit. We also identify and extract stroke information of characters such as dots and dashes to synthesize dates.

Once we gather all stroke information, we concatenate the stroke information to new strings that resemble numbers and dates and render them on a white canvas. Since the stroke information does not include any hint about stroke thickness and color, we randomly select a stroke thickness and color for each synthesized sample to increase the variety of synthesized samples. We note that we do not mix stroke information of different writers to keep the writing style consistent, as we need it to be consistent when applying the

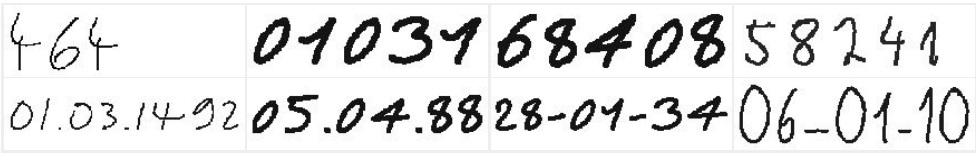


Figure 4.6: Samples synthesized using the stroke information from several writers contained in the IAMONDB. We depict samples that display numbers and dates.

GANWriting model by Kang *et al*. All in all, we can use stroke information from 111 writers of the IAMONDB, leaving us with a variety of writing styles. In figure 4.6, we show examples of the samples synthesized using the online handwriting synthesis approach.

4.4.2.2 Synthesizing Handwriting using the GANWriting Model

Following the synthesis of handwritten digits and dates, we apply another synthesis model to synthesize a broader range of handwriting data. To this end, we adapt the GANWriting model by Kang *et al*. [121], which is a generative model based on a conditional GAN architecture. The GANWriting model can produce word images containing realistic readable handwriting while being conditioned on the content and the handwriting style. A vital characteristic of the model is that it can synthesize words or character combinations that have not been part of the training set, which drastically increases the synthesis possibilities for our training data, compared to the synthesis using only online handwriting data.

In the following, we introduce the GANWriting model in more detail. Compared to a vanilla GAN (see section 2.2.6), the GANWriting model is more sophisticated. Besides the generator and discriminator, the GANWriting model also contains a recognition model and a writer classifier model. The two extra models are used to guide the generator to synthesize not only images that resemble handwriting (the task of the discriminator) but also images that contain readable handwriting (the task of the recognition model) and text that is written in the correct style (writer classifier). We provide an overview of the model in figure 4.7.

Generator The generator of the GANWriting model takes two inputs. First, multiple images depicting handwritten words of one author are passed to the style encoder of the generator. The task of the style encoder is to extract the stylistic features that it needs to synthesize handwriting in the same style. The sylistic features are extracted using a modified VGG-19 [213] that uses instance normalization [227] instead of BatchNorm.

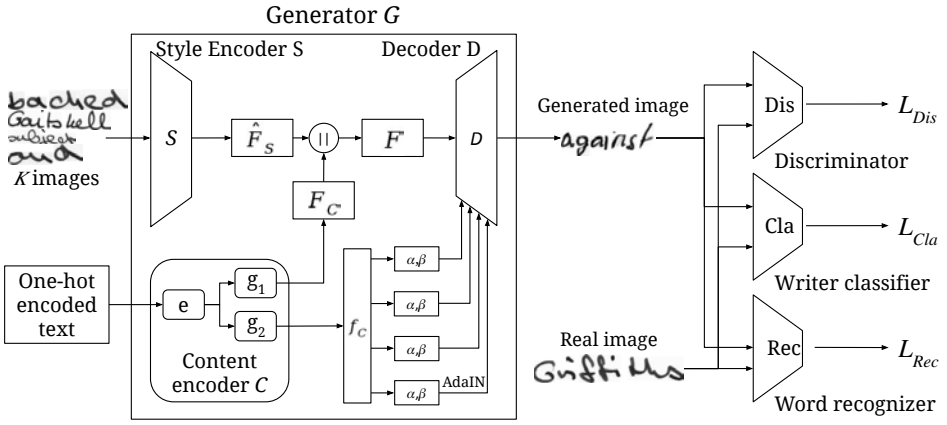


Figure 4.7: Structural overview of the GANWriting model. The model consists of a generator that receives the text that is to be generated and a set of K images that define the desired rendering style. The quality and readability of the synthesized images are assessed by three models: a discriminator, a writer classifier, and a word recognizer.

The second input to the GANWriting generator is a vector with a one-hot encoded representation of the input characters the generator shall render in the resulting output image. First, each character is embedded using a character embedding layer that converts the one-hot representation to a vector of 64 dimensions. Following the character embedding, two encoders are used.

First, the character-wise encoder encodes the embedded characters and brings them to the same dimensionality as stylistic features. The output of the character-wise encoder is then concatenated with the stylistic features. The resulting character and style vector is subsequently used to steer the synthesis of the word in the correct style with the correct input. The original model formulation of Kang *et al*. only allowed the synthesis of words with a word length of up to 7 characters. Since date strings are longer than seven characters, we adapted the padding strategy of the model, effectively allowing the synthesis of words up to a maximum of 25 characters. Originally padding in the GANWriting model works by repeating each character until the string reaches a length of 27 including a start and end of sequence token. The maximum length is set to 27 because the output of the character-wise encoder is concatenated with the output of the style encoder. Requiring the same amount of repeats for each character limits the maximum length of words to be generated by the model severely. Thus, we introduced a new empty string token ϵ. The empty string token is added to the end of the string. It does not include any new information and is solely used to fit the size of the character-wise encoder output to the output of the style encoder. For the word "ExampleTwo" and a (hypo-

thetical) maximum string length of 11, the process would be the following:

1. Pad string: ExampleTwoϵ
2. Start and end token: <S>ExampleTwo<E>ϵ
3. Repeat sequence: <S><S>EExxaammpplleeTTwwoo<E><E>$\epsilon\epsilon$
4. Add extra padding: <S><S>EExxaammpplleeTTwwoo<E><E>$\epsilon\epsilon\epsilon$.

The second encoder that follows the embedding layer is the global string encoder. The global string encoder produces a more general representation of the character string. The character string is represented as a set of multiple mean and variance vectors that are used to guide the Adaptive Instance Normalization (AdaIN) layers of the decoder.

The decoder is the actual generation part of the generator. The architecture of the decoder is based on the MUNIT architecture by Huang *et al*. [105]. The decoder uses residual blocks and AdaIN [104]. The input to the decoder is the feature map obtained by concatenating the style features with the output of the character-wise encoder. The output of the global string encoder is used to drive the AdaIN layers. In the end, the decoder produces an image of size 216×64 (width $\times$ height).

Discriminators The GANWriting model consists of multiple discriminators, a discriminator, a word recognizer, and a writer classifier.

The first model is a discriminator that tries to determine whether the given input image is real or if the generator generated it. The architecture of the discriminator is based on the ResNet architecture, activated with leaky rectified linear unit (ReLU) [163]. The discriminator is trained using binary cross entropy (BCE). Thus, the loss $\mathcal{L}_{\text{Dis}}$ of the discriminator is calculated as:

$$\mathcal{L}_{\text{Dis}} = \text{BCE} = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y})). \tag{4.1}$$

With y denoting the real label, either 0 (fake) or 1 (real) and $\hat{y}$ denoting the output of the discriminator.

The text recognizer is a text recognition network for the recognition of handwriting. It is a sequence to sequence model proposed by Kang *et al*. [122] that consists of a VGG-19 and bidirectional gated recurrent units (GRUs) to encode image features. The output of the text recognizer is generated by an attention-guided bi-directional GRU decoder. Following standard practice, for text recognition, the model is trained using softmax cross-entropy loss for each character, which we denote as $\mathcal{L}_{\text{Rec}}$.

The writer classifier is a simple residual network that predicts the author the generated image was conditioned on. This model is also trained using softmax cross-entropy, which we denote as $\mathcal{L}_{\text{Cla}}$.

Training of the Model The model is trained following standard training practices for GANs (see section 2.2.6). The only difference in the GANWriting

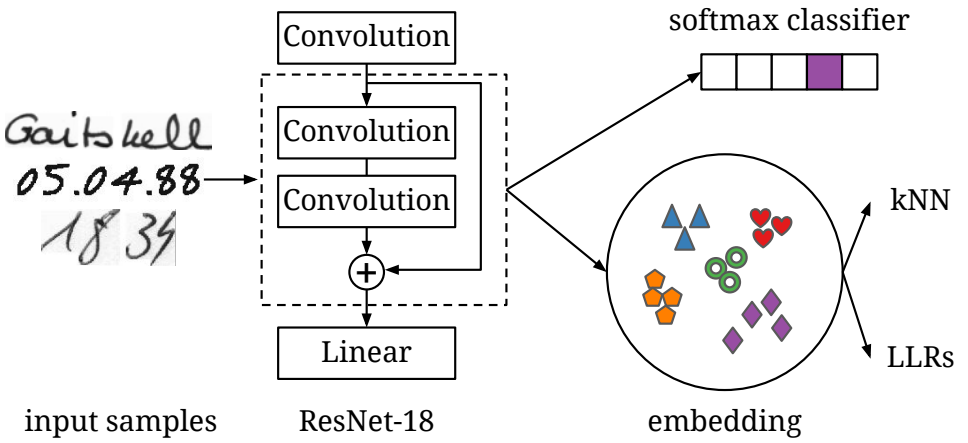


Figure 4.8: Our proposed models for handwriting classification consist of the same neural architecture for feature extraction. However, they handle extracted features differently. Our model based on a softmax classifier directly applies the softmax function on top of the classification layer. Our models based on embeddings learn these embeddings using triplet loss and reach their classification decision using one of two classification algorithms, k-nearest neighbors (kNN) or log-likelihood ratios (LLRs).

model is that the loss of the discriminator update consists of multiple loss terms:

$$\mathcal{L} = \mathcal{L}_{\text{Dis}} + \mathcal{L}_{\text{Rec}} + \mathcal{L}_{\text{Cla}}. \tag{4.2}$$

The loss term $\mathcal{L}$ is only used to update the generator. During the update of the discriminator, each model is updated individually.

4.4.3 Classification of Handwritten Words

In the following, we introduce our general approach and the two classification approaches we propose for handwriting classification. On the one hand, we propose the usage of a simple softmax-based classifier. On the other hand, we propose to use a distance-based embedding model. Here, classification is achieved by measuring the Euclidean distance of embedded samples and deciding on a class based on several methods. The primary motivation behind introducing the distance-based approach is its increased flexibility regarding the classification of samples from classes unseen during training. In figure 4.8, we provide a visual explanation of our proposed approach including inputs and all classification methods.

4.4.3.1 Softmax Classifier

Our first model is a deep neural network with a feature extractor based on the ResNet-18 [94] architecture. On top of the feature extractor, we employ a softmax classifier with a fixed number of classes. We train the model using softmax cross-entropy as our loss function. Classifiers trained using softmax cross entropy are very strong classifiers, *e.g.*, image classification on ImageNet surpasses human performance using softmax based classification [94, 221].

However, softmax models are pretty inflexible. It is, for instance, not possible to add a new class to the classifier without retraining the whole network. In an actual use case, a researcher might want to find documents containing a specific kind of handwriting without the need for a long-running model adjustment and data synthesis. We propose another approach that could be used in such a case and counter the inflexibility of the softmax approach.

4.4.3.2 Distance Based Model

Besides the inflexible softmax classification model, we propose a distance-based model. Our distance-based model consists of a convolutional feature extractor and an embedding layer. As a convolutional feature extractor, we build on a ResNet-18 model. Following the feature extractor, we add a fully connected layer that transforms the feature map to a vector of 512 dimensions. The input to our feature extractor is an image of a cropped handwritten word. We train our model to produce embeddings close to each other (in terms of euclidean distance) if the input images depict the same type of content, *e.g.*, words, dates, numbers, *etc*. To learn the embedding, we train our model with the triplet loss function [203]. Once we obtain a trained model, we categorize images based on two different classification schemes. On the one hand, we experiment with categorization based on k-means [157] for clustering and kNN [10] for classification. On the other hand, we experiment with a distance-based thresholding approach utilizing LLRS [49]. In the following, we first introduce triplet loss, followed by an introduction of our classification approaches.

Triplet Loss Schroff *et al*. originally developed the triplet loss function for the training of face recognition models [203]. The idea of triplet loss is an advancement of the well-known siamese network training scheme [48] where two networks share weights, and the similarity between two inputs is measured and used for classification or verification decisions. Instead of two inputs to measure similarity, a triplet network uses three inputs, as the name already indicates. The three inputs to a triplet network are denoted as *anchor*, *positive*, and *negative*. Anchor and positive belong to the same class, whereas negative belongs to a different class. A set of three deep neural networks that

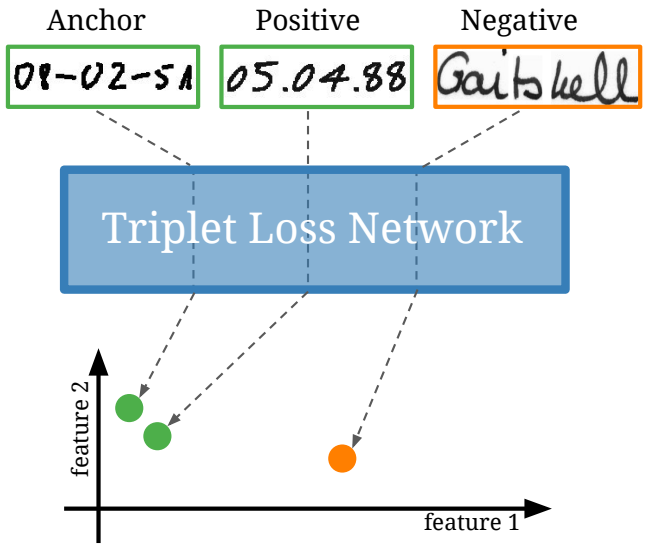


Figure 4.9: Triplet loss encourages a deep neural network to place samples of the same class to each other. Three samples are used to achieve a close placement. Two samples of the same class (*anchor* and *positive*) and one sample of another class (*negative*).

share weights transform the input images to their corresponding embeddings. The loss $\mathcal{L}_{\text{triplet}}$ is then computed using the following formula:

$$\mathcal{L}_{\text{triplet}} = \max(0, d(a,p) - d(a,n) + \alpha). \tag{4.3}$$

With a, p, n denoting anchor, positive and negative sample, respectively. The hyperparameter α represents the margin, *i.e.*, the minimum distance that is enforced between positive and negative samples. The function d denotes the euclidean distance between two embeddings of dimensionality n:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}. \tag{4.4}$$

The triplet network tries to minimize the distance of the embeddings of the anchor and positive sample while simultaneously maximizing the distance of the embeddings of the anchor and the negative sample. We could also say that the triplet loss trains the network to embed images representing the same class close to each other. We provide a visualization of the idea of triplet loss in figure 4.9.

We chose triplet loss as our loss function because related work [192] showed that a model that learned a clustering based on triplet loss could handle samples from previously unseen classes allowing us to gain the flexibility we wish for our network.

Classification using k-means and kNN To determine the class of a given word image, we need to categorize the extracted embedding. One possible approach is to automatically find clusters in the embeddings of the test data using k-means, where k denotes the number of classes that are to be distinguished. The result of k-means are k cluster centroids. Once we extracted the clusters' centroids, we use our trained model to embed a set of support samples, *i.e.*, samples where the class is known. We then use kNN to label the cluster centroids. In the final step, we assign labels to each sample of our test set by assigning it the class of nearest centroid. In our experiments, we denote this approach as the "naive" approach.

Classification using log-likelihood Ratios Another approach for classifying embeddings of word images is to classify embeddings based on a distance threshold, where we do not need to find centroids of embeddings. Related work proposes to use several different thresholding methods. Schroff *et al*. [203] propose to use a hard and fixed threshold. Such a threshold needs to be tuned manually to get the best classification result. In another work, Rantzsch *et al*. [192] propose to use a more flexible distance-based thresholding for their work on signature verification using triplet learning.

Rantzsch *et al*. propose to use the concept of log-likelihood ratios (LLRS) [49]. LLRS produce soft decisions where it is not necessary to tune a distance threshold for each class manually. The soft decisions produced by LLRS provide a relative score that reflects the confidence of a classification decision. The ratio is formally defined as:

$$\text{llr}(d) = \log \frac{P(d|\text{target trial})}{P(d|\text{non-target trial})}. \tag{4.5}$$

The rationale behind the function is to determine it is how likely it is that the calculated distance d is observed for a sample of class A (denoted as *target trial*) or class B (denoted as *non-target trial*). To enable such computation at test time, we need a support set of which each sample is embedded. Based on the embeddings of all samples, we can calculate the distance distributions that are necessary for the calculation of the LLR.

The formulation of LLRS that we present here can not directly be used for multi-class classification. To use LLRS for multi-class classification, we evaluate the embedding of a handwritten word image in a one-vs-all approach. In this setting, the LLR for each class versus all other known classes is computed. Since an LLR represents confidence, we assign the embedding to the class with the highest confidence value.

4.4.4 Experiments

To validate our assumptions and show the feasibility of our proposed approach, we now perform a series of experiments on multiple datasets. In our experiments, we show that our proposed models can be used for application on the problem of handwriting classification. We further validate our hypothesis that our model based on distance embeddings is flexible and can be used with classes not seen during training. We begin by introducing our experimental setup, followed by all datasets we use in our experiments. Last, we present our experimental results on all datasets.

4.4.4.1 Experimental Setup

We base our implementation of the GANWriting model on the PyTorch [184] implementation[8] by the original author of the GANWriting paper [121]. We train the GANWriting model in two rounds (we restart the training after the first round with the fully trained generator but randomly initialized discriminators) of 3000 and 6000 epochs, respectively. For training, we use a batch size of 16, a GPU with a total of 12 GB of RAM, and we use Adam [132] as optimizer, setting the learning rates to 10^{-4} for discriminator and generator and to 10^{-5} for writer classifier and text recognizer.

We implement our classification models using the deep learning framework Chainer [225]. For training, we also use a GPU with a total of 12 GB of RAM, Adam with a learning rate of 10^{-4}, and a batch size of 128. We train our classification models for 20 epochs.

We evaluate our models using accuracy, recall, precision and F1-Score.

4.4.4.2 Datasets

We perform our experiments on multiple datasets. On the one hand, we use two different synthetic datasets for training and evaluation of the feasibility of our approaches. On the other hand, we gather a dataset containing real samples from the archive of the WPI for evaluation purposes only. We provide some samples from each dataset for reference in figure 4.10. In the following, we introduce each dataset in more detail.

Synthetic Datasets As already mentioned in section 4.4.2, getting access to a large-scale annotated training dataset with various types of handwritten content is complex and, in some cases, even impossible. Hence, we use the data synthesis methods we introduced in section 4.4.2 to synthesize a sufficient amount of training data for the training of our proposed models. All in all, we synthesize and assemble two datasets. The first dataset consists of samples

[8] https://github.com/omni-us/research-GANwriting (last accessed August 31, 2021).

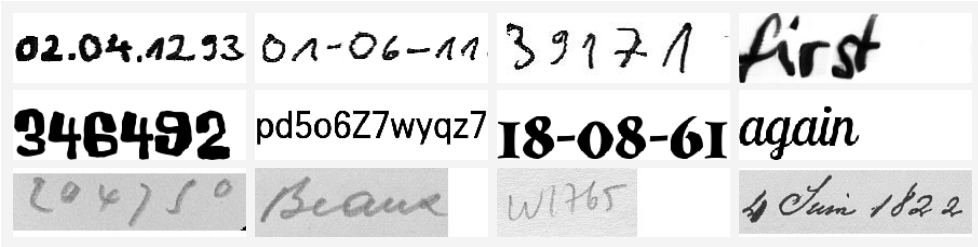


Figure 4.10: Depiction of samples from all datasets we use for our experiments on handwriting classification. In the first row, we show samples from the GANWriting dataset. The second row depicts some printed samples that are added to the samples of the GANWriting dataset to form the Five Classes Handwritten and Printed Text (5CHPT) dataset. The last row shows some samples from the WPI dataset, which solely consists of real images from archival documents and is used for evaluation only.

synthesized by our adaption of the GANWriting model (see section 4.4.2.2). The dataset consists of handwriting samples from three classes, *i.e.*, words, numbers, and dates, totaling in 7920 samples equally distributed over the three classes. We split the dataset into train and test set using a $9:1$ ratio. The purpose of this dataset is to quickly perform experiments to validate the applicability of our proposed models to the task of handwriting classification. For the remainder of this section, we refer to this dataset as the GANWriting dataset.

For the second dataset, we added two additional classes and also a different text modality. We add alphanumeric strings and five-digit numbers resembling zip codes as additional classes. We added alphanumeric strings because we want to investigate the capabilities of our proposed models when confronted with a mixture of digits and letters. Such mixtures are especially interesting for the use case of researchers if they are, for instance, searching for specific alphanumeric identifiers. On the other hand, we add zip codes because we want to investigate the counting capabilities of our models. Besides the two extra classes, we also add samples that are not handwritten but are instances of printed text. We also decided to add printed text because the usage of printed text allows us to scale the amount of available training data by a large margin and adds a great variety of visual appearances in our samples, making the task even more complicated for our models. Since the dataset consists of five classes containing handwritten and printed text, we refer to the dataset as the Five Classes Handwritten and Printed Text (5CHPT) dataset. In total, the dataset consists of 26 400 samples, equally distributed over the five classes.

Table 4.2: Evaluation results for our different handwriting classification models on the GANWriting dataset. Results in **bold font** indicate the best performance.

Experiment	Accuracy (%)	Precision (%)	Recall (%)	F1 Score
Naive	**99.62**	**99.62**	**99.62**	**0.9962**
LLR	99.24	99.25	99.24	0.9924
Softmax	**99.62**	**99.62**	**99.62**	**0.9962**

WPI Dataset Besides our synthetic datasets, we also gathered a test dataset from the archive of the WPI. This dataset consists of 272 cropped word images of handwritten text, showing numbers (112 samples), words (108 samples), alphanumeric strings (31 samples), and dates (21 samples). These samples are very different from the samples we use for the training of our models, as can be seen in the last row of figure 4.10. As we will see in our experimental results, these differences lead to a degradation in the performance of our model. However, we are still confident that our model can be used on original WPI data if we can synthesize data close enough to the real data. For now, we could not do this because not enough samples from a similar data distribution of the WPI samples are available. In section 4.5 we provide an approach that could be helpful to gather other samples that we can use to improve the results of our handwriting classification approaches later.

4.4.4.3 Results on the GANWriting dataset

In our first set of experiments, we assess whether our idea of handwriting classification and whether our proposed models are feasible. To test the feasibility, we train our models on the GANWriting dataset. The results, see table 4.2, show that all of our three approaches can correctly classify samples from the GANWriting dataset. To further understand the reasons of the performance of our distance based embedding methods *naive* and *llr*, we visualize the predicted clusters for each sample in figure 4.11. We obtain the cluster visualization by applying PCA [102] to the embeddings.

The visualization shows that our embedding model can produce three nearly perfectly distinct clusters. Hence, our classification methods on the distance-based embeddings produce near-perfect results. From the visualization, we can also see that the model seems to focus on the text structure in the word image because the cluster of dates is positioned far away from the other two clusters.

4.4.4.4 Results on the 5CHPT Dataset

Following our encouraging results on the GANWriting dataset, we experiment with the samples from the 5CHPT dataset. Here, we examine whether our

Figure 4.11: PCA of samples from the GANWriting dataset

Figure 4.12: PCA of samples from the 5CHPT dataset

Table 4.3: Evaluation results for our different handwriting classification models on the 5CHPT dataset. Results in **bold font** indicate the best performance.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score
Naive	74.32	76.31	74.32	0.7456
LLR	79.89	80.19	79.89	0.7999
Softmax	**90.00**	**90.62**	**90.00**	**0.9004**

Table 4.4: Detailed evaluation results for our different handwriting classification models on the 5CHPT dataset. Results in **bold font** indicate the best performance.

Class	Model	Precision (%)	Recall (%)	F1 Score
Alphanumeric	Naive	**90.19**	62.69	0.7397
	LLR	84.51	81.63	0.8304
	Softmax	85.19	**91.48**	**0.8822**
Date	Naive	97.12	95.64	0.9637
	LLR	99.01	94.70	0.9681
	Softmax	**99.80**	**96.02**	**0.9788**
Number	Naive	48.78	49.24	0.4901
	LLR	64.96	62.50	0.6371
	Softmax	**91.40**	**82.58**	**0.8677**
Zip Code	Naive	60.14	79.17	0.6836
	LLR	65.97	71.97	0.6884
	Softmax	**81.25**	**96.02**	**0.8802**
Word	Naive	85.33	84.85	0.8509
	LLR	86.51	**88.64**	0.8756
	Softmax	**95.47**	83.90	**0.8931**
Average	Naive	76.31	74.32	0.7456
	LLR	80.19	79.89	0.7999
	Softmax	**90.62**	**90.00**	**0.9004**

Table 4.5: Evaluation results for our different handwriting classification models that were trained on the 5CHPT dataset on samples of the WPI dataset. Results in **bold font** indicate the best performance.

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score
Naive	**58.33**	49.82	**58.33**	**0.5263**
LLR	25.76	32.96	25.76	0.2771
Softmax	37.50	**60.61**	37.50	0.4086

models can handle a wider variety of classes more similar to each other than the classes from the last experiment. We also explore how well we can find and extract specific classes of interest that are not much different from other classes, such as zip codes, a subclass of the number class. We present the results of our experiments on the 5CHPT dataset in table 4.3 and table 4.4. The results show that the dataset is far more challenging for our proposed models. We can also observe that the softmax model performs best. This observation is not surprising because the softmax model does not rely on distances for the setting of its classification barrier. However, the results show that our embedding-based models also work well in this case. We can also see that the naive classification approach is outperformed by the approach based on LLRs.

A visualization of the clusters found by our embedding model (see figure 4.12) shows that it is more difficult for the model to distinguish between zip codes and numbers, as the clusters are highly overlapping. We can also see that the structure of the embedding space seems to be semantically structured because the embeddings of the alphanumeric are between numbers and words. All in all, we can conclude that our distance-based approach works best if the visual structure of the words to classify is different, as it is with dates and words or numbers. However, the results still show that the usage of a distance-based model is feasible.

If we examine table 4.4, which shows more detailed results for each class, we can confirm the observations made by examining the clustering evaluation. We can see that the models have no problems with distinguishing the date class from other classes. The most problematic classes for all classification approaches are the zip code and number classes, as we would expect.

4.4.4.5 Results on the WPI Dataset

Following our experiments on the synthetic datasets, we evaluate the models trained on the 5CHPT dataset on the dataset that contains real images from the archive of the WPI. The results (see table 4.5) show a severe degradation in performance if we directly apply a model trained on the 5CHPT dataset

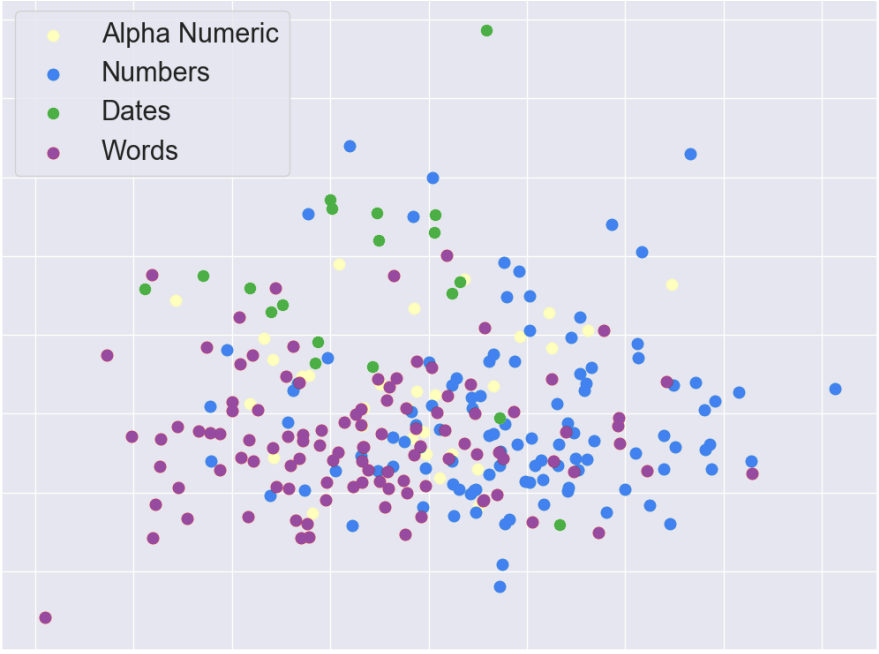


Figure 4.13: PCA of samples from the WPI dataset

on the samples of the WPI. An interesting first observation is that our naive model seems to outperform all other methods, which is an interesting result.

On closer examination of the classification decisions of each model, we are able to determine that the naive classifier performes better only on paper. To evaluate the classification decisions, we again plot a PCA of the embeddings in figure 4.13, provide classification results for each individual class in table 4.6, and also provide a confusion matrix for each classifier in figure 4.14. The reason for the excellent performance of the naive classifier becomes apparent quickly, as we can see that the naive classifier is not able to identify alphanumeric and date strings at all, as the F1-Score is 0 for both classes! The visualization of the clusters shows the reason for this result. The clustering models are not able to extract distinct clusters for each class. Since no clear clusters are visible, we argue that the k-Means algorithm can only find two cluster centroids, one for words and one for numbers. This leads to superior accuracy because the dataset is imbalanced, and the two classes that the naive classifier can identify contains the majority of the samples. In this setting, the LLR classifier shows its strengths because it is still able to identify samples from each class correctly. It also shows the most balanced misclassifications, mainly due to the poor clustering performance of the underlying embedding model. Finally, the softmax classifier incorrectly identifies many samples as alphanumeric strings (as does the LLR classifier), which is technically correct because these are a superclass of words, numbers, and dates.

	Alphanumeric	Date	Number	Word
Alphanumeric	0	0	5	25
Date	0	0	1	18
Number	0	0	67	40
Word	0	0	21	87

(a) naive classifier

	Alphanumeric	Date	Number	Word
Alphanumeric	10	5	6	9
Date	8	3	7	1
Number	40	2	28	37
Word	45	2	34	27

(b) LLR classifier

	Alphanumeric	Date	Number	Zip Code	Word
Alphanumeric	25	1	1	0	3
Date	15	4	0	0	0
Number	45	1	31	7	23
Zip Code	0	0	0	0	0
Word	54	0	11	4	39

(c) softmax classifier

Figure 4.14: Confusion matrices for our different classifiers on the WPI dataset. Rows refer to the actual class, whereas columns refer to the predicted class. The confusion matrix of the softmax model also includes the zip code class because it was trained on the 5CHPT dataset and is therefore restricted to classifying exactly five classes.

Table 4.6: Detailed evaluation results for our different handwriting classification models on the WPI dataset. Results in **bold font** indicate the best performance.

Class	Model	Precision (%)	Recall (%)	F1 Score
Alphanumeric	Naive	0.00	0.00	0.0000
	LLR	7.92	26.67	0.1221
	Softmax	**17.86**	**83.33**	**0.2941**
Date	Naive	0.00	0.00	0.0000
	LLR	20.00	5.26	0.0833
	Softmax	**66.67**	**21.05**	**0.3200**
Number	Naive	62.22	**78.50**	**0.6942**
	LLR	42.47	28.97	0.3444
	Softmax	**72.09**	28.97	0.4133
Word	Naive	55.04	**65.74**	**0.5992**
	LLR	41.18	32.41	0.3627
	Softmax	**60.00**	36.11	0.4509
Average	Naive	49.82	**58.33**	**0.5263**
	LLR	32.96	25.76	0.2771
	Softmax	**60.61**	37.50	0.4086

Table 4.7: Evaluation results for our proposed models when adding a new and unseen class to the classification task. The first row shows the results of our models on the training set with only two classes, while the second row shows the classification result of our models when adding a third, unseen class to the classification task.

Experiment	Model	Accuracy (%)	Precision (%)	Recall (%)	F1 Score
Two Class	Naive	**98.48**	**98.49**	**98.48**	**0.9848**
	LLR	93.75	94.32	93.75	0.9373
Three Class	Naive	65.66	49.33	65.66	0.5472
	LLR	**79.63**	**76.37**	**75.63**	**0.7520**

All in all, the results show that the softmax model can generalize best on unseen data from a different data distribution. Our distance embedding approaches fail in this scenario because the embedding model cannot embed the new images so that distinct clusters could reliably be identified and classified. Although the LLR model shows the worst performance in our evaluation metrics, we conclude that it performs better than the naive approach that entirely misses two classes.

The LLR and naive approach might perform better under different circumstances. The most crucial aspect is the availability of a suitable embedding. Further, the different data distributions of training data and evaluation data hinder all approaches from reaching good evaluation results. One example is the right-most sample of the WPI dataset that we display in the last row of figure 4.10. The sample shows a date. Instead of an entirely numeric date, it contains a month word, totally different from all samples we provided during training. We are confident that we can improve the results if we incorporate such images into our training dataset. Further, we wish to mention that the distance-embedding classification approaches rely on support samples of which the class is known. In our experiments on the WPI dataset, we only used samples from the 5CHPT dataset as support because here we have access to a training split that may be used for this purpose. We argue that our results could be improved if we had access to more labeled data of the WPI, which we try to achieve with the work presented in section 4.5 and further future work.

4.4.4.6 Classification of an Additional Unseen Class

In section 4.4.3.2 we argued that our embedding models are superior to the softmax model because they allow flexible adjustment of the classes to

distinguish. A model based on softmax classification cannot do so without retraining because the classification layer needs to be retrained for each new class individually. A model based on embeddings can distinguish newly introduced classes if the feature extractor can extract meaningful features for the new class. In this case, only a tiny support set of annotated samples is necessary to adjust the classifier to the new class. To verify this, we performed an additional experiment.

In table 4.7, we provide the results of an experiment, where we first trained our classifiers on a subset of the GANWriting dataset that included the *date* and *number* class. After the training is finished, we add the *word* class by embedding some support samples and adjusting the classifiers. The results in table 4.7 show that our classifiers are now indeed able to distinguish between all three classes. The scores are not as good as the scores when directly trained on the full dataset. However, the results show that our assumption holds. We argue that using a robust embedding model handwriting classification could be a great addition to a researcher's workflow.

4.4.5 Summary and Discussion

In this section, we presented a novel pre-processing step that is to be included in a handwriting analysis pipeline. Our novel pre-processing step is located directly before the actual recognition of text. Our proposed step classifies the content of handwritten words. Thus, we can contribute to handwriting analysis in multiple ways. On the one hand, such a classification enables us to select the best recognition model for a particular class of content. On the other hand, we can use the classification result to provide researchers with hints about the content of documents without the need to read the entire document and perform subsequent analysis steps, such as NER.

To this end, we proposed a set of classification methods that we validated in extensive experiments. To train our models, we proposed to use methods of data synthesis. On the one hand, we proposed to use a softmax-based classifier. On the other hand, we proposed using an embedding model that embeds samples of the same class close to each other while embedding samples of different classes far from each other. On top of the embedding model, we propose two different classification approaches. While the softmax classifier reaches the best results on all experiments, we argue that the usage of the softmax classifier should only be considered in specific usage scenarios. One of these scenarios would be handwriting classification in a given corpus, primarily if this classification should be performed to provide samples to specialized recognition models. Our classification approaches on top of the embedding model do not reach the same performance level as the softmax

classifier. However, we have shown that these models are very flexible and can be adjusted to work with new classes without retraining the entire model. Only some support samples are necessary to adjust the models to a new class. The usage scenario for such a classification model is an exploratory search of a corpus where a researcher wishes to find all handwriting occurrences with a specific structure. To offer such functionality to researchers, further research and data synthesis are necessary.

4.5 Segmentation of Printed and Handwritten Text

In this section, we introduce our most advanced approach for the synthesis of training data. In all earlier sections and chapters, we focused on synthesizing training data and possible applications of such synthetic data. While we concentrated on handcrafted algorithms for data synthesis in section 3.5 and section 4.3, in section 4.4 we made use of a combination of manual synthesis algorithms and neural networks. In this section, we propose a novel approach for the synthesis of training data by utilizing the inner knowledge of a GAN for the synthesis of data. The approach itself is still under active research. Hence, here we introduce the main idea of our approach and also provide the first results. Our approach focuses explicitly on data synthesis for pixel-wise semantic segmentation of historical documents. Our approach enables us to use raw image scans without any annotation to create a custom-fit pixel-wise semantic segmentation model for document images. We believe that our idea is applicable to document analysis and other domains where it is difficult and costly to obtain annotated training data, such as the analysis of medical images where experts are necessary for annotation, and privacy concerns make data collection and annotation difficult. We also believe that using generative approaches based on GANs for the synthesis of large-scale training data or even for the annotation of real images is one of the most promising ways to solve the problem of the availability of annotated training data. The findings presented in this section have also been discussed in [28].

4.5.1 Synthesis in Style

An essential step in processing a document in a document analysis pipeline is the binarization of the document. Binarization is essential to remove any background noise that might confuse the recognition algorithm. Especially open source OCR implementations such as Tesseract require a cleanly binarized image for their best performance. The archive of the WPI consists of a multitude of document types. In our analysis, we are primarily interested in the analysis of documents containing handwriting and printed text. We

propose using pixel-wise semantic segmentation to obtain a cleanly binarized document where we can decide which regions of a scanned page contain handwriting and which regions contain printed text.

Semantic segmentation as such is a task that is in focus of computer vision research since quite some time [93, 148, 197]. Semantic segmentation is also researched in the area of historical document analysis (see section 4.2.2). However, so far, pixel-wise segmentation of document images has not been focused on in many research endeavors. One of the reasons why pixel-wise semantic segmentation has not been in broader focus is the non-availability of annotated datasets and the high burden for creating annotated datasets, especially the costs associated with the manual annotation of large-scale datasets. To alleviate the costs of manual annotation and also allow the flexible adjustment of a model to a given data distribution, we propose to use synthetic data synthesized by a GAN that was trained to generate data unconditionally.

To this end, we propose a novel semi-automatic data synthesis pipeline that can be used to synthesize large-scale custom-fit training data for the analysis of data from a given archive. Our proposed pipeline consists of the following steps: First, we train a GAN directly on raw scanned images obtained from an archive. The task of the GAN is to synthesize images that are indistinguishable from real images for the discriminator of the model (see also section 2.2.6). The twist of our proposed pipeline is the fact that we do not only use the trained GAN for the synthesis of realistic-looking samples, we also use the GAN to synthesize a corresponding label image that can be used for the training of a semantic segmentation network. Specifically, we propose to use the generative capabilities of StyleGAN [129, 130] to train an unconditional generative model on a given set of real-data from an archive. We further make use of the observation that intermediate layers of a generative model, such as StyleGAN, might encode semantic information [65]. We use this information during the synthesis process of StyleGAN to create a color image and a semantic segmentation label image simultaneously. To define the semantic class of pixels in the intermediate layers of StyleGAN, we use a semi-automated approach. First, we use an unsupervised clustering approach and assign feature values to these clusters. Then, we require human intervention to classify the found clusters. To simplify the annotation process, we provide a simple annotation tool. Following the cluster classification, human intervention is also required to define steps of an algorithm to combine the information of several intermediate layers of the trained StyleGAN model. Although manual intervention is required, the time necessary to perform these analysis tasks can be performed in roughly 1 h per StyleGAN model. Following the manual adaption of the synthesis process, we can synthesize an arbitrary amount of

annotated training data, which can be used to train an off-the-shelf semantic segmentation model. We provide a more detailed description of our proposed approach in section 4.5.3. We note that concurrently to our work, another work by Zhang *et al*. [255] has been proposed that uses similar ideas to perform pixel-wise semantic segmentation on images of, *e.g*., cars or faces.

In section 4.5.5, we apply our proposed data synthesis model on the data of the archive of the WPI and show in qualitative results that our approach is viable and already shows promising results in its initial state.

In summary, the contributions we make in this section are as follows: (1) We propose a novel approach for the semi-automatic synthesis of custom-made training data for semantic segmentation of document images. (2) We evaluate our approach in depth and show that our idea is viable and could prove fruitful for future dataset-specific semantic analysis of documents. (3) We provide our code and models to the community[9] for further experimentation.

4.5.2 The StyleGAN Architecture

Before introducing our proposed approach in detail, we introduce the StyleGAN model, which is at the heart of our proposed approach. The StyleGAN architecture was first proposed by Karras *et al*. [129] and later improved by Karras *et al*. [130] to provide even better generation results. We will refer to the improved version of StyleGAN as StyleGAN 2. Generative models based on the StyleGAN architecture currently set the state of the art in unconditional high-resolution image generation. In figure 4.15, we show the structure of a GAN based on the StyleGAN architecture.

The StyleGAN architecture is based on the idea of progressive growing for GANs [127], which was also introduced by Karras *et al*. Overall, the architecture of StyleGAN is different from the architecture of other GANs. A StyleGAN generator consists of three main components. The synthesis network is the first component. The task of the synthesis network is the synthesis of the image. The architecture of the synthesis network is inspired by progressive growing GANs. However, in contrast to previous work, the latent vector, generally provided to the synthesis as the input layer to a feed-forward deep neural network, is not provided to the synthesis network. Instead, a constant feature vector is provided. During generation, the synthesis network starts with the synthesis of an image of size 4×4; subsequent synthesis blocks double the size of the generated image until a maximum size of 1024×1024 is reached.

The latent vector that controls the content of the synthesized image is provided to the second principal component, the mapping network. The

[9] https://github.com/Bartzi/synthesis-in-style (last accessed August 31, 2021).

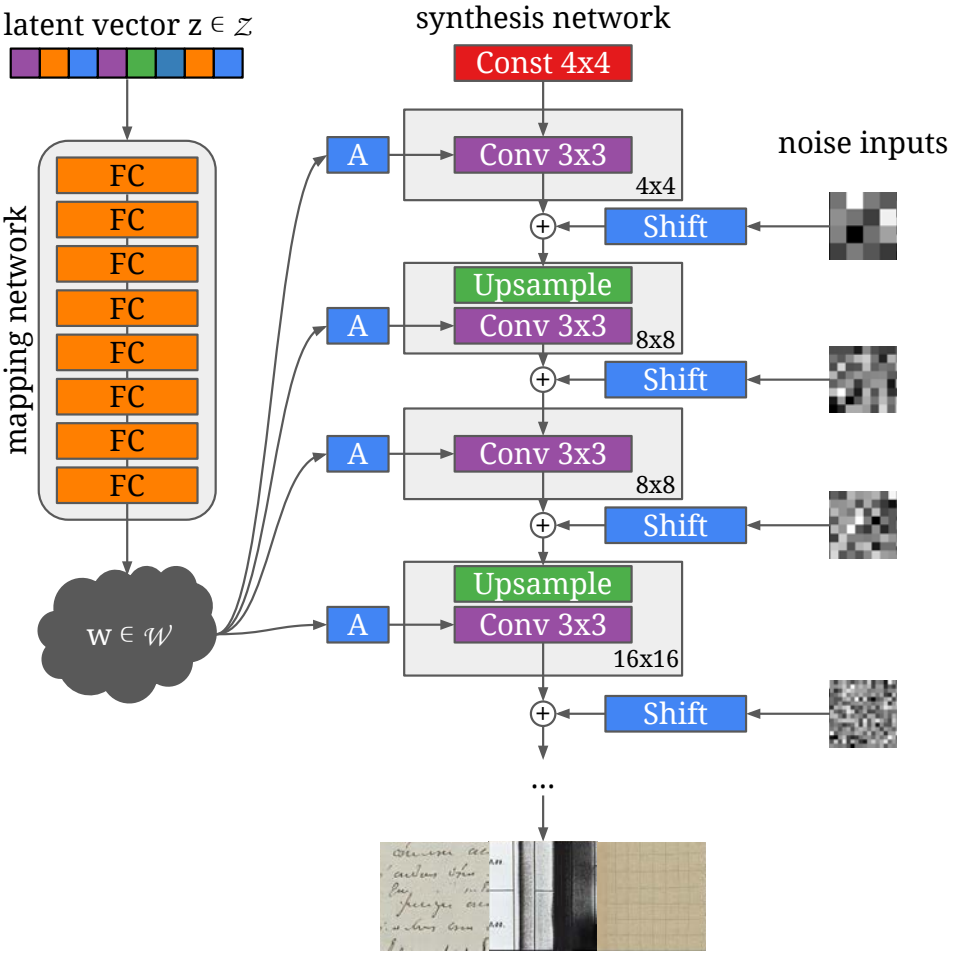


Figure 4.15: StyleGAN consists of three main components. First, the mapping network transforms a latent vector z to another latent vector w. The transformed latent code is then used to guide the generation of the image in the synthesis network. The synthesis network consists of multiple StyleGAN blocks whose size grows progressively. Stochastic noise inputs are the third component; they add stochastic variations, such as hair strains. Image reproduced and adapted with permission from [129].

mapping network is a deep multilayer perceptron (MLP) with leaky-ReLU non-linearities [163], which usually consists of 8 fully connected layers. The mapping network warps the representation of the latent vector $z \in \mathcal{Z}$ with $\mathcal{Z} \in \mathbb{R}^n$ to another latent vector $w \in \mathcal{W}$ with $\mathcal{W} \in \mathbb{R}^n$. The latent vector w is then transformed by learned affine transformations to the inputs of the synthesis network, which are used as styles to drive AdaIN [104] in StyleGAN 1 and as weight modulators in StyleGAN 2. The intuition behind the mapping network is to enable StyleGAN to provide a disentangled latent representation in the latent vector w. A disentangled representation allows the latent space to consist of linear subspaces where each of these subspaces controls one factor of variation, *e.g.*, gender, age, or skin color, to synthesize face images. The latent space $\mathcal{W}$ is disentangled because sampling from $\mathcal{W}$ does not depend on the training data, which might not have the required data density and distribution at each space. Sampling from $\mathcal{W}$ depends on a smooth piecewise linear learned mapping from the entangled input feature space $\mathcal{Z}$ to the intermediate feature space $\mathcal{W}$.

Stochastic noise inputs are the third principal component of StyleGAN. The stochastic noise is added to the feature maps in each synthesis block of the synthesis network. The noise inputs are single-channel images consisting of uncorrelated Gaussian noise. Before being fed to each layer of the synthesis network, the noise images are scaled using learned per-feature scaling factors. The effect of the stochastic noise is the addition of stochastic details, such as freckles, beard stubble, or locations of single hairs to the synthesized images.

4.5.3 A Novel Pipeline for the Synthesis of Segmentation Data

In contrast to recent methods, we focus on training a model entirely on synthetic data, which is directly based on the real data. In other words, we synthesize data that is custom-fit as much as possible to the data we want to analyze. Using such synthetic data allows us to use well-established supervised learning methods for the semantic segmentation of documents. Since we synthesize training data, whose data distribution is very close to the real data distribution, we can create models that can be directly used on the target data distribution, although no annotations of the real data are available. In this section, we introduce our proposed data synthesis pipeline in detail.

Our proposed data synthesis pipeline consists of the following four steps: First, we gather a dataset of documents where we want to perform pixel-wise semantic segmentation. We do not need any annotations to apply our method to this dataset; the raw scans suffice. Second, we train a StyleGAN model on the gathered dataset to perform unconditional data synthesis. We choose to use StyleGAN because, on the one hand, StyleGAN reaches state-of-the-art

performance for unconditional image generation, while on the other hand, StyleGAN inhibits many interesting properties that allow us to control the image generation [2, 3, 23, 65, 180, 188, 226]. Third, we analyze the trained StyleGAN model by making use of the observation that generative models and StyleGAN, in particular, encode semantic information about the class of each pixel in the intermediate layers of their synthesis networks [65]. Based on this observation, we create an algorithm that uses the information contained in intermediate layers to synthesize label images together with the corresponding color images. In the last step, we use our trained StyleGAN model and our devised segmentation algorithm to synthesize an arbitrary amount of training samples to form a large-scale, fully annotated dataset to train an off-the-shelf document segmentation network. In the following, we explain our pipeline using the example application of segmenting and classifying printed and handwritten text in documents images. We also provide an overview of each step in figure 4.16.

4.5.3.1 Training of StyleGAN

The first step after the gathering of a dataset is to train a StyleGAN model [129, 130] on patches of the original document images or the entire document images. We choose to train StyleGAN to generate only patches of the original images because of the same reasons we already used patches for our work presented in section 4.3. The main reason to use patches is that it is simpler to create a patch resembling real data than creating an entire document that faithfully represents the real data distribution. The usage of patches also allows us to analyze documents at a high resolution without compromises in the granularity of the results. Furthermore, patches allow the generator to concentrate on specific properties found in each document, such as areas with printed and handwritten text, images, text decorations, or scanning margins because a patch can not contain many of these properties at the same time. However, patches increase the number of computations necessary to analyze a given document scan. The usage of patches might also add inconsistencies at overlapping borders when assembling predictions.

4.5.3.2 Analysis of a Trained StyleGAN Model

After we obtain a trained StyleGAN model, we want to use it to synthesize an annotated training data set to train a pixel-wise semantic segmentation network. StyleGAN is designed to produce realistic RGB images with high quality and fidelity. Thus, at first glance, using a trained StyleGAN model to synthesize an annotated dataset for pixel-wise semantic segmentation seems not to be possible. However, it is possible to deduce the class information of pixels from the feature maps of the synthesis network. During the generation

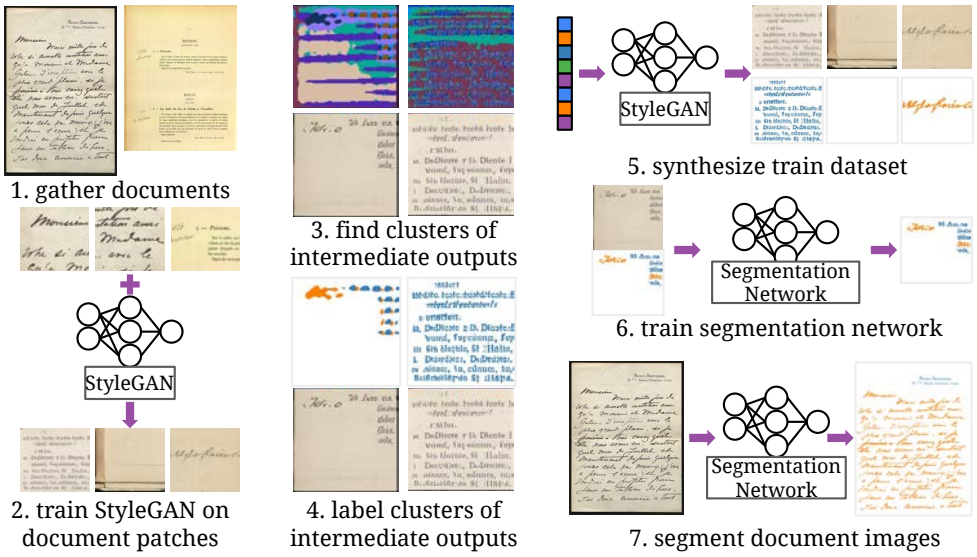


Figure 4.16: Our proposed pipeline for the synthesis of training data for pixel-wise semantic segmentation of historical document images. Our pipeline consists of 7 steps, most of which can run without manual intervention. 1. We gather scans of documents. 2. We train a StyleGAN model to generate document patches that look as similar as possible to real patches extracted from our document corpus. 3. We use an unsupervised clustering algorithm on the intermediate outputs of the synthesis network of our trained StyleGAN model. 4. Found clusters need to be annotated manually. 5. We use the StyleGAN model from step 2 and information about the classified clusters to synthesize a training dataset. 6. We use the synthesized training data to train an off-the-shelf segmentation network on patches of documents. 7. We apply the trained segmentation network on the real document images and obtain a segmented image.

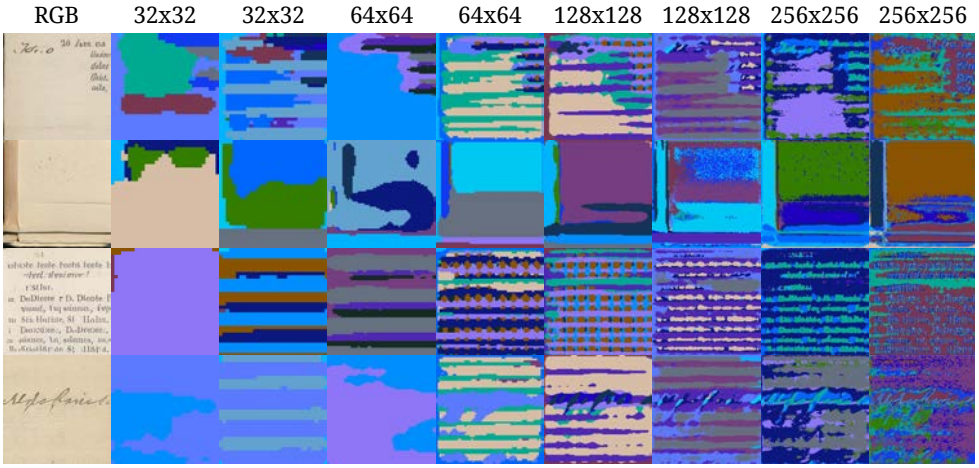


Figure 4.17: Clusters were found by applying spherical k-Means to predict 20 clusters on the activations of intermediate synthesis network layers. The left-most column shows the RGB image generated by the model. The other columns show intermediate feature maps of different sizes. The number on top of each column represents the spatial size of the feature map. The right-most column is the last feature map before the output of the RGB image. It is visible that the intermediate activations correspond to specific classes in the output image (best viewed digitally in color). Feature maps of smaller spatial sizes tend to encode the semantic class of pixels, whereas feature maps of greater spatial size tend to encode textures.

of a sample, StyleGAN encodes the semantic class of pixels in the intermediate layers of the synthesis network. This behavior was first described in [65] where Collins *et al*. used this insight to perform semantically meaningful local edits on faces. We go a step beyond their findings in our work and use the information encoded in StyleGAN to produce a fully annotated dataset. To do this, we apply an unsupervised clustering algorithm, such as spherical k-Means [50] to the activations of each StyleGAN block in the synthesis network.

When following this approach and training a StyleGAN on patches of document images that contain handwritten and printed text (see step 2 in figure 4.16), we can get the clustering result that we depict in figure 4.17. The provided samples show that specific activations of StyleGAN blocks in the synthesis network seem to encode the semantic class of printed or handwritten text. However, we can not directly use the clusters for semantic segmentation and classification simultaneously for several reasons. On the one hand, multiple clusters belong to the same class. On the other hand, it is not possible to use only one layer of the intermediate activations because the resolution/accuracy of the identified text regions could be very low if we use the intermediate output of an early layer in the synthesis network. We can

also not always rely on the classes predicted in a single layer. Thus, we have to take the output of multiple layers into account when creating our annotation image.

To remedy these problems, we propose two steps that require human intervention. Although we require human intervention, a human annotator's amount of time and work required to be performed is minimal. First, we require a human annotator to examine the clusters found by the clustering algorithm and categorize them. We could theoretically label all clusters by examining only one image since we determine the clusters for each StyleGAN block individually based on the activations over 100 images. However, no individual image contains enough information to annotate the found clusters reliably. Instead, we found that examining around 100 images suffices to determine the classes of clusters accurately. To speed up the annotation process, we provide a web-based annotation user interface (UI). Our web-based UI provides the annotator with the synthesized RGB image and all clustered StyleGAN block outputs. There, the user can label individual clusters by clicking them. If the user labels a cluster of one StyleGAN block found in one RGB image, he also labels the same cluster for all other images of the same StyleGAN block. Thus, over time, the user receives a more and more annotated view of new images and can then verify whether the annotation decisions are correct, simplifying the annotation process.

The second step we require a human annotator to perform is the formulation/adaption of an algorithm that decides which pixel of the resulting label image belongs to which class. This algorithm is required because it does not always suffice to take the information in the outputs of a single StyleGAN block. After all, we can not control what and how the underlying StyleGAN model learns and which clusters we find. Instead, in many cases, the information from several blocks need to be combined. The combination of information is essential to obtain fine-grained segmentation results. However, the design of the algorithm can also be simplified because the combination of multiple layers follows reoccurring patterns. In the following, we explain how we design an algorithm based on some exemplary classified intermediate outputs that we also show in figure 4.18. The process mainly consists of two steps. First, all StyleGAN blocks that can be used to determine the semantic class of pixels need to be determined and correctly combined. Second, we need to determine all StyleGAN blocks that can be used to produce the most fine-grained segmentation possible.

Determining StyleGAN Blocks that Contain Semantic Information First, we identify the StyleGAN blocks of the synthesis network that hold information about the semantic class. We identified that usually, the StyleGAN

Figure 4.18: Classified clusters produced by intermediate layers. The clusters were produced by applying k-Means on the activations of the synthesis network of a trained StyleGAN model (see figure 4.17). The numbers on the top indicate the spatial size of the intermediate feature maps of the synthesis network in pixels. The left-most column shows the RGB images generated by the trained StyleGAN model. Blue color highlights pixels classified as printed text, orange pixels represent the handwritten text class, and white pixels depict background pixels. The clusters appear to be noisy. However, by applying a well-designed algorithm to the given feature maps, we can reduce the number of false labels by a large margin. The resulting segmentation images are shown in the right-most column.

blocks creating an intermediate activation image with spatial sizes of 64×64 and 128×128 contain information about the semantic class of a pixel. We can also observe this behavior when inspecting the corresponding columns in figure 4.17 and figure 4.18. In the case of our example depicted in figure 4.18, we distinguish between the two classes printed text (orange color) and handwritten text (light blue color). The columns of the respective spatial sizes show that we can easily distinguish between printed and handwritten text in these layers. However, we can also observe some problems. The predictions for handwritten text at the spatial size of 64×64 are noisy (see the first and last row), but we can correct most of these noisy predictions by keeping only the areas of the same class from both feature maps that have an intersection over union (IOU) greater than 0. Utilizing this metric, we can also drop the noisy predictions shown in the second row of the second column. The model we are analyzing here always predicts a handwriting region directly right of printed text regions, which is a problem. We could mitigate this problem by setting the number of clusters the k-Means algorithm should find to another value of k (in our example, it is set to 20), but since this behavior can be observed in every row of printed text, we can convert every occurrence of handwritten text directly right of printed text to printed text.

Obtaining Fine-Grained Segmentation Annotations The extracted semantic text regions are very coarse. In the next step, we also incorporate the results of the layers with the highest native spatial size (in our case, 256×256). Close examination of the columns in figure 4.17 and figure 4.18 shows that we can not use the two intermediate layers of spatial size 256×256 to determine semantic classes. When comparing the results of these layers with the generated RGB image, we can see that the responses we can use for labeling regions as text mostly corresponds to darker image areas. We think this is because the model only focuses on the texture, not the shape, at this network stage. This behavior is commonly found in neural networks and learning theory [82], which also explains why we cannot directly distinguish between printed and handwritten text at this stage. Nevertheless, we already know the semantic classes of text regions. Thus, we can now use the information about semantic classes in conjunction with the fine-grained information to create a fine-grained annotation image. Here, we use the activations from the StyleGAN blocks of spatial size 256×256 and classify each text region using the already found semantic regions from the previous step to create a fine-grained annotation image, which is shown in the *result* column of figure 4.18.

4.5.3.3 Synthesis of a Large Scale Dataset

Once we adapted our labeling algorithm for a specific StyleGAN model, we can synthesize our dataset. The synthesis of our model is a simple process. First, we draw a random vector z from the input distribution $\mathcal{Z}$ run this vector through the mapping network of StyleGAN, obtaining $w \in \mathcal{W}$. Then we run everything through the synthesis network and obtain the activations of each StyleGAN block in the synthesis network and the corresponding RGB image. We then use the already obtained cluster centers to determine the cluster each pixel of each activation belongs to. Following the cluster assignment, we use our adapted classification algorithm to annotate each pixel and obtain an annotation image. The resulting dataset might be imbalanced, meaning that many samples of one class are present. We can mitigate any imbalances by balancing the dataset in a post-processing step since it is simple to determine the classes contained in each synthesized image because we now have access to a fine-grained annotation image where each color encodes the class of a pixel.

4.5.4 Pixel-wise Segmentation System

Following the creation of an annotated dataset, we can now use the dataset and train a semantic segmentation model using off-the-shelf semantic segmentation networks (step 6 in figure 4.16). We also propose a post-processing method to reassemble or stitch the predicted patches to obtain a fully segmented document image.

Semantic Segmentation Network Thanks to the availability of a large-scale synthetic dataset, we can use any segmentation network. A natural choice would be to use a segmentation network based on the U-Net architecture [197]. In document analysis, two recently introduced network architectures obtained promising results on line segmentation, which is a task close to our task. On the one hand, dhSegment by Oliveira *et al*. [182] is a U-Net-like network for documents. On the other hand, Boillet *et al*. [44] introduced a lightweight network for line segmentation that is also based on U-Net but utilizes dilated convolutions to save computations. For our experiments, we adopt the model of Boillet *et al*. and extend their architecture by removing the dropout layers and using pixel shuffle [210] instead of deconvolution layers in the decoding path of the network.

Stitching of Patches When applying our trained semantic segmentation network on real data, we need to pre-process the input image and post-process the network's outputs. In the pre-processing step, we extract patches

from the input image. The size of the patch is the same size we trained our segmentation network on. To obtain the best result, we overlap patches by a large margin to ensure each pixel is contained in at least three patches. We then perform a forward pass with each patch through our segmentation network and obtain a segmented patch. Following the forward pass, we stitch the image by performing a majority voting of the three most confident predictions for each pixel.

4.5.5 Experiments

The idea behind our proposed system is to apply semantic segmentation on new unlabeled datasets without the need for much manual and costly annotation effort. In our experiments, we apply our entire proposed segmentation pipeline on data from the archive of the WPI. The data is not annotated at all. Thus, we are only able to report qualitative results. In the following, we introduce the dataset of the WPI; we then provide detailed information about our experimental setup. We finish by reporting and discussing the results on some representative examples of the dataset by the WPI.

4.5.5.1 The WPI Dataset

There are, to the best of our knowledge, no public datasets for pixel-wise semantic segmentation available that include a large training dataset, as well as an evaluation dataset. Some large-scale datasets are available for the task of line segmentation [43, 88]. The HORAE dataset [43] even has a large amount of unannotated data available for training, but the ground-truth annotation of these evaluation datasets is too coarse for our task and our model. Thus, we evaluate our proposed model on data that we obtained from the WPI. The vast majority of the data used by us for our experiments is available online on the web pages of the WPI.[10] For now, only the raw document scans are available. In the future, we wish to provide an annotated evaluation dataset based on a subset of the data of the WPI.

For the training of our StyleGAN model that we use to create our dataset for the training of the segmentation model, we used pages from over 11 000 sales catalogs totaling in 722 094 scanned pages. The scanned catalogs come from a considerable period starting in the 17th century and ending in 1959. The data is available on the WPI website.[11]

For the qualitative evaluation, we used several auction catalogues containing handwritten annotations, as well as the very diverse (in terms of document genres) *Paul Ferdinand Gachet and Paul Louis Gachet Papers* [106] dating from

[10] https://digitalprojects.wpi.art (last accessed August 31, 2021).
[11] https://digitalprojects.wpi.art/auctions (last accessed August 31, 2021).

1816 to 1962, as well as the *Stock Books and Inventories* of the Ambroise Vollard Records [107] dating from 1899 to 1938.

4.5.5.2 Experimental Setup

In the following, we introduce our experimental setup. We describe all preprocessing steps, our hardware setup, and hyperparameters and steps applied during inference.

We preprocess our training data by resizing all images so that the largest side has a size of 2000 pixels. Then, we split the resized pages into patches of 256×256 pixels and stop creating further patches after obtaining 5.2 million patches. As a further preprocessing, we apply our handwriting determination model introduced in section 4.3 to balance the dataset leaving us with roughly the same amount of patches that include handwriting and patches that do not contain handwriting. After balancing, we obtain 1 036 778 samples for the training of our StyleGAN model. Currently, we do not use the entire available dataset. However, future experiments could also be performed with a balanced version of the entire dataset.

In principle, all experiments can be performed on a system with a GPU that has at least 11 GB of RAM, *e.g.*, an NVIDIA Gefore 1080Ti, when using a patch size of 256×256. However, we use a range of different machines for our experiments. We train our StyleGAN model on a DGX-1 with 8 NVIDIA V100 GPUs. We follow the hyperparameters for the training of a StyleGAN 2 model by Karras *et al*. [130]. Nevertheless, we adjust the hyperparameters as follows: We set the initial learning rate to 0.001, the number of iterations to 100 000, the batch size to 16, and we use cosine annealing [158] for the learning rate updates. Following the training of StyleGAN, we analyze the StyleGAN model as described in section 4.5.3.2 using 100 samples synthesized by the trained StyleGAN model. Following our analysis and adaption of the segmentation algorithm, we synthesize 400 000 samples for the training of the segmentation network. For segmentation, we use Doc-UFCN as proposed by Boillet *et al*. [44], where we apply small changes to the network architecture. In our model, we do not use dropout, and we do not use deconvolutional layers for upscaling but use Pixelshuffle [210] for the upscaling of feature maps. For the training of Doc-UFCN, we set the initial learning rate to 0.005, the batch size to 16 and train the model for 50 epochs. Here, we also use cosine annealing to update the learning rate during training.

During inference, we perform the following steps: First, we resize the input image so that the largest side has a size of 2000 pixels while keeping the aspect ratio. Then, we split the image into patches that have an overlap of 50 % each. Following the split, we feed each image as input to the trained Doc-UFCN model obtaining the segmentation result for each patch. During

post-processing, we assemble individual patches by taking, for each pixel, the class that was predicted most often. We can use such a voting procedure because the patches overlap, with an overlap ratio of 50 %, we have a maximum of 4 overlapping patches that we analyze using a majority voting setting. Following the voting, we save the resulting segmentation image.

4.5.5.3 Qualitative Results

Since no annotated evaluation dataset is available, we resort to a preliminary qualitative evaluation of the performance of our model. In Figures 4.19 and 4.20, we provide samples and the corresponding prediction of our model. We show results where our model shows promising performance on a range of challenging document images in figure 4.19 while we also provide failure cases of our model that show typical problems with the current state of our proposed model in figure 4.20. In the following, we discuss the provided results.

Success Cases In figure 4.19, we provide four typical examples of documents that can be found in the archive of the WPI. First, we can find scanned pages that only contain printed text in the first row. The results show that we can reliably segment the printed text on the page and classify it correctly. We can further see that our model is not distracted by the colorful book cover seen in some parts of the scan. The example in the second row shows a document entirely written by hand. This document is challenging because of the ink bleeding on both sides of the page. We can see that our model successfully segmented only the text regions that belong to the handwritten text on the page, showing high accuracy. The example in the third row is a document that contains a mixture of printed and handwritten text, making it one of the documents we are most interested in. We can see that our model successfully correctly segments most parts of the image and also classifies handwritten and printed text correctly. The example in the last row shows another challenging document. Here, we have a document with many lines and also an image. Printed text is below the image. Our segmentation model can correctly ignore everything that is not printed text showing that our model can learn to ignore images and figures. The results show that our proposed method is viable and can create a segmentation model directly on unannotated data.

Failure Cases Although the success cases already show good performance, we can also identify samples where our current model fails. Here, we identify typical problem cases that arise from our current system design. In figure 4.20, we provide some samples that show such problems. In the first row, we provide a sample from an inventory of the Ambroise Vollard Records. At first glance,

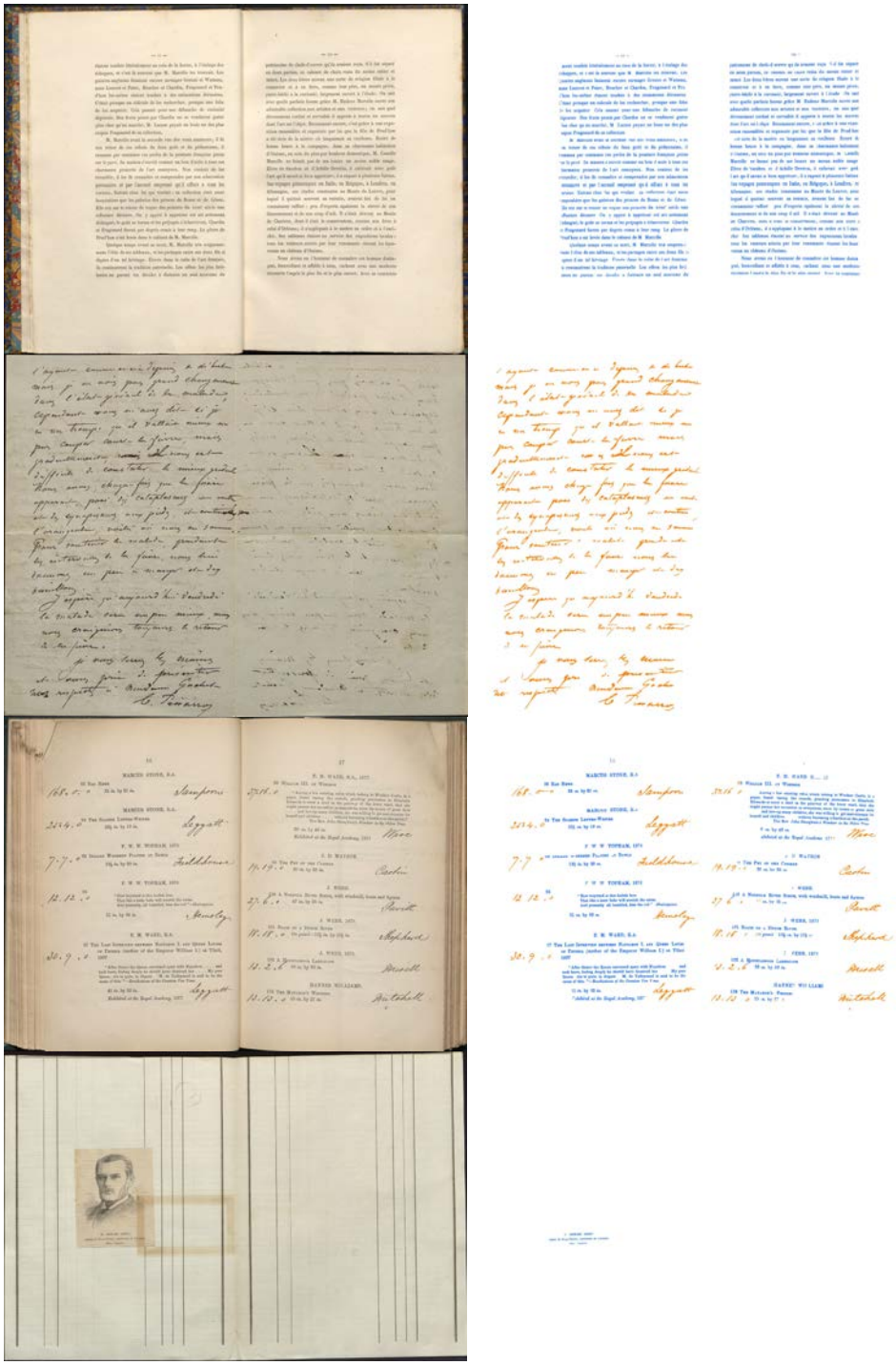


Figure 4.19: Success cases of our proposed model. We show samples illustrating all different use cases, documents with printed text only, documents with handwriting only, documents containing a mixture of handwritten and printed text, and documents with distractors. We always depict the input image on the left, while on the right, we depict the segmentation result. Blue and orange denote printed text and handwritten text, respectively.

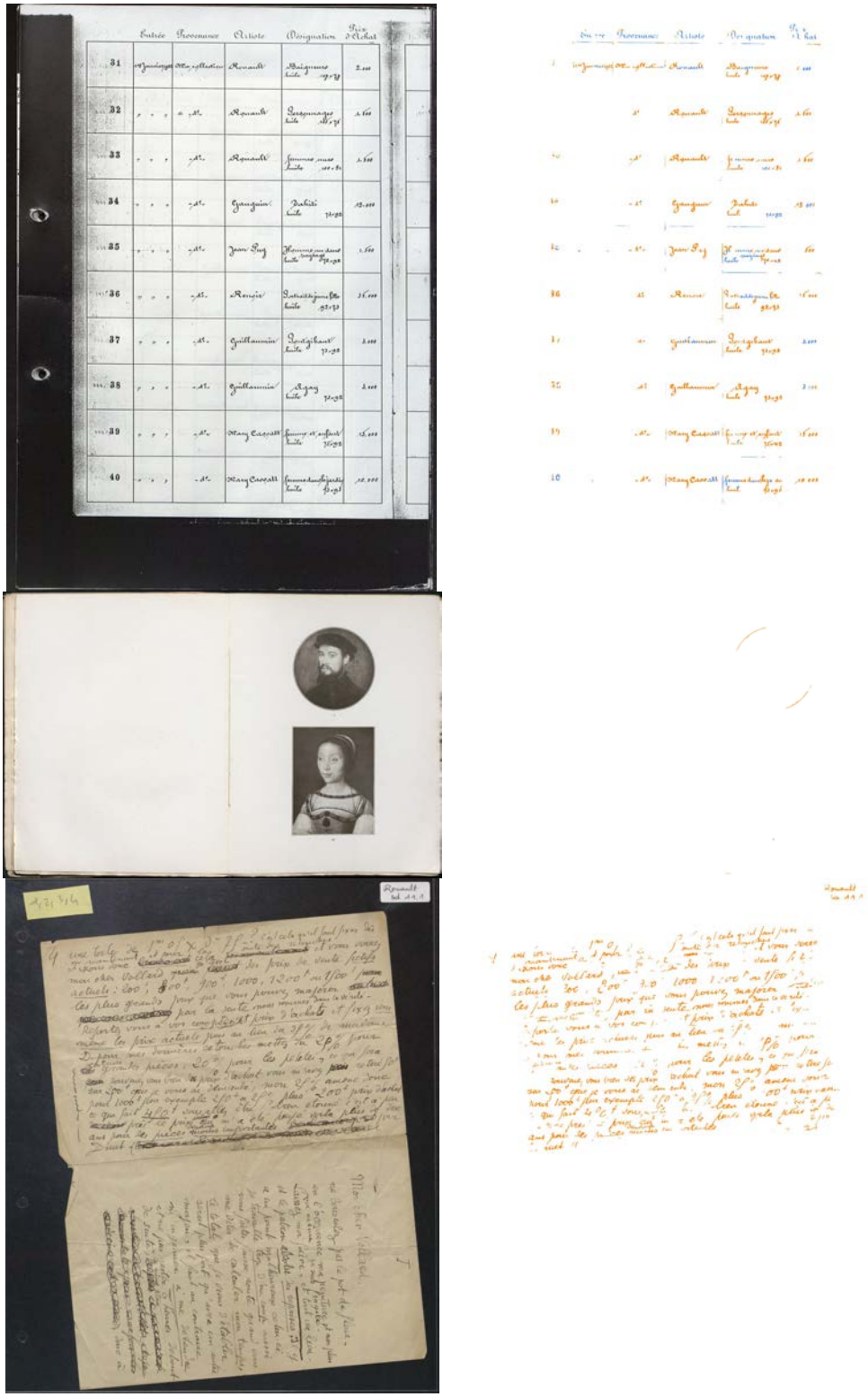


Figure 4.20: Although our model performs well, certain problems persist. Here, we depict images that show typical problems of our current model. Again, blue and orange denote printed and handwritten text, respectively.

the entire document seems to be written by hand. However, the regularity of writing might also indicate printed text. As we can see, the model is not able to correctly distinguish between printed and handwritten text. Although the segmentation picks up nearly every part of the text, the classification is not always correct and indecisive. We argue that the classification accuracy is not very good because no such documents have been in the training data (recall that we trained the segmentation model only on auction catalogs). In this sample, we can also see another problem. The recall of our model is not always perfect; this is primarily due to the background class predictions being too confident. We can solve such problems using more training data (which we have available) and changes in the network architecture and patch voting, as currently, margins of patches mostly show bad segmentation results. The image in the second row is an image taken from an auction catalog. Although our model can ignore the images on the page successfully, it predicts the round border of the top image as handwriting. Such a prediction shows us that the model learned that handwriting consists of round shapes. We are confident that we can overcome such failure cases by incorporating more training data, including round images. We further think that it might be helpful to create several models specialized to specific periods. In the third sample, we show another sample of the Ambroise Vollard Records. Here, we can see that our model picks up handwritten text very well. However, it misses all rotated handwritten text because we did not have any training data with rotated text. We plan to overcome such issues by adding more data augmentation.

4.5.6 Discussion

Our experimental results demonstrate that our proposed method is viable. We further show that it is indeed possible to create deep learning models directly using unannotated data and a small amount of manual annotation work. While our success cases show that the model entirely trained on synthetic data works in many use cases, the failure cases show that most problems are related to the training data. The most severe problem with our approach is that we can not directly control the training data synthesized by our model. Instead, we can control the input to the StyleGAN model we train, which provides us much flexibility. However, this also means that the current approach involves trial and error procedures to get a well-performing generative model. We plan to incorporate new research results in the area of unconditional image generation [81, 128] to allow the creation of even better generative models.

Our fragmentation of the input images into patches proves fruitful but also disadvantageous at the same time. On the one hand, we can analyze images of any size without compromises in resolution. Further, we can balance our

training data for all models using our model introduced in section 4.3. On the other hand, we observe that especially text at the margins of patches is wrongly classified with strong confidence. To mitigate such issues, we need to improve our patch analysis and overlapping strategy further.

Our current evaluation is only based on qualitative measures but no objective quantitative measures. While quantitative measures are out of the scope of this thesis, we will add quantitative measures in future work. We plan to do this by semi-automatically annotating a set of images from the publicly available archives of the WPI.

4.5.7 Summary

In this section, we proposed a novel approach to synthesize large-scale training datasets for pixel-wise semantic segmentation of historical document images. Our approach can directly be applied to raw image scans without any annotations of individual document images. We make use of the generative power of modern unconditional GANs and especially StyleGAN to synthesize patches of document images together with their corresponding annotation for pixel-wise semantic segmentation. We then use the synthesized data to train off-the-shelf semantic segmentation networks. We perform a qualitative analysis of our trained segmentation models on real-world document images from the archive of the WPI. Our results show that our approach is viable and can be applied to real-world data but our approach still needs fine-tuning. We also need to develop an evaluation dataset to quantitatively evaluate our models, which is out of the scope of this thesis.

4.6 Chapter Summary

In this chapter, we presented a range of methods for the analysis of historical document images. Our solutions focus on one of the most pressing problems in historical document analysis: the availability of annotated training data. To this end, we proposed three methods to aid the analysis of historical documents that are entirely based on the usage of synthetic data to train our deep machine learning models. First, we proposed to use patches with synthetic printed and handwritten text training a model to determine whether a given scanned page contains any handwriting. Second, we used synthetic data generated by a generative model to synthesize handwriting images for the training of several models for the classification of handwriting. Here, we proposed using a fixed softmax classifier and a more flexible classification model based on metric learning to semantically classify the content of a handwritten word image into categories such as date, number, alpha-numeric, *etc*. Third, we

proposed a novel approach for the synthesis of ground-truth data for pixel-wise semantic segmentation. Our proposed model directly operates on the available unannotated document images and uses generative properties of GANs such as StyleGAN.

Our proposed models are independent of manual per-image annotations and scale well with the available data. Our models are flexible, which is an essential factor in historical document analysis because the manual annotation of data is costly and time-intensive. The costs of manual annotation in historical document analysis are especially severe because, for most annotation work, it is necessary to involve experts, which are not always available. We note that our current work can only be the "tip of the iceberg" and much more research on the application and how to obtain synthetic training data for the analysis of archives is necessary for future work.

5 Conclusions and Future Work

In this thesis, we researched and introduced several approaches with the objective to reduce the costs of acquiring ground truth annotations for the application of state-of-the-art machine learning methods to optical character recognition pipelines. First, we investigated how we can reduce the annotation cost by using only a fraction of the typically required ground truth annotations for the use case of end-to-end scene text recognition. Second, we investigated how we can use synthetic data to reduce the amount of manual annotation work for the use case of document analysis for archival material. In the following, we summarize our conclusions and contributions presented in this work. Following our conclusions, we propose directions for future research.

5.1 Conclusion

Deep learning has become the de-facto method for OCR in the past few years. Methods based on deep learning are robust and highly accurate. However, robustness and accuracy come with a price. On the one hand, the training of deep neural models requires the usage of high-performance compute accelerators, such as modern GPUs. It is, on the other hand, expensive to obtain training data that can be used to fit and build neural networks for a specific application, such as OCR.

In this thesis, we presented solutions to alleviate the costs of obtaining annotated training data. Specifically, we can summarize the main achievements of this work as follows:

In chapter 3, we introduce a novel weakly supervised approach for the recognition of scene text. We train our proposed approach on publicly available synthetic data and also synthetic data that we generated ourselves. Our proposed approach consists of two neural networks that work together. The first network is tasked with the localization of text areas, such as words or individual characters. The task of the second network is the recognition of the textual content in the localized text areas. Both networks are trained together under the supervision of textual annotation only! We do not need any annotation for the location of words or individual characters, making our approach weakly supervised. Our proposed approach can be used for a multitude of tasks. First, we can use our proposed approach for end-to-end

scene text recognition, where the task is to extract the textual content of all words in an image. Second, we can use our approach directly for the task of scene text recognition. Here, the task is to recognize the textual content of the given word image. In our experiments, we show that our proposed approach can reach competitive results for the task of end-to-end scene text recognition of the French Street Name Signs (FSNS) dataset. In contrast to related work, our model is able to provide users with information about the locations of words and the textual content not only the textual content of individual words. We further show that our model applied for text recognition only is able to achieve state-of-the-art, as well as competitive results on several scene text recognition benchmark datasets, such as ICDAR 2015-1811 (IC15-1811), IIIT5K-Words (IIIT5K), Steet View Text Perspective (SVTP), and CUTE80 (CUTE). Here, our model shows especially good performance on irregular scene text recognition datasets showing the benefit of our combined localization and recognition approach.

In chapter 4, we introduce a set of analysis steps for OCR of archival data. One of the most pressing problems in historical document analysis is the availability of annotated training data. In contrast to tasks such as the classification of natural images, it is not always possible to use crowdsourcing to gather large-scale training datasets. One the one hand, the usage crowdsourced information is not viable because experts are necessary to produce correct annotations. On the other hand, institutions might not have the financial possibilities to start a large-scale annotation effort using the crowd. In this line, we present three approaches that use synthetic data to analyze handwriting in document images. These approaches resemble only parts of an entire document analysis pipeline, as the development of a whole document analysis pipeline is out of the scope of this thesis.

First, we introduce a novel preprocessing step (see section 4.3) where we determine whether a given document page contains any handwriting. To do this, we propose a data synthesis strategy that allows us to synthesize training data for a deep neural network. The trained neural network is tasked to determine whether a given image patch contains handwriting or not. Applying our trained model on document images lets us determine whether a page contains handwriting; we also receive the rough location of the handwriting regions. In our experiments on a real-world dataset with data gathered from the archive of the WPI, we obtain an F1-Score of 0.98.

Second, we present an analysis task that can be used instead of, or in conjunction with, a handwriting text recognition model (see section 4.4). We propose to perform a handwriting classification step as an additional analysis step before the actual recognition of text. The classification step takes an image of a cropped handwritten word as input and categorizes the image into

classes, such as date, number, alphanumeric string, word, or zip code. The classification is entirely based on the visual structure of the word image and does not involve any textual annotations; only class annotations are required. Here, we also use synthetic data to train our models. We synthesize our data using online handwriting information, as well as generative neural networks. We train multiple models and evaluate them thoroughly, investigating their accuracy, flexibility, and generalizability. We find that a simple softmax-based classification model reaches the best accuracy. However, we also show that a model based on metric learning and embeddings is very flexible and can distinguish between previously unseen classes.

Third, we propose a novel data synthesis approach for the generation of data for pixel-wise semantic segmentation, especially in document images (see section 4.5). Our synthesis approach uses the inner structure and knowledge of generative models, such as the StyleGAN architecture. We can train our model directly on the raw image data obtained from any archive. There is no need for any previous annotations. We first train a StyleGAN model on the raw archival images. Later, we use the StyleGAN model to simultaneously synthesize RGB and annotation images to train a segmentation model. Our qualitative analysis shows that our proposed approach is viable and can analyze real data from an archive. However, further research is necessary to improve the proposed approach and further alleviate manual efforts.

5.2 Future Research Directions

We think reducing the amount of manual annotation work for the application of deep neural networks will be an essential aspect of future computer vision research endeavors. Having access to methods that can produce a massive amount of annotated training data in a relatively short time with low financial effort will make it possible to build new computer vision applications on data where it was not possible before. To this end, we proposed methods that can be used to take some steps in such directions. Besides developing synthetic data approaches, we think that the development of weakly supervised or unsupervised methods is also essential because such methods require even less annotated data. In the following, we provide our thoughts about possible extensions of our approaches.

5.2.1 Weak Supervision for Scene Text Detection and Recognition

Current scene text recognition approaches achieve encouraging results with the usage of synthetic data. We have shown that our proposed model can handle even complex scene text entities very well. However, so far, we were

only able to perform end-to-end scene text recognition experiments on a rather "simple" dataset. In the future, a way to simplify the training process of the weakly supervised text localizer should be researched. We already performed first research in this direction and published this research in [30]. However, we think that an auspicious research direction would be the combination of our results presented in [30] with reinforcement learning to learn text detection in a weakly supervised manner.

In pure scene text recognition, we think that our proposed model can further be improved by adding better possibilities to capture rotations. We might be able to better capture rotations by providing more rotated text in our synthetic train datasets. We might also capture rotations better if we use a different strategy to predict the transformation parameters. One possible solution would be the usage of thin-plate splines because they have more degrees of freedom. Nevertheless, even when using thin-plate splines, using more training data is necessary.

5.2.2 Archival Analysis

The methods proposed in this work are only a tiny part of an entire document analysis pipeline. The most severe problem in deep learning for archive analysis still exists: the availability of annotated training data. Although we proposed some approaches that make it simpler to obtain annotated training data by synthesizing new data, there is still much room for improvement.

In the case of our proposed handwriting classification approach, we think that it might be worthwhile to examine the applicability of recently proposed unsupervised classification algorithms [103, 117, 228]. Preliminary experiments show that we can reach an on par classification result on our synthetic validation dataset compared to the softmax model when using a model trained using the training scheme introduced in [117]. To successfully apply handwriting classification on the archival data of the WPI, it is further necessary to gather a large-scale unsupervised training set. We hope to gather such a dataset once we obtain a well-performing semantic segmentation model to extract single handwritten words directly from the archive. Further application areas of our handwriting classification approach could be language classification to identify the correct recognition model for a given handwritten word.

We see the need to decrease the manual effort to a minimum for our work on pixel-wise semantic segmentation. To minimize manual effort, we think it might be helpful to automate as many parts of the manual clustering as possible. Here, we could automatically label some clusters that show the distinctive properties of background clusters, *i.e.*, one cluster fills the

entire patch. Further, we think that we could simplify the adjustment of the synthesis algorithm. Either by providing a simple pipeline tool that allows the combination of several steps or by training a neural network. Besides such optimizations, it is necessary to gather an evaluation dataset to obtain quantitative results and not just qualitative results to measure the improvement and applicability of the presented model. We further think that it might be possible to mix our results presented in [23] with our semantic segmentation model to directly segment original images without the need to train a segmentation model on synthetic data. In this case, we would only need to train a StyleGAN model and then embed images into this model's latent and stochastic noise space and read the semantic information from the network while it reconstructs the image. We also wish to investigate whether our proposed approach is also applicable to other semantic segmentation tasks such as tumor segmentation. Having access to such an approach for medical data could be of high benefit because no (large-scale) manual annotations would be necessary, and it would be possible to use synthetic data to train machine learning models while keeping patient data more private.

In the future, we also plan to investigate further analysis steps besides the recognition of handwriting information from a cropped word image. We think that it makes sense to create a document classification system that uses the raw scanned page as input and classifies whether the image's content is a letter, book, image, *etc*. Such a document classification system could help to classify document stacks in a fine-grained way. Here, we think that the most critical problem to solve is the availability of annotated training data. Since it is challenging to synthesize full document images, we think the best course of action might be to use unsupervised document classification algorithms.

All in all, we think that the work presented in this thesis can only be a first step in the usage and creation of synthetic training data for OCR. Therefore, much research is required to push the knowledge about deep learning and applications of deep learning and improve the day-to-day research work of, *e.g.*, historians.

Bibliography

[1] *1st International Workshop on Document Image Analysis for Libraries (DIAL 2004), 23-24 January 2004, Palo Alto, CA, USA*. IEEE Computer Society, 2004. ISBN: 0-7695-2088-X.

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. "Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?" In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pages 4432–4441. DOI: 10.1109/ICCV.2019.00453.

[3] Rameen Abdal, Yipeng Qin, and Peter Wonka. "Image2StyleGAN++: How to Edit the Embedded Images?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pages 8296–8305. DOI: 10.1109/CVPR42600.2020.00832.

[4] Sherif Abuelwafa, Marco Pedersoli, and Mohamed Cheriet. "Unsupervised Exemplar-Based Learning for Improved Document Image Classification". In: *IEEE Access* 7 (2019), pages 133738–133748. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2940884.

[5] Chandranath Adak, Simone Marinai, Bidyut B. Chaudhuri, and Michael Blumenstein. "Offline Bengali Writer Verification by PDF-CNN and Siamese Net". In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. Apr. 2018, pages 381–386. DOI: 10.1109/DAS.2018.33.

[6] Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M. Breuel, Andreas Dengel, and Marcus Liwicki. "Deepdocclassifier: Document Classification with Deep Convolutional Neural Network". In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. Aug. 2015, pages 1111–1115. DOI: 10.1109/ICDAR.2015.7333933.

[7] Muhammad Zeshan Afzal, Andreas Kölsch, Sheraz Ahmed, and Marcus Liwicki. "Cutting the Error by Half: Investigation of Very Deep CNN and Advanced Training Strategies for Document Image Classification". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (Nov. 2017), pages 883–888. DOI: 10.1109/ICDAR.2017.149. arXiv: 1704.03557.

[8] Reem Alaasam, Berat Kurar, and Jihad El-Sana. "Layout Analysis on Challenging Historical Arabic Manuscripts Using Siamese Network". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 738–742. DOI: 10.1109/ICDAR.2019.00123.

[9] Eloi Alonso, Bastien Moysset, and Ronaldo Messina. "Adversarial Generation of Handwritten Text Images Conditioned on Sequences". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 481–486. DOI: 10.1109/ICDAR.2019.00083.

[10] N. S. Altman. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression". In: *The American Statistician* 46.3 (Aug. 1992), pages 175–185. ISSN: 0003-1305. DOI: 10.1080/00031305.1992.10475879.

[11] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Volume 70. Proceedings of Machine Learning Research. 2017, pages 214–223. arXiv: 1701.07875.

[12] Martín Arjovsky and Léon Bottou. "Towards Principled Methods for Training Generative Adversarial Networks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[13] Muhammad Nabeel Asim, Muhammad Usman Ghani Khan, Muhammad Imran Malik, Khizar Razzaque, Andreas Dengel, and Sheraz Ahmed. "Two Stream Deep Network for Document Image Classification". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 1410–1416. DOI: 10.1109/ICDAR.2019.00227.

[14] Emmanuel Augustin, Jean-marie Brodin, Matthieu Carré, Edouard Geoffrois, Emmanuèle Grosicki, and Françoise Prêteux. "RIMES Evaluation Campaign for Handwritten Mail Processing". In: *Proc. of the Workshop on Frontiers in Handwriting Recognition*. 1. 2006.

[15] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *arXiv:1607.06450 [cs, stat]* (July 2016). arXiv: 1607.06450 [cs, stat].

[16] Youngmin Baek, Bado Lee, Dongyoon Han, Sangdoo Yun, and Hwalsuk Lee. "Character Region Awareness for Text Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pages 9365–9374. DOI: 10.1109/CVPR.2019.00959.

[17] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[18] Fan Bai, Zhanzhan Cheng, Yi Niu, Shiliang Pu, and Shuigeng Zhou. "Edit Probability for Scene Text Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pages 1508–1516.

[19] Souhail Bakkali, Zuheng Ming, Mickael Coustaty, and Marcal Rusinol. "Visual and Textual Deep Feature Fusion for Document Image Classification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pages 562–563.

[20] Souhail Bakkali, Zuheng Ming, Mickaël Coustaty, and Marçal Rusiñol. "Cross-Modal Deep Networks For Document Image Classification". In: *2020 IEEE International Conference on Image Processing (ICIP)*. Oct. 2020, pages 2556–2560. DOI: 10.1109/ICIP40778.2020.9191268.

[21] Duhyeon Bang and Hyunjung Shim. "MGGAN: Solving Mode Collapse Using Manifold Guided Training". In: *arXiv:1804.04391 [cs]* (Apr. 2018). arXiv: 1804.04391 [cs].

[22] Christian Bartz, Joseph Bethge, Haojin Yang, and Christoph Meinel. "KISS: Keeping It Simple for Scene Text Recognition". In: *arXiv:1911.08400 [cs]* (Nov. 2019). arXiv: 1911.08400.

[23] Christian Bartz, Joseph Bethge, Haojin Yang, and Christoph Meinel. "One Model to Reconstruct Them All: A Novel Way to Use the Stochastic Noise in StyleGAN". In: *arXiv:2010.11113 [cs, eess]* (Oct. 2020). arXiv: 2010.11113 [cs, eess].

[24] Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. "Language Identification Using Deep Convolutional Recurrent Neural Networks". In: *Neural Information Processing*. Edited by Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pages 880–889. ISBN: 978-3-319-70136-3. DOI: 10.1007/978-3-319-70136-3_93.

[25] Christian Bartz, Nitisha Jain, Tobias Bredow, Emanuel Metzenthin, Jona Otholt, and Ralf Krestel. "Semantic Analysis of Cultural Heritage Data: Aligning Paintings and Descriptions in Art-Historic Collections". In: *Pattern Recognition. ICPR International Workshops and Challenges*. Edited by Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and

Roberto Vezzani. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pages 517–530. ISBN: 978-3-030-68796-0. DOI: 10.1007/978-3-030-68796-0_37.

[26] Christian Bartz, Nitisha Jain, and Ralf Krestel. "Automatic Matching of Paintings and Descriptions in Art-Historic Archives Using Multimodal Analysis". In: *Proceedings of the 1st International Workshop on Artificial Intelligence for Historical Image Enrichment and Access*. Marseille, France: European Language Resources Association (ELRA), May 2020, pages 23–28. ISBN: 979-10-95546-63-4.

[27] Christian Bartz, Hendrik Rätz, and Christoph Meinel. "Handwriting Classification for the Analysis of Art-Historical Documents". In: *Pattern Recognition. ICPR International Workshops and Challenges*. Edited by Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pages 562–576. ISBN: 978-3-030-68796-0. DOI: 10.1007/978-3-030-68796-0_40.

[28] Christian Bartz, Hendrik Rätz, Haojin Yang, Joseph Bethge, and Christoph Meinel. "Synthesis in Style: Semantic Segmentation of Historical Documents Using Synthetic Data". In: *arXiv:2107.06777 [cs]* (July 2021). arXiv: 2107.06777 [cs].

[29] Christian Bartz, Laurenz Seidel, Duy-Hung Nguyen, Joseph Bethge, Haojin Yang, and Christoph Meinel. "Synthetic Data for the Analysis of Archival Documents: Handwriting Determination". In: *2020 Digital Image Computing: Techniques and Applications (DICTA)*. Nov. 2020, pages 1–8. DOI: 10.1109/DICTA51227.2020.9363410.

[30] Christian Bartz, Haojin Yang, Joseph Bethge, and Christoph Meinel. "LoANs: Weakly Supervised Object Detection with Localizer Assessor Networks". In: *Computer Vision – ACCV 2018 Workshops*. Edited by Gustavo Carneiro and Shaodi You. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pages 341–356. ISBN: 978-3-030-21074-8. DOI: 10.1007/978-3-030-21074-8_29.

[31] Christian Bartz, Haojin Yang, and Christoph Meinel. "SEE: Towards Semi-Supervised End-to-End Scene Text Recognition". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. Apr. 2018.

[32] Christian Bartz, Haojin Yang, and Christoph Meinel. "STN-OCR: A Single Neural Network for Text Detection and Text Recognition". In: *arXiv:1707.08831 [cs]* (July 2017). arXiv: 1707.08831 [cs].

[33] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded Up Robust Features". In: *Computer Vision – ECCV 2006*. Edited by Aleš Leonardis, Horst Bischof, and Axel Pinz. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pages 404–417. ISBN: 978-3-540-33833-8. DOI: 10.1007/11744023_32.

[34] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy Layer-Wise Training of Deep Networks". In: *Proceedings of the 19th International Conference on Neural Information Processing Systems*. NIPS'06. Cambridge, MA, USA: MIT Press, Dec. 2006, pages 153–160.

[35] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. "Curriculum Learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM, 2009, pages 41–48. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553380.

[36] Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. "MeliusNet: An Improved Network Architecture for Binary Neural Networks". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pages 1439–1448.

[37] Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. "MeliusNet: Can Binary Neural Networks Achieve MobileNet-Level Accuracy?" In: *arXiv:2001.05936 [cs, stat]* (Mar. 2020). arXiv: 2001.05936 [cs, stat].

[38] Joseph Bethge, Christian Bartz, Haojin Yang, and Christoph Meinel. "BMXNet 2: An Open Source Framework for Low-Bit Networks - Reproducing, Understanding, Designing and Showcasing". In: *Proceedings of the 28th ACM International Conference on Multimedia*. MM '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pages 4469–4472. ISBN: 978-1-4503-7988-5. DOI: 10.1145/3394171.3414539.

[39] Joseph Bethge, Haojin Yang, Christian Bartz, and Christoph Meinel. "Learning to Train a Binary Neural Network". In: *arXiv:1809.10463 [cs, stat]* (Sept. 2018). arXiv: 1809.10463 [cs, stat].

[40] Ayan Kumar Bhunia, Abhirup Das, Ankan Kumar Bhunia, Perla Sai Raj Kishore, and Partha Pratim Roy. "Handwriting Recognition in Low-Resource Scripts Using Adversarial Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pages 4767–4776. DOI: 10.1109/CVPR.2019.00490.

[41] Alessandro Bissacco, Mark Cummins, Yuval Netzer, and Hartmut Neven. "PhotoOCR: Reading Text in Uncontrolled Conditions". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pages 785–792. DOI: 10.1109/ICCV.2013.102.

[42] James F. Blinn. "A Generalization of Algebraic Surface Drawing". In: *ACM Trans. Graph.* 1.3 (July 1982), pages 235–256. ISSN: 0730-0301. DOI: 10.1145/357306.357310.

[43] Mélodie Boillet, Marie-Laurence Bonhomme, Dominique Stutzmann, and Christopher Kermorvant. "HORAE: An Annotated Dataset of Books of Hours". In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. HIP '19. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pages 7–12. ISBN: 978-1-4503-7668-6. DOI: 10.1145/3352631.3352633.

[44] Mélodie Boillet, Christopher Kermorvant, and Thierry Paquet. "Multiple Document Datasets Pre-Training Improves Text Line Detection with Deep Neural Networks". In: *25th International Conference on Pattern Recognition, ICPR 2020, Virtual Event / Milan, Italy, January 10-15, 2021*. IEEE, 2020, pages 2134–2141. DOI: 10.1109/ICPR48806.2021.9412447.

[45] A. A. Brink, J. Smit, M. L. Bulacu, and L. R. B. Schomaker. "Writer Identification Using Directional Ink-Trace Width Measurements". In: *Pattern Recognition* 45.1 (Jan. 2012), pages 162–171. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2011.07.005.

[46] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[47] Andy Brock, Soham De, Samuel L. Smith, and Karen Simonyan. "High-Performance Large-Scale Image Recognition without Normalization". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Edited by Marina Meila and Tong Zhang. Volume 139. Proceedings of Machine Learning Research. PMLR, 2021, pages 1059–1071.

[48] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. "Signature Verification Using a "Siamese" Time Delay Neural Network". In: *Proceedings of the 6th International Conference on Neural Information Processing Systems*. NIPS'93. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Nov. 1993, pages 737–744. DOI: 10.1142/S0218001493000339.

[49] Niko Brümmer and Johan du Preez. "Application-Independent Evaluation of Speaker Detection". In: *Computer Speech & Language*. Odyssey 2004: The Speaker and Language Recognition Workshop 20.2 (Apr. 2006), pages 230–275. ISSN: 0885-2308. DOI: 10.1016/j.csl.2005.08.001.

[50] Christian Buchta, Martin Kober, Ingo Feinerer, and Kurt Hornik. "Spherical K-Means Clustering". In: *Journal of Statistical Software* 50.10 (Sept. 2012), pages 1–22. ISSN: 1548-7660. DOI: 10.18637/jss.v050.i10.

[51] Quang Anh Bui, David Mollard, and Salvatore Tabbone. "Automatic Synthetic Document Image Generation Using Generative Adversarial Networks: Application in Mobile-Captured Document Analysis". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 393–400. DOI: 10.1109/ICDAR.2019.00070.

[52] Marius Bulacu and Lambert Schomaker. "Text-Independent Writer Identification and Verification Using Textural and Allographic Features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.4 (Apr. 2007), pages 701–717. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2007.1009.

[53] Samuele Capobianco, Leonardo Scommegna, and Simone Marinai. "Historical Handwritten Document Segmentation by Using a Weighted Loss". In: *Artificial Neural Networks in Pattern Recognition*. Edited by Luca Pancioni, Friedhelm Schwenker, and Edmondo Trentin. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pages 395–406. ISBN: 978-3-319-99978-4. DOI: 10.1007/978-3-319-99978-4_31.

[54] Victor Carbune, Pedro Gonnet, Thomas Deselaers, Henry Rowley, Alexander Daryin, Marcos Calvo, Li-Lun Wang, Daniel Keysers, Sandro Feuz, and Philippe Gervais. "Fast Multi-Language LSTM-Based Online Handwriting Recognition". In: *International Journal on Document Analysis and Recognition (IJDAR)* (2020). DOI: 10.1007/s10032-020-00350-4.

[55] Cauchy, Augustin. "Methode Generale Pour La Resolution Des Systemes d'equations Simultanees". In: *Comp. Rend. Sci. Paris* 25.1847 (1847), pages 536–538.

[56] Tim Causer, Kris Grint, Anna-Maria Sichani, and Melissa Terras. "'Making Such Bargain': Transcribe Bentham and the Quality and Cost-Effectiveness of Crowdsourced Transcription1". In: *Digital Scholarship in the Humanities* 33.3 (Sept. 2018), pages 467–487. ISSN: 2055-7671. DOI: 10.1093/llc/fqx064.

[57] F. Cesarini, M. Lastri, S. Marinai, and G. Soda. "Encoding of Modified X-Y Trees for Document Classification". In: *Proceedings of Sixth International Conference on Document Analysis and Recognition*. Sept. 2001, pages 1131–1136. DOI: 10.1109/ICDAR.2001.953962.

[58] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, and Francis Bach, editors. *Semi-Supervised Learning*. Adaptive Computation and Machine Learning Series. Cambridge, MA, USA: MIT Press, Sept. 2006. ISBN: 978-0-262-03358-9.

[59] Xiangrong Chen and A.L. Yuille. "Detecting and Reading Text in Natural Scenes". In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Volume 2. June 2004, pages II–II. DOI: 10.1109/CVPR.2004.1315187.

[60] Zhanzhan Cheng, Yangliu Xu, Fan Bai, Yi Niu, Shiliang Pu, and Shuigeng Zhou. "AON: Towards Arbitrarily-Oriented Text Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pages 5571–5579. DOI: 10.1109/CVPR.2018.00584.

[61] Chee Kheng Chng and Chee Seng Chan. "Total-Text: A Comprehensive Dataset for Scene Text Detection and Recognition". In: *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*. IEEE, 2017, pages 935–942. DOI: 10.1109/ICDAR.2017.157.

[62] N. D. Cilia, C. De Stefano, F. Fontanella, C. Marrocco, M. Molinara, and A. Scotto Di Freca. "An End-to-End Deep Learning System for Medieval Writer Identification". In: *Pattern Recognition Letters* 129 (Jan. 2020), pages 137–143. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2019.11.025.

[63] Christian Clausner, Apostolos Antonacopoulos, Tom Derrick, and Stefan Pletschacher. "ICDAR2019 Competition on Recognition of Early Indian Printed Documents – REID2019". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 1527–1532. DOI: 10.1109/ICDAR.2019.00246.

[64] Stéphane Clinchant, Hervé Déjean, Jean-Luc Meunier, Eva Maria Lang, and Florian Kleber. "Comparing Machine Learning Approaches for Table Recognition in Historical Register Books". In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. Apr. 2018, pages 133–138. DOI: 10.1109/DAS.2018.44.

[65] Edo Collins, Raja Bala, Bob Price, and Sabine Susstrunk. "Editing in Style: Uncovering the Local Semantics of GANs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pages 5771–5780. DOI: 10.1109/CVPR42600.2020.00581.

[66] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Machine Learning* 20.3 (Sept. 1995), pages 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[67] Adrian Cosma, Mihai Ghidoveanu, Michael Panaitescu-Liess, and Marius Popescu. "Self-Supervised Representation Learning on Document Images". In: *Document Analysis Systems*. Edited by Xiang Bai, Dimosthenis Karatzas, and Daniel Lopresti. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pages 103–117. ISBN: 978-3-030-57058-3. DOI: 10.1007/978-3-030-57058-3_8.

[68] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. "Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning". In: *IEEE Communications Magazine* 27.11 (Nov. 1989), pages 41–46. ISSN: 0163-6804. DOI: 10.1109/35.41400.

[69] G. Cybenko. "Approximation by Superpositions of a Sigmoidal Function". In: *Mathematics of Control, Signals and Systems* 2.4 (Dec. 1989), pages 303–314. ISSN: 0932-4194, 1435-568X. DOI: 10.1007/BF02551274.

[70] E. E. Fournier d'Albe. "On a Type-Reading Optophone". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 90.619 (July 1914), pages 373–375. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.1914.0061.

[71] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. June 2009, pages 248–255. DOI: 10.1109/CVPR.2009.5206848.

[72] Patrick Doetsch, Michal Kozielski, and Hermann Ney. "Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition". In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. Sept. 2014, pages 279–284. DOI: 10.1109/ICFHR.2014.54.

[73] Patrick Doetsch, Albert Zeyer, and Hermann Ney. "Bidirectional Decoder Networks for Attention-Based End-to-End Offline Handwriting Recognition". In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Oct. 2016, pages 361–366. DOI: 10.1109/ICFHR.2016.0074.

[74] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. "A Learned Representation for Artistic Style". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[75] B. Epshtein, E. Ofek, and Y. Wexler. "Detecting Text in Natural Scenes with Stroke Width Transform". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference On*. June 2010, pages 2963–2970. DOI: 10.1109/CVPR.2010.5540041.

[76] Michael Fink, Thomas Layer, Georg Mackenbrock, and Michael Sprinzl. "Baseline Detection in Historical Documents Using Convolutional U-Nets". In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. Apr. 2018, pages 37–42. DOI: 10.1109/DAS.2018.34.

[77] Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. "Transcription Alignment of Latin Manuscripts Using Hidden Markov Models". In: *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*. HIP '11. New York, NY, USA: Association for Computing Machinery, Sept. 2011, pages 29–36. ISBN: 978-1-4503-0916-5. DOI: 10.1145/2037342.2037348.

[78] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. "Lexicon-Free Handwritten Word Spotting Using Character HMMs". In: *Pattern Recognition Letters*. Special Issue on Awards from ICPR 2010 33.7 (May 2012), pages 934–942. ISSN: 0167-8655. DOI: 10.1016/j.patrec.2011.09.009.

[79] Andreas Fischer, Muriel Visani, Van Cuong Kieu, and Ching Y. Suen. "Generation of Learning Samples for Historical Handwriting Recognition Using Image Degradation". In: *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*. HIP '13. New York, NY, USA: Association for Computing Machinery, Aug. 2013, pages 73–79. ISBN: 978-1-4503-2115-0. DOI: 10.1145/2501115.2501123.

[80] G.E. Hinton and T.J. Sejnowski. "Learning and Relearning in Boltzmann Machines". In: *Parallel distributed processing: Explorations in the microstructure of cognition* 1 (1968), pages 282–317.

[81] Rinon Gal, Dana Cohen, Amit Bermano, and Daniel Cohen-Or. "SWAGAN: A Style-Based Wavelet-Driven Generative Model". In: *arXiv:2102.06108 [cs, eess]* (Feb. 2021). arXiv: 2102.06108.

[82] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. "ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy

and Robustness". In: *International Conference on Learning Representations*. Sept. 2018.

[83] Ian Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. "Multi-Digit Number Recognition from Street View Imagery Using Deep Convolutional Neural Networks". In: *ICLR2014*. 2014.

[84] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pages 2672–2680.

[85] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. "Maxout Networks". In: *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. Volume 28. JMLR Workshop and Conference Proceedings. JMLR.org, 2013, pages 1319–1327.

[86] A. Graves, N. Jaitly, and A. r Mohamed. "Hybrid Speech Recognition with Deep Bidirectional LSTM". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Dec. 2013, pages 273–278. DOI: 10.1109/ASRU.2013.6707742.

[87] Emmanuèle Grosicki and Haikal El Abed. "ICDAR 2009 Handwriting Recognition Competition". In: *2009 10th International Conference on Document Analysis and Recognition*. July 2009, pages 1398–1402. DOI: 10.1109/ICDAR.2009.184.

[88] T. Grüning, R. Labahn, M. Diem, F. Kleber, and S. Fiel. "READ-BAD: A New Dataset and Evaluation Scheme for Baseline Detection in Archival Documents". In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. Apr. 2018, pages 351–356. DOI: 10.1109/DAS.2018.38.

[89] Tobias Grüning, Gundram Leifert, Tobias Strauß, Johannes Michael, and Roger Labahn. "A Two-Stage Method for Text Line Detection in Historical Documents". In: *International Journal on Document Analysis and Recognition (IJDAR)* 22.3 (Sept. 2019), pages 285–302. ISSN: 1433-2825. DOI: 10.1007/s10032-019-00332-1.

[90] Ankush Gupta, Andrea Vedaldi, and Andrew Zisserman. "Synthetic Data for Text Localisation in Natural Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 2315–2324. DOI: 10.1109/CVPR.2016.254.

[91] Anmol Hamid, Maryam Bibi, Momina Moetesum, and Imran Siddiqi. "Deep Learning Based Approach for Historical Manuscript Dating". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 967–972. DOI: 10.1109/ICDAR.2019.00159.

[92] Anmol Hamid, Maryam Bibi, Imran Siddiqi, and Momina Moetesum. "Historical Manuscript Dating Using Textural Measures". In: *2018 International Conference on Frontiers of Information Technology (FIT)*. Dec. 2018, pages 235–240. DOI: 10.1109/FIT.2018.00048.

[93] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pages 2961–2969. DOI: 10.1109/ICCV.2017.322.

[94] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 770–778. DOI: 10.1109/CVPR.2016.90.

[95] Pan He, Weilin Huang, Yu Qiao, Chen Change Loy, and Xiaoou Tang. "Reading Scene Text in Deep Convolutional Sequences". In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 2016, pages 3501–3508.

[96] Sheng He, Petros Samara, Jan Burgers, and Lambert Schomaker. "A Multiple-Label Guided Clustering Algorithm for Historical Document Dating and Localization". In: *IEEE Transactions on Image Processing* 25.11 (Nov. 2016), pages 5252–5265. ISSN: 1941-0042. DOI: 10.1109/TIP.2016.2602078.

[97] Sheng He, Petros Sammara, Jan Burgers, and Lambert Schomaker. "Towards Style-Based Dating of Historical Documents". In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. Sept. 2014, pages 265–270. DOI: 10.1109/ICFHR.2014.52.

[98] Geoffrey E. Hinton and Terrence Joseph Sejnowski. *Unsupervised Learning: Foundations of Neural Computation*. MIT Press, 1999. ISBN: 978-0-262-58168-4.

[99] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pages 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[100] Sepp Hochreiter. "Untersuchungen Zu Dynamischen Neuronalen Netzen". PhD thesis. 1991.

[101] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Networks* 2.5 (1989), pages 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

[102] H. Hotelling. "Analysis of a Complex of Statistical Variables into Principal Components". In: *Journal of Educational Psychology* 24.6 (1933), pages 417–441. ISSN: 1939-2176(Electronic),0022-0663(Print). DOI: 10.1037/h0071325.

[103] Jiabo Huang, Shaogang Gong, and Xiatian Zhu. "Deep Semantic Clustering by Partition Confidence Maximisation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pages 8849–8858. DOI: 10.1109/CVPR42600.2020.00887.

[104] Xun Huang and Serge Belongie. "Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pages 1501–1510. DOI: 10.1109/ICCV.2017.167.

[105] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. "Multimodal Unsupervised Image-to-Image Translation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pages 172–189. DOI: 10.1007/978-3-030-01219-9_11.

[106] The Wildenstein Plattner Institute Inc. *Paul Ferdinand Gachet and Paul Louis Gachet Papers*. 1816.

[107] The Wildenstein Plattner Institute Inc. *Stock Books and Inventories, Ambroise Vollard Records*. 1899.

[108] R. Reeve Ingle, Yasuhisa Fujii, Thomas Deselaers, Jonathan Baccash, and Ashok C. Popat. "A Scalable Handwritten Text Recognition System". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 17–24. DOI: 10.1109/ICDAR.2019.00013.

[109] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pages 448–456.

[110] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pages 5967–5976. DOI: 10.1109/CVPR.2017.632.

[111] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. "Reading Text in the Wild with Convolutional Neural Networks". In: *International Journal of Computer Vision* 116.1 (Jan. 2016), pages 1–20. DOI: 10.1007/s11263-015-0823-z.

[112] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition". In: *Workshop on Deep Learning, NIPS*. 2014.

[113] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pages 2017–2025.

[114] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. "Deep Features for Text Spotting". In: *Computer Vision – ECCV 2014*. Lecture Notes in Computer Science 8692. Springer International Publishing, Sept. 2014, pages 512–528. ISBN: 978-3-319-10592-5 978-3-319-10593-2. DOI: 10.1007/978-3-319-10593-2_34.

[115] Nitisha Jain. "Domain-Specific Knowledge Graph Construction for Semantic Analysis". In: *The Semantic Web: ESWC 2020 Satellite Events*. Edited by Andreas Harth, Valentina Presutti, Raphaël Troncy, Maribel Acosta, Axel Polleres, Javier D. Fernández, Josiane Xavier Parreira, Olaf Hartig, Katja Hose, and Michael Cochez. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pages 250–260. ISBN: 978-3-030-62327-2. DOI: 10.1007/978-3-030-62327-2_40.

[116] Nitisha Jain and Ralf Krestel. "Who Is Mona L.? Identifying Mentions of Artworks in Historical Archives". In: *Digital Libraries for Open Knowledge*. Edited by Antoine Doucet, Antoine Isaac, Koraljka Golub, Trond Aalberg, and Adam Jatowt. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pages 115–122. ISBN: 978-3-030-30760-8. DOI: 10.1007/978-3-030-30760-8_10.

[117] Xu Ji, Joao F. Henriques, and Andrea Vedaldi. "Invariant Information Clustering for Unsupervised Image Classification and Segmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pages 9865–9874. DOI: 10.1109/ICCV.2019.00996.

[118] Fan Jiang, Zhihui Hao, and Xinran Liu. "Deep Scene Text Detection with Connected Component Proposals". In: *arXiv:1708.05133 [cs]* (Aug. 2017). arXiv: 1708.05133 [cs].

[119] Simon Jordan, Mathias Seuret, Pavel Král, Ladislav Lenc, Jiří Martínek, Barbara Wiermann, Tobias Schwinger, Andreas Maier, and Vincent Christlein. "Re-Ranking for Writer Identification and Writer Retrieval".

In: *Document Analysis Systems*. Edited by Xiang Bai, Dimosthenis Karatzas, and Daniel Lopresti. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pages 572–586. ISBN: 978-3-030-57058-3. DOI: 10.1007/978-3-030-57058-3_40.

[120] Nicholas Journet, Muriel Visani, Boris Mansencal, Kieu Van-Cuong, and Antoine Billy. "DocCreator: A New Software for Creating Synthetic Ground-Truthed Document Images". In: *Journal of Imaging* 3.4 (Dec. 2017), page 62. DOI: 10.3390/jimaging3040062.

[121] 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 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIII*. Edited by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Volume 12368. Lecture Notes in Computer Science. Springer, 2020, pages 273–289. DOI: 10.1007/978-3-030-58592-1_17.

[122] Lei Kang, J. Ignacio Toledo, Pau Riba, Mauricio Villegas, Alicia Fornés, and Marçal Rusiñol. "Convolve, Attend and Spell: An Attention-Based Sequence-to-Sequence Model for Handwritten Word Recognition". In: *Pattern Recognition*. Edited by Thomas Brox, Andrés Bruhn, and Mario Fritz. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pages 459–472. ISBN: 978-3-030-12939-2. DOI: 10.1007/978-3-030-12939-2_32.

[123] Seokjun Kang, Brian Kenji Iwana, and Seiichi Uchida. "Cascading Modular U-Nets for Document Image Binarization". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 675–680. DOI: 10.1109/ICDAR.2019.00113.

[124] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. "ICDAR 2015 Competition on Robust Reading". In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. Aug. 2015, pages 1156–1160. DOI: 10.1109/ICDAR.2015.7333942.

[125] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, Lluis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere de las Heras. "ICDAR 2013 Robust Reading Competition". In: *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pages 1484–1493. DOI: 10.1109/ICDAR.2013.221.

[126] Romain Karpinski and Abdel Belaïd. "Semi-Synthetic Data Augmentation of Scanned Historical Documents". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 268–273. DOI: 10.1109/ICDAR.2019.00051.

[127] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive Growing of GANs for Improved Quality, Stability, and Variation". In: *International Conference on Learning Representations*. 2018.

[128] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. "Training Generative Adversarial Networks with Limited Data". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual*. Edited by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. 2020.

[129] Tero Karras, Samuli Laine, and Timo Aila. "A Style-Based Generator Architecture for Generative Adversarial Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pages 4401–4410. DOI: 10.1109/CVPR.2019.00453.

[130] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. "Analyzing and Improving the Image Quality of StyleGAN". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pages 8110–8119. DOI: 10.1109/CVPR42600.2020.00813.

[131] B. Kiessling. "A Modular Region and Text Line Layout Analysis System". In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Sept. 2020, pages 313–318. DOI: 10.1109/ICFHR2020.2020.00064.

[132] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Edited by Yoshua Bengio and Yann LeCun. 2015.

[133] John F. Kolen and Stefan C. Kremer. "Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies". In: *A Field Guide to Dynamical Recurrent Networks*. IEEE, 2001, pages 237–243. ISBN: 978-0-470-54403-7. DOI: 10.1109/9780470544037.ch14.

[134] Amandus Krantz and Florian Westphal. "Cluster-Based Sample Selection for Document Image Binarization". In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Volume 5. Sept. 2019, pages 47–52. DOI: 10.1109/ICDARW.2019.40080.

[135] Praveen Krishnan and C. V. Jawahar. "Generating Synthetic Data for Text Recognition". In: *arXiv:1608.04224 [cs]* (Aug. 2016). arXiv: 1608.04224 [cs].

[136] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pages 1097–1105.

[137] Anh Duc Le, Daichi Mochihashi, Katsuya Masuda, Hideki Mima, and Nam Tuan Ly. "Recognition of Japanese Historical Text Lines by an Attention-Based Encoder-Decoder and Text Line Generation". In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. HIP '19. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pages 37–41. ISBN: 978-1-4503-7668-6. DOI: 10.1145/3352631.3352641.

[138] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the Loss Landscape of Neural Nets". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pages 6391–6401.

[139] Hui Li, Peng Wang, and Chunhua Shen. "Towards End-To-End Text Spotting With Convolutional Recurrent Neural Networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pages 5238–5246. DOI: 10.1109/ICCV.2017.560.

[140] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. "Show, Attend and Read: A Simple and Strong Baseline for Irregular Text Recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pages 8610–8617. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33018610.

[141] Minghui Liao, Pengyuan Lyu, Minghang He, Cong Yao, Wenhao Wu, and Xiang Bai. "Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019), pages 1–1. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2019.2937086.

[142] Minghui Liao, Baoguang Shi, and Xiang Bai. "TextBoxes++: A Single-Shot Oriented Scene Text Detector". In: *IEEE Transactions on Image Processing* 27.8 (2018), pages 3676–3690. DOI: 10.1109/TIP.2018.2825107.

[143] Minghui Liao, Jian Zhang, Zhaoyi Wan, Fengming Xie, Jiajun Liang, Pengyuan Lyu, Cong Yao, and Xiang Bai. "Scene Text Recognition from Two-Dimensional Perspective". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (July 2019), pages 8714–8721. ISSN: 2374-3468. DOI: 10.1609/aaai.v33i01.33018714.

[144] Minghui Liao, Zhen Zhu, Baoguang Shi, Gui-Song Xia, and Xiang Bai. "Rotation-Sensitive Regression for Oriented Scene Text Detection". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pages 5909–5918. DOI: 10.1109/CVPR.2018.00619.

[145] 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*. Edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pages 740–755. ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_48.

[146] Tsungnan Lin, B. G. Horne, P. Tino, and C. L. Giles. "Learning Long-Term Dependencies in NARX Recurrent Neural Networks". In: *IEEE Transactions on Neural Networks* 7.6 (Nov. 1996), pages 1329–1338. ISSN: 1045-9227. DOI: 10.1109/72.548162.

[147] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. "On the Variance of the Adaptive Learning Rate and Beyond". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[148] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. "Path Aggregation Network for Instance Segmentation". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pages 8759–8768. DOI: 10.1109/CVPR.2018.00913.

[149] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Springer, Cham, Oct. 2016, pages 21–37. DOI: 10.1007/978-3-319-46448-0_2.

[150] Wei Liu, Chaofeng Chen, and Kwan-Yee K. Wong. "Char-Net: A Character-Aware Neural Network for Distorted Scene Text Recognition". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. Apr. 2018.

[151] Xuebo Liu, Ding Liang, Shi Yan, Dagui Chen, Yu Qiao, and Junjie Yan. "FOTS: Fast Oriented Text Spotting with a Unified Network". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pages 5676–5685. DOI: 10.1109/CVPR.2018.00595.

[152] Yuliang Liu and Lianwen Jin. "Deep Matching Prior Network: Toward Tighter Multi-Oriented Text Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pages 3454–3461. DOI: 10.1109/CVPR.2017.368.

[153] Yuliang Liu and Lianwen Jin. "Deep Matching Prior Network: Toward Tighter Multi-Oriented Text Detection". In: *arXiv:1703.01425 [cs]* (Mar. 2017). arXiv: 1703.01425 [cs].

[154] M. Liwicki and H. 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)*. Aug. 2005, 956–961 Vol. 2. DOI: 10.1109/ICDAR.2005.132.

[155] Marcus Liwicki and Horst Bunke. "HMM-Based On-Line Recognition of Handwritten Whiteboard Notes". In: *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, Oct. 2006.

[156] Marcus Liwicki, Horst Bunke, James A. Pittman, and Stefan Knerr. "Combining Diverse Systems for Handwritten Text Line Recognition". In: *Machine Vision and Applications* 22.1 (Jan. 2011), pages 39–51. ISSN: 1432-1769. DOI: 10.1007/s00138-009-0208-9.

[157] S. Lloyd. "Least Squares Quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (Mar. 1982), pages 129–137. ISSN: 1557-9654. DOI: 10.1109/TIT.1982.1056489.

[158] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[159] D. G. Lowe. "Object Recognition from Local Scale-Invariant Features". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Volume 2. Sept. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[160] Simon M. Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, Robert Young, Kazuki Ashida, Hiroki Nagai, Masayuki Okamoto, Hiroaki Yamamoto, Hidetoshi Miyao, JunMin Zhu, WuWen Ou, Christian Wolf, Jean-Michel Jolion, Leon Todoran, Marcel Worring,

and Xiaofan Lin. "ICDAR 2003 Robust Reading Competitions: Entries, Results, and Future Directions". In: *International Journal of Document Analysis and Recognition (IJDAR)* 7.2-3 (July 2005), pages 105–122. ISSN: 1433-2833, 1433-2825. DOI: 10.1007/s10032-004-0134-3.

[161] Canjie Luo, Lianwen Jin, and Zenghui Sun. "MORAN: A Multi-Object Rectified Attention Network for Scene Text Recognition". In: *Pattern Recognition* 90 (June 2019), pages 109–118. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2019.01.020.

[162] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. "Arbitrary-Oriented Scene Text Detection via Rotation Proposals". In: *IEEE Trans. Multim.* 20.11 (2018), pages 3111–3122. DOI: 10.1109/TMM.2018.2818020.

[163] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Citeseer, 2013.

[164] Ranju Mandal, Partha Pratim Roy, Umapada Pal, and Michael Blumenstein. "Multi-Lingual Date Field Extraction for Automatic Document Retrieval by Machine". In: *Information Sciences* 314 (Sept. 2015), pages 277–292. ISSN: 0020-0255. DOI: 10.1016/j.ins.2014.08.037.

[165] U.-V. Marti and H. Bunke. "A Full English Sentence Database for Off-Line Handwriting Recognition". In: *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*. Sept. 1999, pages 705–708. DOI: 10.1109/ICDAR.1999.791885.

[166] U.-V. Marti and H. Bunke. "Handwritten Sentence Recognition". In: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Volume 3. Sept. 2000, 463–466 vol.3. DOI: 10.1109/ICPR.2000.903584.

[167] U.-V. Marti and H. Bunke. "The IAM-Database: An English Sentence Database for Offline Handwriting Recognition". In: *International Journal on Document Analysis and Recognition* 5.1 (Nov. 2002), pages 39–46. ISSN: 1433-2833. DOI: 10.1007/s100320200071.

[168] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pages 115–133. ISSN: 0007-4985, 1522-9602. DOI: 10.1007/BF02478259.

[169] Aleksei Melnikov and Ivan Zagaynov. "Fast and Lightweight Text Line Detection on Historical Documents". In: *Document Analysis Systems*. Edited by Xiang Bai, Dimosthenis Karatzas, and Daniel Lopresti. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pages 441–450. ISBN: 978-3-030-57058-3. DOI: 10.1007/978-3-030-57058-3_31.

[170] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. Edited by Yoshua Bengio and Yann LeCun. 2013.

[171] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *arXiv:1411.1784 [cs, stat]* (Nov. 2014). arXiv: 1411.1784 [cs, stat].

[172] Anand Mishra, Karteek Alahari, and Cv Jawahar. "Scene Text Recognition Using Higher Order Language Priors". In: *BMVC 2012-23rd British Machine Vision Conference*. British Machine Vision Association, 2012, pages 127.1–127.11. ISBN: 1-901725-46-4. DOI: 10.5244/C.26.127.

[173] Ranjan Mondal, Deepayan Chakraborty, and Bhabatosh Chanda. "Learning 2D Morphological Network for Old Document Image Binarization". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 65–70. DOI: 10.1109/ICDAR.2019.00020.

[174] Richard S. Morgan. "Optical Readers: 1970". In: *Computers and the Humanities* 5.2 (Nov. 1970), pages 75–78. ISSN: 0010-4817, 1572-8412. DOI: 10.1007/BF02402284.

[175] Guenter Muehlberger, Louise Seaward, Melissa Terras, Sofia Ares Oliveira, Vicente Bosch, Maximilian Bryan, Sebastian Colutto, Hervé Déjean, Markus Diem, Stefan Fiel, Basilis Gatos, Albert Greinoecker, Tobias Grüning, Guenter Hackl, Vili Haukkovaara, Gerhard Heyer, Lauri Hirvonen, Tobias Hodel, Matti Jokinen, Philip Kahle, Mario Kallio, Frederic Kaplan, Florian Kleber, Roger Labahn, Eva Maria Lang, Sören Laube, Gundram Leifert, Georgios Louloudis, Rory McNicholl, Jean-Luc Meunier, Johannes Michael, Elena Mühlbauer, Nathanael Philipp, Ioannis Pratikakis, Joan Puigcerver Pérez, Hannelore Putz, George Retsinas, Verónica Romero, Robert Sablatnig, Joan Andreu Sánchez, Philip Schofield, Giorgos Sfikas, Christian Sieber, Nikolaos Stamatopoulos, Tobias Strauß, Tamara Terbul, Alejandro Héctor Toselli, Berthold Ulreich, Mauricio Villegas, Enrique Vidal, Johanna Walcher, Max Weidemann, Herbert Wurster, and Konstantinos Zagoris. "Transforming Scholarship in the Archives through Handwritten Text Recog-

nition: Transkribus as a Case Study". In: *Journal of Documentation* 75.5 (Jan. 2019), pages 954–976. ISSN: 0022-0418. DOI: 10.1108/JD-07-2018-0114.

[176] Vinod Nair and Geoffrey E Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010, pages 807–814.

[177] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. "Reading Digits in Natural Images with Unsupervised Feature Learning". In: *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. Volume 2011. 2011, page 5.

[178] L. Neumann and J. Matas. "Real-Time Scene Text Localization and Recognition". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2012, pages 3538–3545. DOI: 10.1109/CVPR.2012.6248097.

[179] Lukas Neumann and Jiri Matas. "A Method for Text Localization and Recognition in Real-World Images". In: *Computer Vision – ACCV 2010*. Lecture Notes in Computer Science 6494. Springer Berlin Heidelberg, Nov. 2010, pages 770–783. ISBN: 978-3-642-19317-0 978-3-642-19318-7. DOI: 10.1007/978-3-642-19318-7_60.

[180] Weili Nie, Tero Karras, Animesh Garg, Shoubhik Debnath, Anjul Patney, Ankit Patel, and Animashree Anandkumar. "Semi-Supervised StyleGAN for Disentanglement Learning". In: *International Conference on Machine Learning*. PMLR, Nov. 2020, pages 7360–7369.

[181] Mehdi Noroozi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *Computer Vision – ECCV 2016*. Edited by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pages 69–84. ISBN: 978-3-319-46466-4. DOI: 10.1007/978-3-319-46466-4_5.

[182] S. Ares Oliveira, B. Seguin, and F. Kaplan. "dhSegment: A Generic Deep-Learning Approach for Document Segmentation". In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Aug. 2018, pages 7–12. DOI: 10.1109/ICFHR-2018.2018.00011.

[183] Nobuyuki Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE transactions on systems, man, and cybernetics* 9.1 (1979), pages 62–66.

[184] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019, pages 8026–8037.

[185] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pages 1532–1543. DOI: 10.3115/v1/D14-1162.

[186] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pages 1532–1543.

[187] James Philips and Nasseh Tabrizi. "Historical Document Processing: A Survey of Techniques, Tools, and Trends". In: *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2020, Volume 1: KDIR, Budapest, Hungary, November 2-4, 2020*. Edited by Ana L. N. Fred and Joaquim Filipe. SCITEPRESS, 2020, pages 341–349. DOI: 10.5220/0010177403410349.

[188] Stanislav Pidhorskyi, Donald A. Adjeroh, and Gianfranco Doretto. "Adversarial Latent Autoencoders". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pages 14104–14113. DOI: 10.1109/CVPR42600.2020.01411.

[189] Vinaychandran Pondenkandath, Michele Alberti, Michaël Diatta, Rolf Ingold, and Marcus Liwicki. "Historical Document Synthesis with Generative Adversarial Networks". In: *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Volume 5. Sept. 2019, pages 146–151. DOI: 10.1109/ICDARW.2019.40096.

[190] J. Puigcerver. "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?" In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Volume 01. Nov. 2017, pages 67–72. DOI: 10.1109/ICDAR.2017.20.

[191] Trung Quy Phan, Palaiahnakote Shivakumara, Shangxuan Tian, and Chew Lim Tan. "Recognizing Text with Perspective Distortion in Natural Scenes". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pages 569–576. DOI: 10.1109/ICCV.2013.76.

[192] Hannes Rantzsch, Haojin Yang, and Christoph Meinel. "Signature Embedding: Writer Independent Offline Signature Verification with Deep Metric Learning". In: *Advances in Visual Computing*. Edited by George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, and Tobias Isenberg. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pages 616–625. ISBN: 978-3-319-50832-0. DOI: 10.1007/978-3-319-50832-0_60.

[193] Hendrik Rätz. "Handwriting Classification on Archival Documents Using Deep Neural Networks". Master's thesis. Potsdam: University of Potsdam, Oct. 2020.

[194] 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*. 2016, pages 779–788. DOI: 10.1109/CVPR.2016.91.

[195] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems 28*. Edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pages 91–99.

[196] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. "A Robust Arbitrary Text Detection System for Natural Scene Images". In: *Expert Systems with Applications* 41.18 (Dec. 2014), pages 8027–8048. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2014.07.008.

[197] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*. Edited by Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi. Volume 9351. Lecture Notes in Computer Science. Springer, 2015, pages 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[198] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning Representations by Back-Propagating Errors". In: *Nature* 323.6088 (Oct. 1986), pages 533–536. DOI: 10.1038/323533a0.

[199] Eugen Rusakov, Kai Brandenbusch, Denis Fisseler, Turna Somel, Gernot A. Fink, Frank Weichert, and Gerfrid G. W. Müller. "Generating Cuneiform Signs with Cycle-Consistent Adversarial Networks". In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. HIP '19. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pages 19–24. ISBN: 978-1-4503-7668-6. DOI: 10.1145/3352631.3352632.

[200] Ruslan Salakhutdinov and Geoffrey Hinton. "Deep Boltzmann Machines". In: *Artificial Intelligence and Statistics*. PMLR, Apr. 2009, pages 448–455.

[201] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved Techniques for Training GANs". In: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., Dec. 2016, pages 2234–2242. ISBN: 978-1-5108-3881-9.

[202] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Mądry. "How Does Batch Normalization Help Optimization?" In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS'18. Red Hook, NY, USA: Curran Associates Inc., Dec. 2018, pages 2488–2498.

[203] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pages 815–823. DOI: 10.1109/CVPR.2015.7298682.

[204] Anna Scius-Bertrand, Lars Voegtlin, Michele Alberti, Andreas Fischer, and Marc Bui. "Layout Analysis and Text Column Segmentation for Historical Vietnamese Steles". In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. HIP '19. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pages 84–89. ISBN: 978-1-4503-7668-6. DOI: 10.1145/3352631.3352634.

[205] Burr Settles. *Active Learning Literature Survey*. Technical Report. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[206] Baoguang Shi, Xiang Bai, and Serge J. Belongie. "Detecting Oriented Text in Natural Images by Linking Segments". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI,*

USA, July 21-26, 2017. IEEE Computer Society, 2017, pages 3482–3490. DOI: 10.1109/CVPR.2017.371.

[207] Baoguang Shi, Xiang Bai, and Cong Yao. "An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016). DOI: 10.1109/TPAMI.2016.2646371.

[208] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. "Robust Scene Text Recognition With Automatic Rectification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 4168–4176. DOI: 10.1109/CVPR.2016.452.

[209] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. "ASTER: An Attentional Scene Text Recognizer with Flexible Rectification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.9 (Sept. 2019), pages 2035–2048. ISSN: 0162-8828, 2160-9292, 1939-3539. DOI: 10.1109/TPAMI.2018.2848939.

[210] Wenzhe Shi, Jose Caballero, Ferenc Huszar, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 1874–1883. DOI: 10.1109/CVPR.2016.207.

[211] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Advances in Neural Information Processing Systems 28*. Edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pages 802–810.

[212] Foteini Simistira, Mathias Seuret, Nicole Eichenberger, Angelika Garz, Marcus Liwicki, and Rolf Ingold. "DIVA-HisDB: A Precisely Annotated Large Dataset of Challenging Medieval Manuscripts". In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. Oct. 2016, pages 471–476. DOI: 10.1109/ICFHR.2016.0093.

[213] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*. 2015.

[214] R. Smith. "An Overview of the Tesseract OCR Engine". In: *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*. IEEE Computer Society, 2007, pages 629–633. DOI: 10.1109/ICDAR.2007.4376991.

[215] Raymond Smith, Chunhui Gu, Dar-Shyang Lee, Huiyi Hu, Ranjith Unnikrishnan, Julian Ibarz, Sacha Arnoud, and Sophia Lin. "End-to-End Interpretation of the French Street Name Signs Dataset". In: *Computer Vision – ECCV 2016 Workshops*. Springer, Cham, Oct. 2016, pages 411–426. DOI: 10.1007/978-3-319-46604-0_30.

[216] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of machine learning research* 15.1 (2014), pages 1929–1958.

[217] "Statistical Machine". 1838389. Dec. 1931.

[218] Seth Stewart and Bill Barrett. "Document Image Page Segmentation and Character Recognition As Semantic Segmentation". In: *Proceedings of the 4th International Workshop on Historical Document Imaging and Processing*. HIP2017. New York, NY, USA: ACM, 2017, pages 101–106. ISBN: 978-1-4503-5390-8. DOI: 10.1145/3151509.3151518.

[219] Ilya Sutskever, Oriol Vinyals, and Quoc V. V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pages 3104–3112.

[220] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going Deeper with Convolutions". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. 2015, pages 1–9. DOI: 10.1109/CVPR.2015.7298594.

[221] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 2818–2826. DOI: 10.1109/CVPR.2016.308.

[222] Yiping Tang, Kohei Hatano, and Eiji Takimoto. "Recognition of Japanese Historical Hand-Written Characters Based on Object Detection Methods". In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. HIP '19. New York, NY, USA: Association for Computing Machinery, Sept. 2019, pages 72–77. ISBN: 978-1-4503-7668-6. DOI: 10.1145/3352631.3352642.

[223] Chris Tensmeyer, Mike Brodie, Daniel Saunders, and Tony Martinez. "Generating Realistic Binarization Data with Generative Adversarial Networks". In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. Sept. 2019, pages 172–177. DOI: 10.1109/ICDAR.2019.00036.

[224] Zhi Tian, Weilin Huang, Tong He, Pan He, and Yu Qiao. "Detecting Text in Natural Image with Connectionist Text Proposal Network". In: *Computer Vision – ECCV 2016*. Springer, Cham, Oct. 2016, pages 56–72. DOI: 10.1007/978-3-319-46484-8_4.

[225] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. "Chainer: A Next-Generation Open Source Framework for Deep Learning". In: *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-Ninth Annual Conference on Neural Information Processing Systems (NIPS)*. 2015.

[226] Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. "Designing an Encoder for StyleGAN Image Manipulation". In: *arXiv:2102.02766 [cs]* (Feb. 2021). arXiv: 2102.02766 [cs].

[227] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: *arXiv:1607.08022 [cs]* (Nov. 2017). arXiv: 1607.08022.

[228] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. "SCAN: Learning to Classify Images Without Labels". In: *Computer Vision – ECCV 2020*. Edited by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pages 268–285. ISBN: 978-3-030-58607-2. DOI: 10.1007/978-3-030-58607-2_16.

[229] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30*. Edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pages 5998–6008.

[230] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. "Show and Tell: A Neural Image Caption Generator". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pages 3156–3164. DOI: 10.1109/CVPR.2015.7298935.

[231] Lars Vögtlin, Manuel Drazyk, Vinaychandran Pondenkandath, Michele Alberti, and Rolf Ingold. "Generating Synthetic Handwritten Historical Documents With OCR Constrained GANs". In: *arXiv:2103.08236 [cs]* (May 2021). arXiv: 2103.08236 [cs].

[232] Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. "Image Captioning with Deep Bidirectional LSTMs". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pages 988–997. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2964299.

[233] Jianfeng Wang and Xiaolin Hu. "Convolutional Neural Networks with Gated Recurrent Connections". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), pages 1–1. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3054614.

[234] Kai Wang, B. Babenko, and S. Belongie. "End-to-End Scene Text Recognition". In: *2011 International Conference on Computer Vision*. Nov. 2011, pages 1457–1464. DOI: 10.1109/ICCV.2011.6126402.

[235] Kai Wang and Serge Belongie. "Word Spotting in the Wild". In: *Computer Vision – ECCV 2010*. Edited by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Lecture Notes in Computer Science 6311. Springer Berlin Heidelberg, Sept. 2010, pages 591–604. ISBN: 978-3-642-15548-2 978-3-642-15549-9. DOI: 10.1007/978-3-642-15549-9_43.

[236] Peng Wang, Lu Yang, Hui Li, Yuyan Deng, Chunhua Shen, and Yanning Zhang. "A Simple and Robust Convolutional-Attention Network for Irregular Text Recognition". In: *arXiv:1904.01375 [cs]* (Apr. 2019). arXiv: 1904.01375 [cs].

[237] Qingqing Wang, Ye Huang, Wenjing Jia, Xiangjian He, Michael Blumenstein, Shujing Lyu, and Yue Lu. "FACLSTM: ConvLSTM with Focused Attention for Scene Text Recognition". In: *Science China. Information Sciences* 63.2 (2020), page 120103. DOI: 10.1007/s11432-019-2713-1.

[238] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng. "End-to-End Text Recognition with Convolutional Neural Networks". In: *2012 21st International Conference on Pattern Recognition (ICPR)*. Nov. 2012, pages 3304–3308.

[239] Tao Wang, David J. Wu, Adam Coates, and Andrew Y. Ng. "End-to-End Text Recognition with Convolutional Neural Networks". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. Nov. 2012, pages 3304–3308.

[240] Xiaobing Wang, Yingying Jiang, Zhenbo Luo, Cheng-Lin Liu, Hyunsoo Choi, and Sungjin Kim. "Arbitrary Shape Scene Text Detection with Adaptive Text Region Representation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pages 6449–6458. DOI: 10.1109/CVPR.2019.00661.

[241] Curtis Wigington, Seth Stewart, Brian Davis, Bill Barrett, Brian Price, and Scott Cohen. "Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network". In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Volume 01. Nov. 2017, pages 639–645. DOI: 10.1109/ICDAR.2017.110.

[242] Curtis Wigington, Chris Tensmeyer, Brian Davis, William Barrett, Brian Price, and Scott Cohen. "Start, Follow, Read: End-to-End Full-Page Handwriting Recognition". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pages 367–383. DOI: 10.1007/978-3-030-01231-1_23.

[243] Zbigniew Wojna, Alex Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. "Attention-Based Extraction of Structured Information from Street View Imagery". In: *arXiv:1704.03549 [cs]* (Apr. 2017). arXiv: 1704.03549 [cs].

[244] Zbigniew Wojna, Alexander N. Gorban, Dar-Shyang Lee, Kevin Murphy, Qian Yu, Yeqing Li, and Julian Ibarz. "Attention-Based Extraction of Structured Information from Street View Imagery". In: *14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9-15, 2017*. IEEE, 2017, pages 844–850. DOI: 10.1109/ICDAR.2017.143.

[245] Yuxin Wu and Kaiming He. "Group Normalization". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIII*. 2018, pages 3–19. DOI: 10.1007/978-3-030-01261-8_1.

[246] Shanyu Xiao, Liangrui Peng, Ruijie Yan, and Shengjin Wang. "Deep Network with Pixel-Level Rectification and Robust Training for Handwriting Recognition". In: *SN Computer Science* 1.3 (Apr. 2020), page 145. ISSN: 2661-8907. DOI: 10.1007/s42979-020-00133-y.

[247] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. 2015, pages 2048–2057.

[248] Haojin Yang, Martin Fritzsche, Christian Bartz, and Christoph Meinel. "BMXNet: An Open-Source Binary Neural Network Implementation Based on MXNet". In: *Proceedings of the 25th ACM International Conference on Multimedia*. MM '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pages 1209–1212. ISBN: 978-1-4503-4906-2. DOI: 10.1145/3123266.3129393.

[249] Haojin Yang, Cheng Wang, Christian Bartz, and Christoph Meinel. "SceneTextReg: A Real-Time Video OCR System". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pages 698–700. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2973811.

[250] Cong Yao, Xiang Bai, Baoguang Shi, and Wenyu Liu. "Strokelets: A Learned Multi-Scale Representation for Scene Text Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pages 4042–4049. DOI: 10.1109/CVPR.2014.515.

[251] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv:1212.5701 [cs]* (Dec. 2012). arXiv: 1212.5701 [cs].

[252] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Lecture Notes in Computer Science 8689. Springer International Publishing, Sept. 2014, pages 818–833. ISBN: 978-3-319-10589-5 978-3-319-10590-1. DOI: 10.1007/978-3-319-10590-1_53.

[253] Fangneng Zhan and Shijian Lu. "ESIR: End-To-End Scene Text Recognition via Iterative Image Rectification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pages 2059–2068. DOI: 10.1109/CVPR.2019.00216.

[254] Richard Zhang, Phillip Isola, and Alexei A. Efros. "Colorful Image Colorization". In: *Computer Vision – ECCV 2016*. Edited by Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pages 649–666. ISBN: 978-3-319-46487-9. DOI: 10.1007/978-3-319-46487-9_40.

[255] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. "DatasetGAN: Efficient Labeled Data Factory With Minimal Human Effort". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pages 10145–10155.

[256] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. "EAST: An Efficient and Accurate Scene Text Detector". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pages 2642–2651. DOI: 10.1109/CVPR.2017.283.

[257] Zhi-Hua Zhou. "A Brief Introduction to Weakly Supervised Learning". In: *National Science Review* 5.1 (Jan. 2018), pages 44–53. ISSN: 2095-5138. DOI: 10.1093/nsr/nwx106.

[258] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pages 2242–2251. DOI: 10.1109/ICCV.2017.244.

[259] C. Lawrence Zitnick and Piotr Dollár. "Edge Boxes: Locating Object Proposals from Edges". In: *Computer Vision – ECCV 2014*. Edited by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pages 391–405. ISBN: 978-3-319-10602-1. DOI: 10.1007/978-3-319-10602-1_26.

Appendix

6 Collaborations with Students

This thesis would not have been possible without the support of a student writing his master thesis. Here, I will shortly provide information about the support of said student and my role while working together with him.

The work presented in section 4.4 was achieved with the support of Hendrik Rätz in his master thesis "Handwriting Classification on Archival Documents using Deep Neural Networks" [193]. Here, Mr. Rätz supported me by implementing my ideas for handwriting classification, discussion about problems and possible solutions, as well as by performing all experiments. The work on this thesis also led to the publication of [27] together with Mr. Rätz.

7 Publication List

Conferences

- Cheng Wang, Haojin Yang, Christian Bartz, and Christoph Meinel. "Image Captioning with Deep Bidirectional LSTMs". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pages 988–997. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2964299
- Haojin Yang, Cheng Wang, Christian Bartz, and Christoph Meinel. "SceneTextReg: A Real-Time Video OCR System". In: *Proceedings of the 24th ACM International Conference on Multimedia*. MM '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pages 698–700. ISBN: 978-1-4503-3603-1. DOI: 10.1145/2964284.2973811
- Christian Bartz, Tom Herold, Haojin Yang, and Christoph Meinel. "Language Identification Using Deep Convolutional Recurrent Neural Networks". In: *Neural Information Processing*. Edited by Derong Liu, Shengli Xie, Yuanqing Li, Dongbin Zhao, and El-Sayed M. El-Alfy. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pages 880–889. ISBN: 978-3-319-70136-3. DOI: 10.1007/978-3-319-70136-3_93
- Haojin Yang, Martin Fritzsche, Christian Bartz, and Christoph Meinel. "BMXNet: An Open-Source Binary Neural Network Implementation Based on MXNet". In: *Proceedings of the 25th ACM International Conference on Multimedia*. MM '17. New York, NY, USA: Association for Computing Machinery, Oct. 2017, pages 1209–1212. ISBN: 978-1-4503-4906-2. DOI: 10.1145/3123266.3129393
- Christian Bartz, Haojin Yang, and Christoph Meinel. "SEE: Towards Semi-Supervised End-to-End Scene Text Recognition". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. Apr. 2018
- Christian Bartz, Laurenz Seidel, Duy-Hung Nguyen, Joseph Bethge, Haojin Yang, and Christoph Meinel. "Synthetic Data for the Analysis of Archival Documents: Handwriting Determination". In: *2020 Digital Image Computing: Techniques and Applications (DICTA)*. Nov. 2020, pages 1–8. DOI: 10.1109/DICTA51227.2020.9363410
- Joseph Bethge, Christian Bartz, Haojin Yang, and Christoph Meinel. "BMXNet 2: An Open Source Framework for Low-Bit Networks - Reproducing, Understanding, Designing and Showcasing". In: *Proceedings of the 28th ACM Interna-*

tional Conference on Multimedia. MM '20. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pages 4469–4472. ISBN: 978-1-4503-7988-5. DOI: 10.1145/3394171.3414539

- Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. "MeliusNet: An Improved Network Architecture for Binary Neural Networks". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pages 1439–1448

Workshops

- Christian Bartz, Haojin Yang, Joseph Bethge, and Christoph Meinel. "LoANs: Weakly Supervised Object Detection with Localizer Assessor Networks". In: *Computer Vision – ACCV 2018 Workshops*. Edited by Gustavo Carneiro and Shaodi You. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019, pages 341–356. ISBN: 978-3-030-21074-8. DOI: 10.1007/978-3-030-21074-8_29
- Christian Bartz, Nitisha Jain, and Ralf Krestel. "Automatic Matching of Paintings and Descriptions in Art-Historic Archives Using Multimodal Analysis". In: *Proceedings of the 1st International Workshop on Artificial Intelligence for Historical Image Enrichment and Access*. Marseille, France: European Language Resources Association (ELRA), May 2020, pages 23–28. ISBN: 979-10-95546-63-4
- Christian Bartz, Nitisha Jain, Tobias Bredow, Emanuel Metzenthin, Jona Otholt, and Ralf Krestel. "Semantic Analysis of Cultural Heritage Data: Aligning Paintings and Descriptions in Art-Historic Collections". In: *Pattern Recognition. ICPR International Workshops and Challenges*. Edited by Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pages 517–530. ISBN: 978-3-030-68796-0. DOI: 10.1007/978-3-030-68796-0_37
- Christian Bartz, Hendrik Rätz, and Christoph Meinel. "Handwriting Classification for the Analysis of Art-Historical Documents". In: *Pattern Recognition. ICPR International Workshops and Challenges*. Edited by Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pages 562–576. ISBN: 978-3-030-68796-0. DOI: 10.1007/978-3-030-68796-0_40

Technical Reports

- Christian Bartz, Haojin Yang, and Christoph Meinel. "STN-OCR: A Single Neural Network for Text Detection and Text Recognition". In: *arXiv:1707.08831 [cs]* (July 2017). arXiv: 1707.08831 [cs]
- Joseph Bethge, Haojin Yang, Christian Bartz, and Christoph Meinel. "Learning to Train a Binary Neural Network". In: *arXiv:1809.10463 [cs, stat]* (Sept. 2018). arXiv: 1809.10463 [cs, stat]
- Christian Bartz, Joseph Bethge, Haojin Yang, and Christoph Meinel. "KISS: Keeping It Simple for Scene Text Recognition". In: *arXiv:1911. 08400 [cs]* (Nov. 2019). arXiv: 1911.08400
- Joseph Bethge, Christian Bartz, Haojin Yang, Ying Chen, and Christoph Meinel. "MeliusNet: Can Binary Neural Networks Achieve MobileNet-Level Accuracy?" In: *arXiv:2001.05936 [cs, stat]* (Mar. 2020). arXiv: 2001.05936 [cs, stat]
- Christian Bartz, Joseph Bethge, Haojin Yang, and Christoph Meinel. "One Model to Reconstruct Them All: A Novel Way to Use the Stochastic Noise in StyleGAN". In: *arXiv:2010.11113 [cs, eess]* (Oct. 2020). arXiv: 2010.11113 [cs, eess] - in submission
- Christian Bartz, Hendrik Rätz, Haojin Yang, Joseph Bethge, and Christoph Meinel. "Synthesis in Style: Semantic Segmentation of Historical Documents Using Synthetic Data". In: *arXiv:2107.06777 [cs]* (July 2021). arXiv: 2107.06777 [cs]

8 Curriculum Vitæ

Die Seiten 181-182 (Curriculum vitæ) enthalten persönliche Daten. Sie sind deshalb nicht Bestandteil der Veröffentlichung
The pages 181-182 (Curriculum vitæ) contain personal information. Thus, they are not part of the publication.

Die Seiten 181-182 (Curriculum vitæ) enthalten persönliche Daten. Sie sind deshalb nicht Bestandteil der Veröffentlichung
The pages 181-182 (Curriculum vitæ) contain personal information. Thus, they are not part of the publication.