SELF-TRAINING FOR HANDWRITTEN WORD RECOGNITION AND RETRIEVAL

Dissertation
zur Erlangung des Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

FABIAN WOLF

Dortmund

2024

Tag der mündlichen Prüfung:	7.11.2024
Dekan:	Prof. Dr. Jens Teubner
Gutachter:	Prof. Dr. Gernot A. Fink Prof. Dr. Alicia Fornés

ACKNOWLEDGMENTS

Despite being right at the beginning of this thesis, this chapter is the last chapter in a long journey. For traveling, people sometimes ask what is more important? The destination or the journey? The best answer I have heard so far was “the company”, which could not be more true in this case. Even though my name is on the front matter of this document, it is the work of many people who deserve all my gratitude. These people not only shaped this work, but their company over the last few years also shaped me as a person and I would not want to have missed a single memory that we have created. Countless evenings, traveling together, a pandemic, sharing our laughs and worries have made you more than colleagues.

I want to start my long list of “thank yous”, with Prof. Gernot A. Fink. Thank you, Gernot, for giving me the opportunity to pursue this title and for allowing me to be part of this fascinating world called research. During my time as a PhD student, I always felt your support and genuine interest in the path I was following. Thanks for your door always being open and creating an environment which allowed me to achieve my goals. While Gernot was part of this journey from the beginning, I want to thank Prof. Dr. Alicia Fornés for joining the final steps. Thanks, Alicia, for reviewing my thesis and joining the doctoral committee. Having a reviewer with such a positive and personal attitude and the commitment to join my defense in person meant a lot to me. I also want to thank Prof. Dr. Buchholz and Prof. Dr. Harmeling for participating in the committee.

This journey started when I stumbled in Fernando’s and Eugen’s office looking for a topic for my master’s thesis. You both have been there from day one and never would I have guessed that this coincidence would turn out to be one of the best things that has happened to me. Thank you, Fernando, for being the enthusiastic person you are and for being a great supervisor. Thanks for seeing more in me than just another student and showing me the excitement of the world that we have worked in. Thanks for having a couch for me in times when I needed one, and thanks for becoming a friend.

Thinking about the early days of this work, I want to thank Dr. Leonard Rothacker. Thanks for being a great mentor, teaching me how to navigate the world as a PhD student and accompanying me for my first steps.

I am incredibly thankful that I was part of this amazing group of people that call themselves colleagues but are so much more. This work would not exist without Eugen, Fernando, Philipp, Kai, Dominik, Arthur, Oliver, Nilah, office Tim and remote Tim. You made this time something very special. I always appreciated so much that we were able to share the good and the bad moments. We laughed together, we complained together, we shared our worries and picked each other up. No idea how many, but there are countless memories. The alphabet nights, pub quizzes, traveling in India with Kai, a romantic campfire trip with Fernando and Arthur, conferences and summer schools.... Seeing you all together, appreciating and enjoying each other’s company when we celebrated my defense is a moment I will never forget. Thank you all for allowing me to share this time with you.

This last paragraph is dedicated to my family. Thank you, Kara, for being a better partner than I could wish for. Last, I want to thank my parents. Thank you Mum and Dad. You made me the person I am today, supported me at every step in my life. Thank you for being so proud and on the other hand not caring at all. Thanks for the fact that I know that you will always be there for me, no matter what.

I love you.

CONTENTS

PUBLICATIONS

[WF24] F. Wolf and G. A. Fink. "Self-Training for Handwritten Word Recognition and Retrieval." In: *Int. Journal on Document Analysis and Recognition* 27.3 (2024), pp. 225–244.

[Tue+22] O. Tueselmann, F. Mueller, F. Wolf, and G. A. Fink. "Recognition-free Question Answering on Handwritten Document Collections." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Vol. 13639. Hyderabad, India, 2022, pp. 259–273.

[WF22a] F. Wolf and G. A. Fink. "Combining Self-Training and Minimal Annotations for Handwritten Word Recognition." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Vol. 13639. Hyderabad, India, 2022, pp. 300–315.

[WF22b] F. Wolf and G. A. Fink. "Self-Training of Handwritten Word Recognition for Synthetic-to-Real Adaptation." In: *Proc. of the Int. Conf. on Pattern Recognition*. Montreal, Canada: IEEE, 2022, pp. 3885–3892.

[RWF21] L. Rothacker, F. Wolf, and G. A. Fink. "Annotation-free Word Spotting with Bag-of-Features HMMs." In: *Int. Journal on Pattern Recognition and Artificial Intelligence* 35.4 (2021), 2153001:1–2153001:37.

[TWF21] O. Tueselmann, F. Wolf, and G. A. Fink. "Are End-to-End Systems Really Necessary for NER on Handwritten Document Images?" In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 12822. Lecture Notes in Computer Science. Lausanne, Switzerland, 2021, pp. 808–822.

[WFF21] F. Wolf, A. Fischer, and G. A. Fink. "Graph Convolutional Neural Networks for Learning Attribute Representations for Word Spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 12821. 2021, pp. 50–64.

[TWF20] O. Tueselmann, F. Wolf, and G. A. Fink. "Identifying and Tackling Key Challenges in Semantic Word Spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Dortmund, Germany (virtual), 2020, pp. 55–60.

[WBF20] F. Wolf, K. Brandenbusch, and G. A. Fink. "Improving Handwritten Word Synthesis for Annotation-free Word Spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Dortmund, Germany (virtual), 2020, pp. 61–66.

[WF20] F. Wolf and G. A. Fink. "Annotation-free Learning of Deep Representations for Word Spotting using Synthetic Data and Self Labeling." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vol. 12116. Winner of the Nakano Best Paper Award. Wuhan, China (virtual): Springer, 2020, pp. 293–308.

[WOF19] F. Wolf, P. Oberdiek, and G. A. Fink. “Exploring Confidence Measures for Word Spotting in Heterogeneous Datasets.” In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 583–588.

NOTATION

When using mathematical expressions in this thesis all variables, functions and arguments are defined where they appear first. The following notation is used for specific mathematical elements:

$a, b, c, \ldots$	scalar
$\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$	vector
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	matrix or tensor
$A, B, C, \ldots$	univariate random variable
$\mathsf{A}, \mathsf{B}, \mathsf{C}, \ldots$	integer value for, e.g., counts, cardinalities or dimensionalities
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	set
$\lvert\mathbf{A}\rvert$	the cardinality of set $\mathbf{A}$
$a^{(i)}$	the i-th element in set $\mathbf{S} = \left\{\mathbf{a}^{(i)}\right\}_{i=1}^{n}$
a_i	the i-th element of vector $\mathbf{a}$
${}^{x}a$	the vector representation corresponding to element x
$y, \hat{y}, \bar{y}$	the label, prediction, pseudo-label
$p, P(x), P(y\lvert x)$	probabilities
$\mathcal{X}, \mathcal{V}, \mathcal{H}$	vector space
$\mathcal{Y}$	the label space
$\mathbf{I}, \mathcal{I}$	an image, the image space
$f(x), f(\mathbf{x})$	scalar function with scalar or vector argument
$\mathcal{G}_e, \mathcal{G}_d$	an encoder or decoder function
$\mathcal{L}(\cdot)$	a loss function
$c(\cdot)$	a confidence estimate
$d(\cdot, \cdot)$	a distance function
$[\mathbf{a}, \mathbf{b}]$	the concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$
θ, Θ	a set of model parameters, the parameter space
$\mathbb{L}$	a lexicon containing N strings $\mathbb{L} = \{l_0, \ldots, l_N\}$

1 INTRODUCTION

For many centuries, handwritten documents have been an important medium to carry and preserve information. The use of handwriting in literature, in letters and in bureaucracy, resulted in the creation of a tremendous number of documents. Today, large scale document collections that are stored and preserved in archives and libraries exist. A significant part of all the data stored in these physical document collections is largely inaccessible. Without a digital copy of the documents, extracting information is a cumbersome process. Simple problems such as the search for a specific piece of information is almost infeasible and requires a large amount of human involvement.

Digitizing document collections mainly involves two steps. First, a digital image of the respective document is created through scanning. This preserves the document contents and allows for long time storage but still, by itself, having a digital image does not make the information more accessible. Extracting information usually implies a transcription of the handwritten document or to a transfer of its content to some form of structured data. Even though automatic recognition systems have been extensively researched for many years, this process is still mostly done manually.

Recognizing handwriting is a challenging problem due to its variability and ambiguities that are, additionally, often affected by degradation. Recent systems achieve impressive results when confronted with contemporary writing. Nonetheless, a severe drop in performance can be perceived with respect to historic documents with a more distinct style. This performance difference can be easily explained by the nature of the applied models. Modern automatic recognition systems rely on neural networks that are trained on sample data. This requires the collection of representative data that needs to be annotated manually, which is cumbersome process. For many historic collections, the creation of a specific annotated training dataset is required but is impossible to create due to its manual effort. Therefore, the use of machine learning models for extracting information from these types of collections is often unrealistic.

This thesis investigates a method that alleviates this problem by training machine learning models without the use of any manually annotated sample data. It is shown that it is feasible to derive well performing models for two classical document analysis problems, namely *Word Spotting* and *Handwritten Text Recognition*, with no manually labeled data involved in the training process. At the core of this works lays the concept of self-training which describes the process of learning from pseudo-labels generated by an intermediate model. By proposing the use of an initial model trained on synthetic data, all labels required for training the different models are generated automatically. This thesis shows the applicability of that idea to the respective document analysis tasks and evaluates several crucial considerations with respect to the training scheme and synthetic data generation. Aspects of the different contributions of this work have been previously published in several publications, as described in the following section.

1.1 CONTRIBUTIONS

This thesis and its related publications have made several contributions to the field of document analysis. Overall, the main concern was the development of a training paradigm that allows for training neural networks in an annotation-free manner, which means without the need of manually annotated data. Therefore, this line of research developed a self-training scheme for two classical tasks for handwritten document analysis that both rely heavily on the use of neural networks. In order to use self-training in an annotation-free manner, several core aspects need to be solved. Additional to the general application of the self-training scheme, core contributions were made with respect to synthetic data generation and confidence estimation for an effective pseudo-label selection. A summary of this work and its main findings is published in [WF24]. In the following, core contributions by the author that have been previously published will be discussed in more detail.

Self-Training for Word Retrieval and Recognition

An initial publication considered the word spotting model TPP-PHOCNet and showed that self-training is a viable option to train the respective neural network. The TPP-PHOCNet is a well established neural network for attribute prediction that was originally proposed in [SF16] and can be applied to the problem of segmentation-based word spotting. In [WF20], the author showed that a self-training approach can be used to train the network without the need of manually annotated data. In case of the TPP-PHOCNet, two main problems needed to be solved. First, self-training relies on an initial model. As a first step, this initial model was derived by training on a synthetic dataset from the literature. Furthermore, it was not obvious how to derive pseudo-labels in the case of the attribute-based retrieval model. In [WF20], the author showed that it is feasible to use lexicon-based recognition to predict pseudo-labels that can be used as training targets of the TPP-PHOCNet. A further main conclusion of the publication was that no accurate lexicon is required and the use of the most common English words as an approximations already suffices to observe significant performance improvements.

While the proof of applicability of self-training to the retrieval model was published in [WF20], the author later adapted the same idea to a handwriting recognition model. In [WF22b], a sequence-to-sequence based recognition model was trained following the principles of [WF20]. Similar findings were made and it was shown that self-training significantly helps to adapt the respective model from synthetic to real data. In contrast to the retrieval model, it was shown that self-training is feasible without the inclusion of an external lexicon.

In both publications [WF20; WF22b], the author was responsible for developing the general training approach. All experiments and the evaluation of the respective performances were conducted by the author of this thesis.

Confidence Estimation and its Integration to Self-Training

Finding a numerical estimate on how confident a neural network is in its prediction is an open research question. In the context of self-training, finding such a confidence

measure offers the potential to select less erroneous pseudo-labels. The author made several contributions with regards to confidence estimations for document analysis models in close relation to self-training.

In [WOF19], the idea of confidence estimation was first explored for a TPP-PHOCNet in the context of word spotting. Several different confidence measures were proposed and it was shown that they generally measure prediction quality. The author was responsible for the implementation of the non-metaclassifier based confidence measures, the experimental setup and the evaluation. The idea of confidence estimation was later picked up and integrated in the self-training approaches for pseudo-label selection. The author showed that including confidence-based selection improves performances for word spotting and handwriting recognition in [WF20] and [WF22b]. In both cases, the author developed the method on how to integrate the confidence measures in the training scheme and conducted the experiments providing the empirical evidence of their effectiveness.

Synthetic Data Generation and its Language Implications

For the proposed self-training approach it is necessary to train an initial model. In order to avoid the use of manually-annotated data, synthetic datasets offer a viable solution. Several works in the literature show that training a model on synthetic data and later fine-tuning on a small labeled dataset is a promising training strategy. In [WF20] and [WF22b], the author showed that despite the comparably poor performances, initial models solely trained on synthetic data can be used for self-training. In the literature, the process of font-based synthetic data generation is not extensively researched. Many works simply generate a large number of samples based on a predefined lexicon or a collection of texts. Little consideration has been given to the variability of visual or language properties. The common approach is typically to aim for large scale datasets with as much variability as possible.

In this regard, the author made several contributions by investigating how visual and language properties of a synthetic dataset influence performances. In [WBF20], several experiments were conducted on generating synthetic data for the aforementioned word spotting model TPP-PHOCNet. The author concluded that the selection of the vocabulary underlying the synthesis pipeline heavily influences the final model performance. Furthermore, a method to adapt the visual style of the synthetic dataset to a target dataset was presented. Style adaptation was performed by training a font and slant predictor network on synthetic data, which was then used to predict a font and slant distribution for the target dataset. The assumption was that fonts more frequently predicted were visually similar to the real data. The derived distributions were then used to generated style-adapted synthetic datasets.

In [WF22b], experiments on synthetic data generation for handwriting recognition were published. Here, mainly different language properties of synthetic datasets and the implications on final recognition performances were investigated. The author showed that training a model on synthetic data conducts a form of implicit language modeling and that removing a natural word distribution affects performances.

For the mentioned publications, the author of this work was responsible for designing the synthesis pipeline, designing the methods and conducting the respective experiments.

1.2 OUTLINE

This thesis is organized in six chapters, with the current chapter introducing the main problem, the motivation and the contributions made by the author. The remaining chapters are organized as follows:

Chapter 2 - Fundamentals

The models that are used to address the problems of word recognition and retrieval are based on neural network architectures. This chapter introduces the fundamental building blocks of modern neural architectures and how to optimize them. First, the artificial neuron as the basic computing unit is introduced. This is followed by the discussion of the most relevant building blocks of deep neural networks and how to find a set of weights by training on sample data.

Chapter 3 - Handwritten Document Analysis

Document analysis is a long-standing research field in the computer vision community. In this chapter, the most significant works with respect to word recognition and retrieval in handwritten documents is discussed. Additionally, relevant works on data synthesis are reviewed as this is a major concern of the proposed method.

Chapter 4 - Self-Training

This chapter presents the proposed method for self-training of the retrieval and recognition model. After a discussion of the background of the training method, the respective models are described in detail. First the process of generating a suitable synthetic dataset for training an initial model is described. Given the trained initial model, the chapter explains how self-training can be applied to increase model performances and how to integrate a confidence-based selection of pseudo-labels.

Chapter 5 - Experimental Evaluation

In order to evaluate the proposed methods, empirical experiments are conducted on several datasets. This chapter describes the evaluation procedure including the benchmarks, protocols and metrics. The obtained results allow for various insights in the different aspects of the proposed method. Furthermore, the annotation-free performances for retrieval and recognition are compared with results from the literature.

Chapter 6 - Conclusions

The final chapter concludes with a summary of the main findings of this thesis and a discussion of further research directions.

2 FUNDAMENTALS

Perceiving their environment, recognizing patterns and taking actions accordingly are tasks which humans constantly perform. The entire recognition process from raw data, such as a visual or an auditory signal, to a known category is mainly performed subconsciously. While humans are excellent in solving diverse recognition tasks, designing machines that are capable of reproducing or even exceeding human perception poses a tremendously complex problem. The field of computer science that is concerned with modeling this recognition process is called *Pattern Recognition* [DHS01, p. 1]. In this thesis, the tasks of interest are word recognition and retrieval. Both constitute classical pattern recognition problems and are usually related to image processing. While pattern recognition in general is not limited to specific types of raw data, word recognition and retrieval can be categorized as *Computer Vision* tasks, which refer to the problem of classifying and processing image contents. Classical computer vision systems share common characteristics and may be broken down into a shared processing pipeline [DHS01, p. 2]. Figure 1 presents the essential building blocks that can be identified in a traditional computer vision system.

First, a digital representation of the visual perception of the environment is necessary. This step is called *image acquisition* and describes the process of mapping a three-dimensional scene onto a two-dimensional digital image. In order to acquire an image, a physical process, for example the reflection of light is observed by an imaging system. The image is then formed by projecting the received radiation on a two-dimensional plane such as a camera sensor. Finally, the continuous signal is sampled and quantized to produce a digital image.

After acquiring the digital image, *preprocessing* is commonly performed. Due to the acquisition process and the signal itself, the collected data usually contains a lot of variance and noise that are irrelevant with respect to the designated task. For example, considering the text recognition process for a word image, the color of the text is likely to be irrelevant for recognizing characters. Therefore, binarizing text images into fore- and background pixels is a commonly applied technique. In general, the goal of preprocessing is to remove unwanted variance from data to benefit the later processing steps. Common preprocessing techniques involve normalization, binarization and filtering techniques [Sze10, p. 101]. Designing preprocessing pipelines is vastly heuristic and their usefulness may only be proven empirically with respect to a specific task.

Images constitute data of high dimensionality with a lot of information irrelevant to a later classification task. Finding a numeric representation that is descriptive with respect to the task at hand is referred to as *feature extraction*. Two principal approaches are commonly disinguished, which are analytic and heuristic methods. Analytical methods, such as Principal Component Analysis (PCA), optimize a defined criterion; in this case maximizing the variance along each dimension of the data [DHS01, pp. 115-118]. On the contrary, heuristic features are based on human design involving experience and intuition. Heuristic features share the same problem as preprocessing in the sense that their effectiveness can only be evaluated given

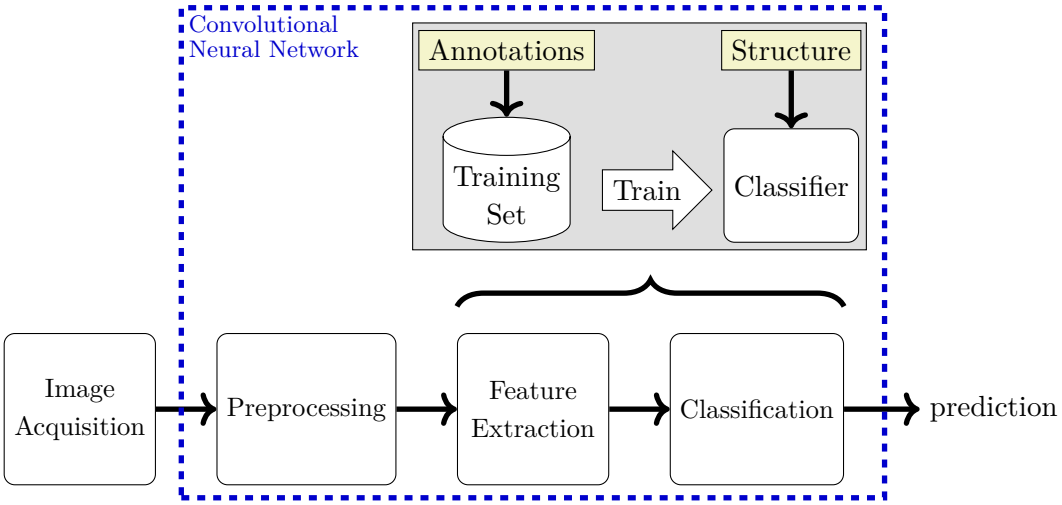


Figure 1: General architecture of a classical computer vision system. As indicated, modern convolutional neural networks combine multiple steps of the pipeline in a single computational model.

the entire processing pipeline regarding the ultimate task. In computer vision, local image features such as edges, texture, motion and image descriptors were popular [Sze10, p. 333]. Local image descriptors, especially those which represent a small image patch by gradient statistics, built the basis for the majority of computer vision systems. Common descriptors that were also heavily explored for text recognition are SIFT [Low04], HOG [DT05] or LBP [OPH94].

For a classification task, a classifier then makes the final prediction based on the given set of features. In order to derive a classifier, first the general structure is selected a priori. Here, structure not only means the choice of the classifier method itself but also its architecture, which also directly relates to the number of parameters of the model. Without knowledge of the data to be processed, a single best choice for the type of classifier does not exist. This fact is referred to as the "No Free Lunch" theorem [Wol96]. Therefore, choosing and designing a suitable classifier needs to be performed by a human expert and is usually optimized by empirical evaluations on test data. Over the years, a large number of methods were established for varying classification tasks. Typical examples are Support Vector Machines (SVMs), Random Forests, Bayes classifiers or Artificial Neural Networks. Given a parameterized classifier, solving the classification task boils down to finding an optimal set of parameters. This can be done by exploiting sample data which consist of representative data points with respective annotations. Based on this training set, algorithms exist for all kinds of classifiers that try to find an optimal set of parameters. This process is called supervised training. While a majority of the success of machine learning techniques can be explained by learning from sample data, it also poses a tremendous drawback. Annotations for sample data need to be created manually and, especially for complex tasks, samples are required in a large quantity. In this thesis, it is shown that the paradigm of supervised learning can be extended to learning from unlabeled data, thereby reducing the annotation effort to a minimum. As annotation effort is one of the most limiting factors in the application of a computer vision system based on machine learning, partially removing the necessity of

supervison has been of high interest. For a detailed discussion of various supervision schemes, see Section 2.3.2.

Looking at modern computer vision systems shows that the field is mainly dominated by neural network such as Convolutional Neural Networks (CNNs) and transformer networks. This holds also true for document analysis tasks to an extent such that neural networks are almost exclusively the one classifier type that is applied. Even though this contradicts the "No Free Lunch" theorem at first sight, it can be explained by the end-to-end structure of neural networks for vision applications. CNNs and transformer networks allow the steps of preprocessing, feature extraction and classification to be integrated in a single model that may be optimized with respect to annotated sample data. As discussed earlier, preprocessing and feature design include a high amount of human involvement. In this regard, neural networks have an inherent advantage by integrating the aforementioned steps into a single model in addition to the availability of efficient optimization algorithms. Nonetheless, learning potential preprocessing steps and descriptive feature representation also results in larger annotated datasets that are required to train the given model.

In the following chapters, the essential components of neural networks will be discussed in more detail, as they are the foundation of the models in this work. After the introduction of the perceptron as the basic computing unit of an artificial neural network in Section 2.1, the building blocks of modern deep neural network architectures, especially CNNs and Recurrent Neural Networks (RNNs) are introduced in Section 2.2. Section 2.3 discusses standard techniques on how to optimize the parameters of a neural network in a fully supervised setting. Finally, Section 2.3.2 discusses alternative learning settings to the fully supervised paradigm, in which the self-training method presented in this work also positions itself.

2.1 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) are nowadays the dominant model for all kinds of pattern recognition applications. In general, they consist of a high number of simplistic units that are connected in large networks. This concept is loosely inspired by the structure of biological neural networks and the functionality of the human brain [Kon05]. Today, neural networks have evolved from their biological counterpart, but can rather be considered a powerful mathematical framework that allows for end-to-end machine learning. This chapter introduces the fundamental components of an ANN and the structure of the most basic network. The basic computing unit that lies at the core of any neural network is the so-called perceptron. ANNs are implemented by the connection of multiple perceptrons in combination with non-linear activation functions.

2.1.1 *Perceptron*

The idea of an artificial neuron as a computing unit was first proposed in [MP43]. In its simplest form, this neuron has n binary input signals $x_0, x_1, \ldots, x_n$ and computes

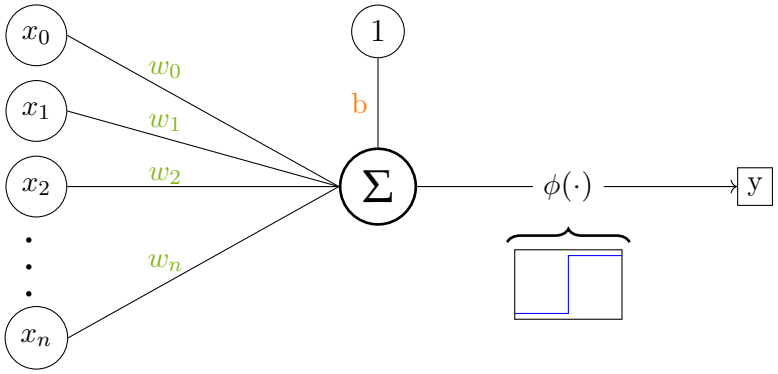


Figure 2: The structure of an artificial neuron with n inputs x, weights w, a bias b and an activation function ϕ.

a binary output. The neuron itself implements a function f that computes the sum of all inputs and compares the result against a threshold τ:

$$f(x_0, x_1, ..., x_n) = \begin{cases} 1, & \text{if } \sum_{i=0}^{n} x_i \geq \tau \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Already this simplistic structure shows the power of neurons as function approximators. By choosing different values for τ, different distinct binary functions can be realized. Consider, for example, a neuron with only two binary inputs x_0 and x_1. With $\tau = 1$ the neuron realizes the logical **OR**, while setting $\tau = 2$ implements the logical **AND**.

The aforementioned general structure of the artifical neuron persevered. In [Ros58], Frank Rosenblatt extended the neuron by the addition of weights to each input and a general bias. Additionally, Minsky and Papert generalized the model to use real values in [MP69] resulting in the basic structure of the general perceptron. Figure 2 visualizes the perceptron with inputs $\mathbf{x} \in \mathbb{R}^{D_x}$, weights $\mathbf{w} \in \mathbb{R}^{D_w}$, a bias $b \in \mathbb{R}$, an activation function $\phi : \mathbb{R} \to \mathbb{R}$ and the output $y \in \mathbb{R}$. The perceptron essentially computes a weighted sum over its inputs which is then the input to a non-linear activation function:

$$f(\mathbf{x}) = \phi(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b). \tag{2}$$

The updated perceptron as presented in [MP69] allows for the realization of complex functions and can, for example, naturally implement a simple binary classification task. Given two classes $y \in 0, 1$ and using the step function as the activation function $\phi(\cdot)$, the perceptron may classify an input vector $\mathbf{x}$:

$$y(\mathbf{x}) = \begin{cases} 1, & \text{if } \mathbf{w}^{\mathrm{T}}\mathbf{x} + b > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

As shown in [MP69] the capabilities of the model are still extremely limited, as it is not able to solve problems that are not linearly separable, like the **XOR**.

2.1.2 *Multilayer Perceptron*

The main goal of a neural network is to approximate an unknown function f^*. This function can, for example, represent a classification problem where a numerical fea-

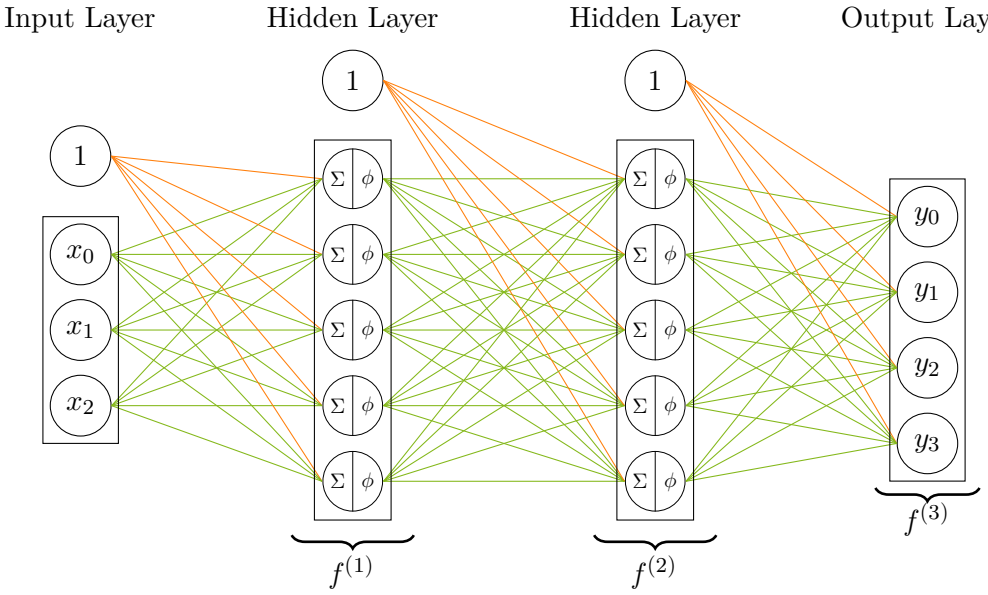


Figure 3: A multilayer perceptron with an input layer, two hidden layers and an output layer. Green connections represent weights and the orange connections biases.

ture representation $\mathbf{x}$ has to be mapped to an output class y with $y = f^*(x, \theta)$. Here, θ represents the set of weights corresponding to a perceptron that are optimized based on sample data. As discussed in Section 2.1.1, the capabilities of a single perceptron as function approximator are quite limited. Nonetheless, connecting multiple perceptrons in a layerwise fashion has been shown to yield a powerful model.

Figure 3 shows a simple network architecture that uses perceptrons as basic building blocks [GBC16, pp. 164 - 167]. Each set of neurons can be considered to implement a basic function f. In the given example, three such basic functions, $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$, are realized by 5, 5 and 4 neurons, respectively. For a given input $\mathbf{x}$ that is fed into the network by the *input layer*, the overall output is computed by composing the three basic functions as $\mathbf{y} = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Each of the basic functions can be considered a layer of the network and the number of layers is called depth. The last layer of the network is the *output layer* with, in the case of a classification network, each neuron corresponding to a specific class. All layers between the input and output layer are considered *hidden layers*, as during training only samples for the relation of input and output are provided. In this simple architecture, all neurons from one layer are connected to all neurons of the next layer. Therefore, the considered network is a fully-connected feedforward network [Kon05]. It especially means that in contrast to recurrent networks no feedback connections exist [GBC16, p. 164]. The information flow of the network is unidirectional and each layer only has access to the outputs of the previous layer.

At each layer, the individual neurons compute a weighted sum over their inputs as described in Equation 2. Therefore, every connection in the network is associated with a trainable weight leading to a high number of parameters for fully-connected architectures. In order to benefit from multiple layers of perceptrons to approximate complex functions, non-linear activation functions are necessary. Without the introduced nonlinearity, the function implemented by the network could be reduced to

a multiplication of the input vector $\mathbf{x}$ and a single large weight matrix $\mathbf{W}$. Given a suitable non-linear activation function, this is not the case and the model is able to approximate highly complex functions.

Interestingly, the Multi Layer Perceptron (MLP) is already an incredibly powerful model. In [HSW89], it has been shown that an MLP with only one hidden layer and the output layer containing weights is capable of approximating any real-valued multivariate function. This proof is known as the "Universal Approximation Theorem" [HSW89] and shows the potential of neural networks which are built from high numbers of simple neurons. While the MLP is theoretically capable of approximating any function, in practice other architectures are combined with fully-connected parts. This is mainly due to the fact that the approximation of a complex function usually coincides with a high number of required neurons. By introducing more neurons, the number of parameters is drastically increased especially for high dimensional data.

2.1.3 *Activation Functions*

As discussed in Section 2.1.1, each neuron of an ANN applies an activation function to the weighted sum of its inputs. In order to build complex network architectures, it is essential for these functions to be non-linear. This chapter discusses the most common choices for activation functions that can be found in state-of-the-art architectures.

Early approaches attempted to model the thresholding characteristic of a step function, as inspired by the biological example. Furthermore, differentiable functions were of desire as many optimization algorithms are based on computing gradients. One function that fulfills these criteria is the *sigmoid* function that was historically a popular choice as a non-linear activation function [Roj96]. The sigmoid functions maps its input to the range of $[0, 1]$, intersects the y-axis at a value of $y = 0.5$, converges to zero for $x \to -\infty$ and converges to one for $x \to \infty$, see Figure 4, top left. Mathematically, the sigmoid and its derivative are defined by

$$\phi(x) = \sigma(x) = \frac{1}{1 + \mathrm{e}^{-x}} \qquad \phi'(x) = \sigma'(x) = \frac{\mathrm{e}^{-x}}{(1 + \mathrm{e}^{-x})^2}. \tag{4}$$

One drawback of the sigmoid function is that it is not zero-centered. This is considered to be unfavorable for gradient-based optimization algorithms [Roj96]. To overcome this property, a scaled and translated version of the sigmoid can be used. The resulting hyperbolic tangent maps an input to the dynamic range of $[-1, 1]$, while maintaining most of the properties of the classical sigmoid, see Figure 4, top middle. The *tanh* function is defined as

$$\phi(x) = \tanh(x) = 2(\sigma(x) - 1) = \frac{1 - \mathrm{e}^{-x}}{1 + \mathrm{e}^{-x}} \tag{5}$$

$$\phi'(x) = \tanh'(x) = 1 - \tanh^2(x)). \tag{6}$$

While sigmoid and tanh activations have been historically popular, they are rarely used today. Both functions have a problematic characteristic with respect to gradient-based optimization. Computing the gradient of the network with respect to a given goal function results in the multiplication of the derivatives of the activation functions at each layer. As the derivatives of the sigmoid and tanh function, as defined

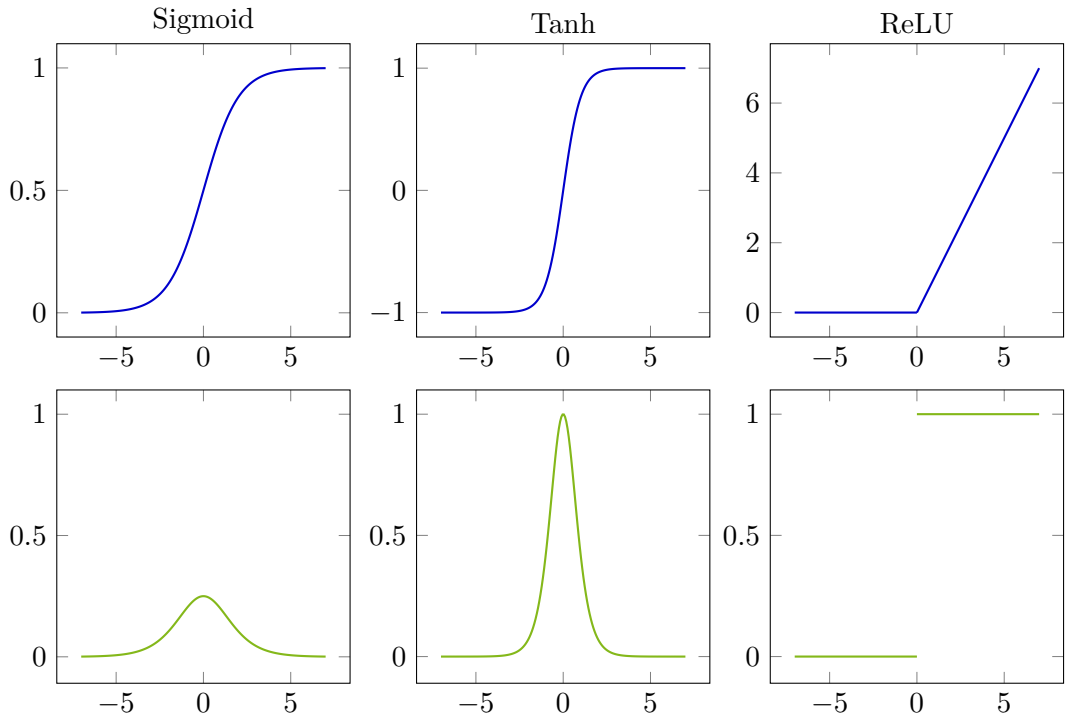


Figure 4: Common activation functions (top row) and their respective derivatives (botton row).

in Equation 6 (see Figure 4 bottom row), do not exceed a value of one, the gradient diminishes for deeper network architecture. Therefore, weights are barely updated which leads to a poor optimization behavior. This problem is known in the literature as the "Vanishing Gradient" problem [GBC16, p. 290].

The popularity and success of neural networks in recent years can be explained to some extent by solving the "Vanishing Gradient" problem which allows for larger architectures that contain a high number of hidden layers. In this regard, the proposal of the Rectified Linear Unit (ReLU) as an activation function can be considered a major breakthrough [GBB11]. As depicted in Figure 4, top right, the ReLU realizes a non-linear thresholding behavior by mapping its input to zero for all values smaller than zero and computing the identity for all values larger than zero:

$$\phi(x) = f(x) = \max(0, x) \qquad \phi'(x) = \begin{cases} 1, & \text{if } x \leq 0 \\ 0, & \text{if} x > 0. \end{cases} \tag{7}$$

A closer look at the derivative (Figure 4, bottom right) shows that the derivatives of the ReLU is either zero or one. Therefore, multiplying the derivatives per layer does not lead to an incremental scaling of the gradients. Additionally, the ReLU is computationally simple as it does not require the evaluation of exponential functions, as is the case for sigmoid and tanh activation functions.

Several variants of the ReLU have been proposed that address certain disadvantages of the classical ReLU. Due to the ReLU having a gradient of zero for negative inputs, certain neurons may never be activated. For example, this can happen if, during training, one of the neurons ends up with negative weights. With ReLU acti-

vations in the previous layer, the input of the ReLU is always negative and its gradient zero. Therefore, the neuron won't activate and its weights won't be updated. In the literature, this characteristic is known as "dead neurons" [GBC16, p. 238]. Dead neurons can be avoided by using a LeakyReLU which does not map negative values to zero, but rather scales them by a small factor λ [MHN+13]. The gradient of the LeakyReLU is, therefore, not zero for negative inputs and weights may still be updated. The Parametric ReLU follows the same idea but defines the slope λ as a learnable parameter [He+15a]. All previously described variants of the ReLU are not differentiable at zero. The Exponential Linear Unit (ELU) solves this by using a scaled exponential function for negative values [CUH16]. While all these variants of the ReLU have been shown to be empirically beneficial for certain applications, performance differences are comparably small [Li+18]. Therefore, the classic ReLU is by far the most used and most popular activation function due to its simplicity.

The aforementioned activation functions only consider individual neurons. When applying a neural network to a classification task, the output layer usually relies on a different activation function. Commonly, each neuron of the output layer corresponds to a class. For classification, the network shall activate the neuron corresponding to a specific class and shall output zero for all other neurons. In order to enforce this characteristic, the *softmax* activation function has become common practice [Bis06, p. 198]. With x_i being the linear activations of the neurons of the output layer, the softmax computes the activation $\phi(\mathbf{x})_i$ of each neuron i based on the relation of the individual activations to the sum of all output activations:

$$\phi(\mathbf{x})_i = \text{softmax}(\mathbf{x})_i = \frac{\mathrm{e}^{\mathrm{x_i}}}{\sum_{j=1}^{k} \mathrm{e}^{x_j}}. \tag{8}$$

An interesting characteristic of the softmax function is that it maps its input to the range of $(0, 1)$ and the sum over all outputs equals one:

$$\sum_i \text{softmax}(\mathbf{x})_i = 1. \tag{9}$$

Due to its similarity to a probability distribution, the softmax output is often considered to be an estimate for the probability $p(y_i|\mathbf{x})$ to observe class y_i given the activation vector $\mathbf{x}$.

2.2 DEEP NEURAL ARCHITECTURES

As discussed in the previous Section 2.1, an ANN building upon fully-connected perceptrons is capable of approximating any arbitrary complex function, if it contains at least one hidden layer and a sufficient number of neurons. Nonetheless, networks that only contain fully-connected neurons are rarely used for common pattern recognition and especially not for computer vision problems. A main reason for this is the high number of parameters corresponding to fully-connected architectures given high dimensional input data. Consider, for example, the MNIST dataset which was one of the first computer vision benchmarks also tackled with neural networks [LeC+98]. The dataset contains images of handwritten digits with a dimension of 28×28 pixels. Already a very small MLP, with 500 neurons in one hidden layer and an output layer with 10 neurons corresponding to the respective classes, contains

almost 400,000 trainable parameters. This poses a tremendous optimization problem. Particularly, a high number of parameters usually coincides with a requirement for large amounts of annotated training data which is a severe limitation to the applicability of a machine learning system.

State-of-the-art architectures circumvent this problem by applying different types of layers that usually allow for sparse interactions between the neuron and the preceding layer. Modern architectures usually stack a high number of layers in a hierarchical fashion [SZ15; He+16; Hua+17]. Networks with many hierachical layers are called Deep Neural Networks (DNN) and have been proven to be a very successful approach to tackling pattern recognition problems [Ben09]. No clear definition of the term *deep* exists and some authors already consider any network with more than three layers to be *deep* [GBB11].

This chapter discusses typical layers required to build deep network architectures that are also the essential building blocks of the models applied in this work. *Convolutional networks* allow for sparse interactions and are common practice for most computer vision applications. Whenever sequential data is considered, networks often rely on *recurrent* connections that model context information. Recent architectures leverage the concept of *attention*. Attention introduces a set of learnable weights that allows the network to focus on specific parts of its input data.

2.2.1 *Convolutional Neural Networks*

Convolutional Neural Networks rapidly became the predominant model in computer vision after their first application to a handwritten digit classification task in [LeC+98]. Motivated by the success of the proposed *LeNet*, CNNs were the model of choice for various computer vision tasks to achieve state-of-the-art performances and they dominated typical benchmark challenges [Rus+15b]. They usually contain three different types of layers, namely *convolutional*, *pooling* and *fully-connected* layers, that will be discussed in the following.

Convolutional Layers

A convolutional layer is derived from the convolution operation. Let $x : \mathbb{R} \to \mathbb{R}$ be a real valued function of t and let $w : \mathbb{R} \to \mathbb{R}$ be a weighting function. The convolution operation denoted with an asterix $*$ is defined as [GBC16, p. 322]:

$$f(t) = (x * w)(t) = \int x(a)w(t-a)da. \tag{10}$$

A convolutional layer is based on the generalization of the convolutional operation to image data. Therefore the operation needs to be reformulated for a discrete signal $x(t)$. With an integer valued time index t, the discrete covolution can be noted down as:

$$f(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{11}$$

In order to apply a convolution to image data, a two dimensional image $\mathbf{I}$ needs to serve as an input instead of the discrete signal x. Let $I : \mathbb{N}^2 \to \mathbb{R}$ be a function that describes an input image by mapping pixel positions (i, j) to real valued pixel val-

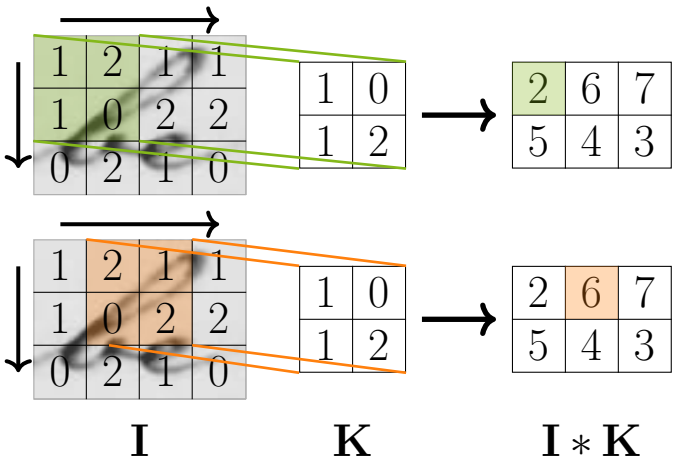


Figure 5: The convolution operation of a two-dimensional image $\mathbf{I}$ of size 3×4 with a kernel $\mathbf{K}$ of size 2×2. $\mathbf{I} * \mathbf{K}$ refers to the resulting feature map.

ues. For a two-dimensional kernel $\mathbf{K}$ of fixed dimensions the following convolutional oparation is derived:

$$F(i,j) = (\mathbf{K} * \mathbf{I})(i,j) = \sum_m \sum_n \mathbf{I}(i,j)\mathbf{K}(i-m, j-n). \tag{12}$$

Intuitively, this operation can be interpreted as visualized in Figure 5 which shows the convolution of an image $\mathbf{I}$ of size 3×4 with a kernel $\mathbf{K}$ of size 2×2. The size of the kernel defines a patch that is continuously moved over the image in a sliding window fashion. The corresponding step width is called stride s and is equal to one in the given example. At every window position the dot product between the kernel weights and the inputs is computed. The resulting two-dimensional output is known as a feature map. Applying the two-dimensional convolution is highly similar to a perceptron. In contrast to a connection to all input pixels, the convolutional neuron is only sparesly connected to the pixels inside the moving window which is also called the receptive field. Similar to the classical perceptron, the convolutional neuron is combined with a non-linear activation function. Multiple kernels are used with k being the number of kernels in the convolutional layer. The convolutional layer is then defined by its number of kernels k, their dimensions f and the stride s [FJK23].

When considering convolutional networks, the input data as well as the computed feature maps can be represented as volumes with width, height and depth $[W, H, D]$. An *RGB* image, for example, then corresponds to a volume with respective image dimensions and a depth of $D_{RGB} = 3$. The dimensions of the feature map computed by a convolutional layer depends on the input dimensions, its stride and the number of kernels. Furthermore, it is common practice to enclose the input with a number of p zeros which is known as zero padding. The resulting dimensions $[W_2, H_2, D_2]$ of a feature map computed by a convolutional layer for an input with dimensions $[W_2, H_2, D_2]$ are given by [FJK23]:

$$W_2 = \frac{W_1 - f - 2p}{s+1} \qquad H_2 = \frac{H_1 - f - 2p}{s+1} \qquad D_2 = k. \tag{13}$$

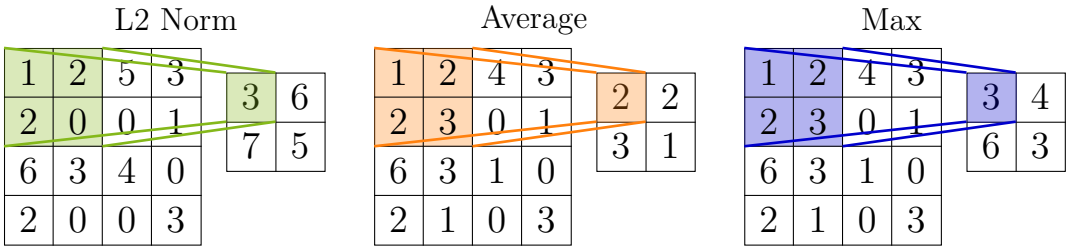


Figure 6: Pooling layer with three different aggregation functions.

Without the addition of a bias, the number of weights in a convolutional layer boils down to $f \cdot f \cdot D_1 \cdot k$. As the kernel is usually significantly smaller than the input, using convolutional layers leads to a comparably small number of weights.

Besides the sparse connectivity, convolutional layers have further beneficial properties [GBC16, p. 325]. While a fully-connected perceptron computes its function once for the entire input volume, parameters are shared in a convolutional layer. Each kernel realizes a function that is evaluated for different inputs defined by the position of the receptive field. Therefore, the network does not have to learn a distinct set of features for different positions.

Pooling Layers

Additional to convolutional layers, common CNN architectures employ so-called pooling layers [GBC16, p. 330]. They operate in a similar way with respect to sliding a window over the input data and computing a function at each position. The main goal of using pooling layers is to downsample their input by aggregating the values of the input window into a single value. This aggregation stage serves multiple purposes. It further reduces the dimensionality of the feature maps leading to fewer parameters in the later stages of the network. The receptive field of a neuron is considered all input pixels that contribute to the input of a neuron. With respect to the input data, the introduction of pooling layers increases the receptive fields of succeeding neurons. Furthermore, pooling layers introduce an invariance to local translations. The exact position of a feature inside the pooling window is not relevant, which can be an advantage if the sheer presence or absence of a feature is more useful with respect to the classification problem [GBC16, p. 332].

Figure 6 visualizes three different pooling functions with a window size $f = 2$ applied to a feature map of 4×4. With a stride of $s = 2$, the dimension of the resulting feature is reduced to 2×2. L2-norm pooling considers the input window to be a one-dimensional vector and computes the respective L2-norm. Average pooling computes the mean over the respective input values [GBC16, p. 330]. A common approach, which is also the most simple computationally, is max-pooling [ZC88]. In this case, only the maximum of the input window is propagated.

For an input volume of dimension $[W_1, H_1, D_1]$, a pooling layer with stride s and size f leads to a smaller ouput volume of size $[W_2, H_2, D_2]$ [FJK23]:

$$W_2 = \frac{W_1 - f}{s + 1} \qquad H_2 = \frac{H_1 - f}{s + 1} \qquad D_2 = D_1. \tag{14}$$

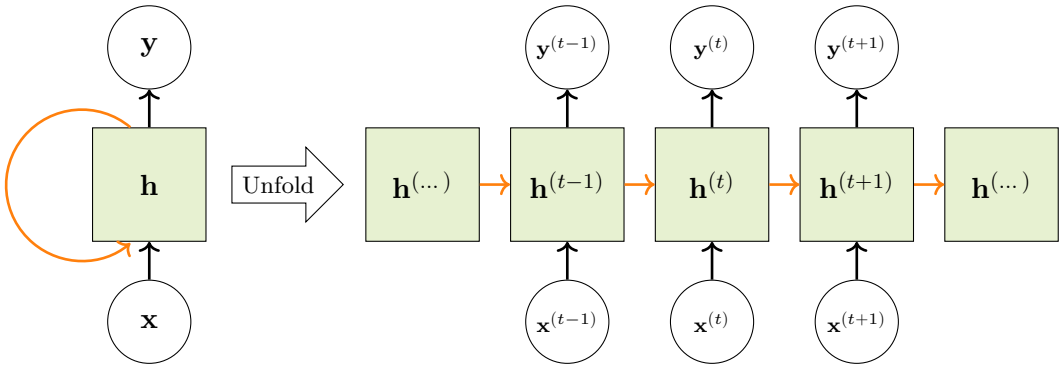


Figure 7: A recurrent neural network including a simple feedback connection propagating the output at each time step t to its input. The recurrent architecture can be translated to a feedforward network by unfolding the temporal dependencies.

Fully Connected Layers

The final layers of a CNN are usually fully-connected. For classification, these layers follow the same principal as an MLP with each output neuron corresponding to a designated class. Therefore, the last feature map of size $[W, H, D]$ given by a pooling or convolutional layer is flattened to a one-dimensional vector with a corresponding size of $[(W \cdot H \cdot D), 1]$ [FJK23]. The fully-connected layer is defined by its number N of neurons, which are each connected to all elements of the input vector. Despite the dimensionality reductions in the earlier layers, many networks have most of their weights in their fully-connected part. Given an additive bias for each neuron the number of weights can be computed as $W \cdot H \cdot D \cdot N + N$.

2.2.2 *Recurrence*

Many pattern recognition tasks operate aim at the prediction of sequences. In this regard, text recognition is a classical example as the recognition result is a sequence of individual characters. It is a natural assumption that individual predictions are not independent from each other, but rather depend on previous outputs. Recurrent Neural Networks are a common approach to model these dependencies [RHW86]. In contrast to feedforward networks, RNNs contain recurrent connections.

Let $\mathbf{x}$ be a sequence of feature vectors $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots \mathbf{x}^{(T)})$, from which we aim at the prediction of an output sequence $\mathbf{y}$ of $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)}, \dots \mathbf{y}^{(T)})$. A simple recurrent network can be designed by providing the prediction vector $\mathbf{y}^{(t)}$ as an additional input to the network at each time step. Traditionally, recurrent models define a hidden state $\mathbf{h}$. See Figure 7 for a visualization of a recurrent model [GBC16, p. 368]. At every time step the internal hidden state is updated according to the model parameters θ, the previous hidden state $\mathbf{h}^{(t-1)}$ and the input vector $\mathbf{x}^{(t)}$ at time step t. The corresponding output at time step t is then predicted given the current model state:

$$\mathbf{h}^{(t)} = f_h(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}, \theta) \qquad \mathbf{y}^{(t)} = f_y(\mathbf{h}^{(t)}, \theta). \tag{15}$$

Functions f_h and f_y can be, for example, realized by fully connected neural networks. The hidden state $\mathbf{h}^{(t)}$ captures information on the past sequence of inputs. While

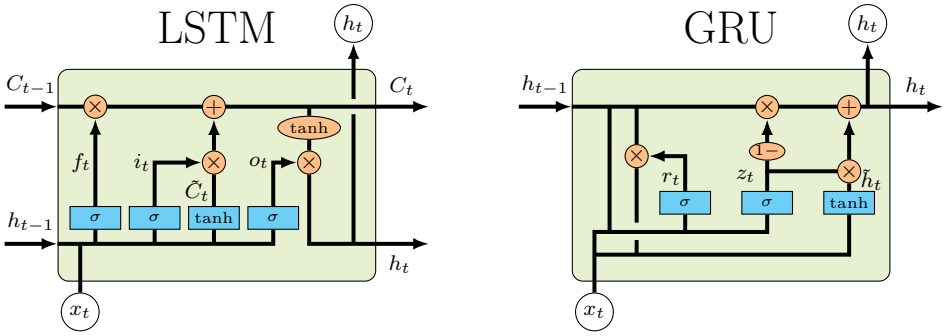


Figure 8: Structure of a Long Short-Term Memory cell (LSTM) and a Gated Recurrent Unit (GRU).

the input sequence is of arbitrary length, the hidden state of the model is usually a vector representation with fixed dimensions. The hidden state can only capture a summary of some potentially task-relevant aspects from the past input sequence [GBC16, p. 368].

Computationally, an RNN can be transferred into a non-recurrent structure by *unfolding*. This leads to a chain-like structure where the module that saves the hidden state is repeated. Unfolding the model, as illustrated in Figure 7, leads to a feedforward structure, where parameters are shared over the different time steps. Therefore, similar to CNNs, recurrence is another way to implement the idea of parameter sharing in a neural network. Here, parameters are not shared over different positions with respect to an input, but are shared over the time steps of the input sequence. By learning a single model for all time steps the model is also capable of generalizing to sequences of variable lengths.

The RNN architecture as previously introduced suffers from several problematic properties with regard to optimization with gradient-based methods. Similar to deep neural networks, the "vanishing gradient" problem occurs and, additionally, a problem known as "exploding gradient" can prevent efficient optimization [BSF94; Hoc+01]. Furthermore, it has been observed that traditional RNNs struggle with modeling long term dependencies. In order to overcome this problem modern recurrent architectures rely on so-called *memory cells*.

One of the most successful memory cells is the Long Short-Term Memory (LSTM) that was first proposed in [HS97]. While the standard RNN relies on repeating modules built upon a single network layer with non-linear activation, the LSTM depends on four distinct network layers interacting in a specific way [Zha+23]. Figure 8, left, depicts the structure of an LSTM cell with an internal cell state $\mathbf{C}_t$. Considering the LSTM as an unfolded feedforward network, the LSTM cell takes the previous cell state $\mathbf{C}_{t-1}$ as an input and outputs an updated cell state $\mathbf{C}_t$. Updating the cell state is realized by so-called gates. The gate functions are realized by trainable neural network layers and will be explained from left to right with respect to Figure 8.

First, the *forget gate* determines which information shall be removed from the cell state. Therefore, a fully-connected network with weights $\mathbf{W}_f$, biases $\mathbf{b}_f$ and sigmoid activation predicts a vector f_t that is multiplied by the previous cell state $\mathbf{C}_{t-1}$.

Given that the output of the sigmoid activation is in the range of $[0, 1]$ this can be seen as either keeping or removing components from the cell state vector:

$$f_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f). \tag{16}$$

Next, the new cell state $\mathbf{C}_t$ is computed by adding an update vector. The computation of the update is split into the computation of a set of the vector $\tilde{\mathbf{C}}_t$ and into the computation of the *input gate* $\mathbf{i}_t$. Structurally, the input gate is analog to the forget gate and computes the vector $\mathbf{i}_t$. The so-called candidate values described by $\tilde{\mathbf{C}}_t$ constitute the potential updates to the cell state. Combining the candidate values and the input gate defines which values of the old cell state are updated. Finally, the new cell state $\mathbf{C}_t$ is given by

$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \tag{17}$$
$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C), \tag{18}$$
$$\mathbf{C}_t = f_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t, \tag{19}$$

with $\odot$ denoting the element-wise multiplication. The hidden state $\mathbf{h}_t$ corresponds to the output of the cell as seen by other layers. Therefore, the *output gate* generates a filtered version of the cell state. Following the same concept as the input and forget gate, the output vector $\mathbf{o}_t$ is computed. Before filtering the cell state, the tanh function is applied. The final output or, more specifically, the forwarded hidden state $\mathbf{h}_t$ is then derived by element-wise multiplication:

$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o) \tag{20}$$
$$\mathbf{C}_t = o_t \odot \tanh(\mathbf{C}_t). \tag{21}$$

Many variants of the LSTM, which change the interactions between the different gates, have been proposed in the literature. In [GS00], so-called peephole connections are introduced, that provide the cell state $\mathbf{C}_{t-1}$ or $\mathbf{C}_t$ to each of the gates. Other variants couple the forget and input gates of the cell.

In this work, another variation is used for a sequential text recognition model, which is the Gated Recurrent Unit (GRU) proposed in [Cho+14]. Its structure is shown in Figure 8, right. The GRU summarizes hidden and internal cell states into a single hidden state vector $\mathbf{h}_t$ [Zha+23]. Similar to the LSTM, gates are used to control the information flow. Here, only two gates, the *reset* and *update* gate, are applied. The reset gate $\mathbf{r}_t$ decide which elements of the previous hidden state are persevered. After the computation of a set of candidate hidden state values $\tilde{\mathbf{h}}_t$ the update gate $\mathbf{z}_t$ combines the old state and the candidates. Mathematically, the computation can be summarized as:

$$\mathbf{z}_t = \sigma(\mathbf{W}_z[\mathbf{h}_{t-1}, \mathbf{x}_t]), \tag{22}$$
$$\mathbf{r}_t = \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}, \mathbf{x}_t]), \tag{23}$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}[\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{x}_t]), \tag{24}$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t. \tag{25}$$

While most variants of the LSTM show similar performances [Gre+17], the GRU became popular as the simplified structure speeds up computation without a significant loss in performance [Chu+14].

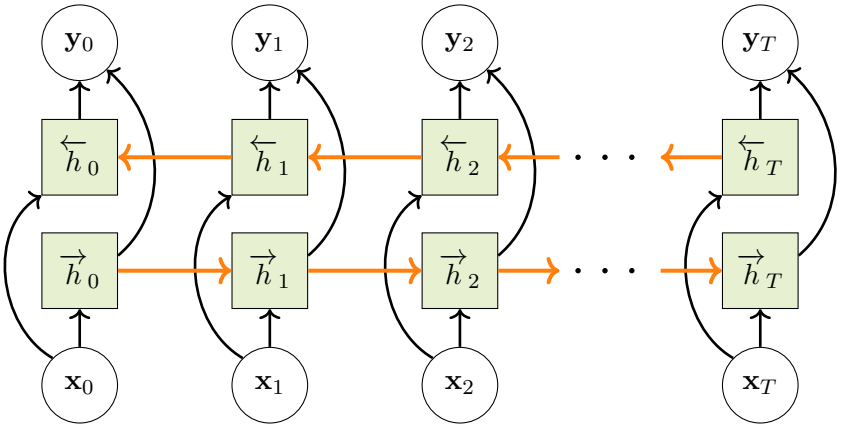


Figure 9: A bidirectional RNN, which is based on two recurrent network layers.

The standard RNN only considers the past of the sequence. Nonetheless, for many prediction tasks the future input vectors are available and it is beneficial to also consider them as context information. A simple extension of the standard unidirectional RNN is to chain two RNN layers to create a bidirectional RNN [SP97] as depicted in Figure 9. One layer processes the input sequence in regular order as described earlier. The second layer considers the respective last input of the sequence $\mathbf{x}_T$ as the first input to the recurrent layer [Zha+23]. Essentially, the layers traverse the input vector sequence in opposite directions. The respective output vectors are derived from concatenating the outputs of the individual RNN layers.

2.2.3 *Attention*

The recurrent architectures introduced in Section 2.2.2 commonly predict one output token per input feature vector. Nonetheless, many prediction tasks aim at mapping one sequence to another sequence of different length. A common solution for this type of problem is the application of so-called endcoder-decoder networks. Here, the general idea is to use an encoder to predict a context vector $\mathbf{c}$ for a given input sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$. The context vector $\mathbf{c}$ is then provided to the decoder which predicts outputs $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_T$ until the prediction of a designated end-of-signal token.

One of the first realizations of this principle was proposed in [Cho+14] as visualized in Figure 10, left. In this case, the encoder as well as the decoder are implemented by RNNs. The encoder processes the input sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$ and the final hidden state of the encoder $\mathbf{h}_{eT}$ constitutes the context vector $\mathbf{c}$. The context vector is then provided to the decoder at each time step. In [SVL14], Sutskever et al. used a similar approach but only provided the context at the initial decoding step, see Figure 10, middle. Therefore, the context vector is only used for initializing the decoder. All context information is then required to be carried on to the later decoding steps by the decoder's hidden state vector $\mathbf{h}_d$.

For both approaches, [Cho+14] and [SVL14], saving all context information in a single vector of fixed sized is considered a severe bottleneck. Especially for longer input sequences, this approach is known to struggle with modeling long-term depen-

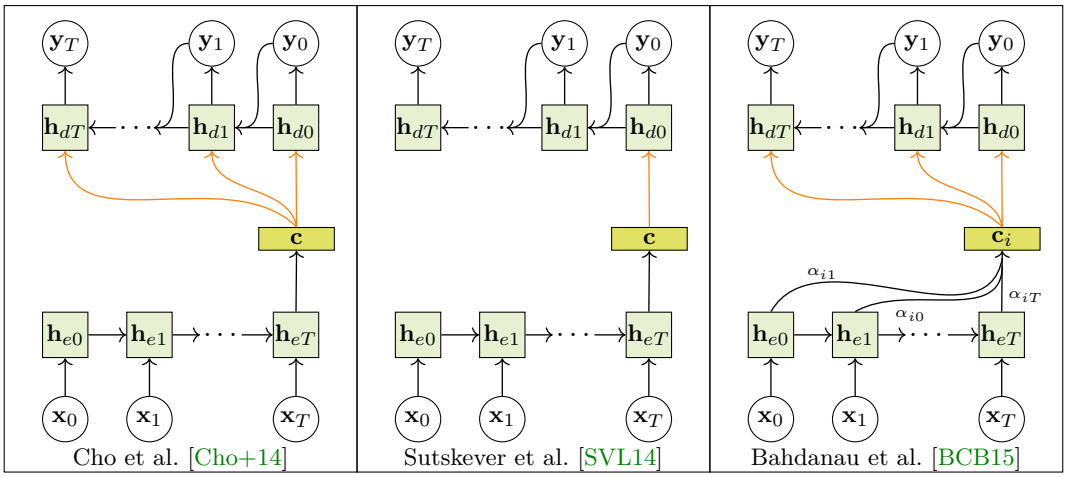


Figure 10: Principles of context vector computation in encoder-decoder architectures for sequence-to-sequence prediction.

dencies. This problem can be overcome by using the so-called *attention mechanism* as introduced by Bahndanau et al. [BCB15]. Instead of using a single context vector $\mathbf{c}$, a designated context vector $\mathbf{c}_i$ for each decoder step i is computed. This is done by weighting and aggregating the hidden states of the encoder for every decoder step:

$$\mathbf{c}_i = \sum_{j=0}^{T_x} \alpha_{ij} \mathbf{h}_{ej}. \tag{26}$$

Here, $\alpha(\mathbf{q}, \mathbf{k}_i)$ defines a scalar attention weight $\alpha \in \mathbb{R}$. Therefore, attention implements an efficient mechanism that allows a network to focus on relevant parts of its input data as shown in Figure 10, right.

The attention weights α_i are usually considered a function of the preceding decoder state at step $i-1$ and the input sequence $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$. Furthermore, the attention weights are desired to fulfill certain requirements. The weights should be non negative and sum up to one. This can be ensured by applying a softmax to the function. Let $e_{i,j}$ be a weighting function with respect to weight j and decoder step i. The attention weight α_{ij} is then computed as:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_k \exp(e_{ik})}. \tag{27}$$

Thereby, any function e can be used as a weighting function. Practically, e is commonly implemented by a neural network layer.

The described mechanism is differentiable and its gradient is not prone to vanishing. Therefore, attention mechanisms are easy to add to existing network architectures.

2.2.4 *Architectures*

Neural networks establish a mathematical framework that allows for the design of highly complex models based on a collection of layers. Given the well-established

building blocks introduced in the previous chapters, designing a network for a problem poses a tremendous optimization problem. The influences of different structural choices, such as an increase in depth or the use of a pooling layer or attention mechanism, are not well understood and their effectiveness is usually proven empirically. This results in a severe problem for neural architecture search [EMH19; Ren+22]. While optimization methods are suitable to tackle the general architecture search problem, a main challenge is that evaluating a given architecture requires training a potentially very large network and evaluating it on a test dataset. In many scenarios, the approach of optimizing the network architecture for the designated problem at hand is infeasible. Instead, the common practice is to adapt established architectures that have been proven successful in related problems. For many years, the ImageNet challenge has been a driving force in this regard for the computer vision community [Rus+15b]. The challenge defines an image classification task and a corresponding image dataset. This has fostered the development of methods and also allowed for a direct comparison of different approaches and architectures. Many architectures in computer vision are at least inspired by networks that have been successful in the classification challenge. Often, parts of common networks are used and the weights derived from training on the ImageNet dataset serve as an initialization [Stu+19].

One of the first popular network architectures actually predates the ImageNet challenge. The LeNet was proposed as a classification model for the handwritten digit benchmark MNIST in [LeC+98]. The network is built upon three convolutional layers of size 5×5, two average pooling layers and two fully-connected layers. All layers are followed by tanh activations except the softmax activated output layer. As the LeNet significantly outperformed all other classification models based on the respective benchmark, the work of LeCun et al. showed the effectiveness of convolutions in neural network for computer vision tasks. The overall network contained about $60,000$ parameters which is smaller by magnitudes than recent architectures. Nonetheless, the LeNet architecture is still popular for comparisons and proofs of concepts in research.

The first neural network to win the ImagNet challenge was proposed in 2012 [KSH12]. The AlexNet, also known as *Super Vision*, outperformed the other competing methods by such a margin that it significantly contributed to making CNNs and deep learning the dominant approach in computer vision. Compared to the LeNet, the AlexNet follows a similar design strategy but scales the number of layers and, thereby, parameters. It employs five convolutional, three pooling and three fully-connected layers. The original network was comprised of two separate CNNs that were interconnected at different points. This concept of two parallel branches that later fuse their representations is called a two-stream network [FPZ16]. Using two streams was mainly motivated by computational requirements and was later often omitted. The large success of the AlexNet can be mainly explained by its large number of parameters (60 million), combined with the availability of a large annotated dataset.

The availability of large datasets and the increase in computational resources motivated the exploration of deeper network architectures. Scaling CNNs in size and especially in the number of layers resulted in a steady improvement in performance. The networks in this thesis, as well as many others, rely on the famous VGG networks as introduced in [SZ15]. Therein, the *VGG-16* and the *VGG-19* were proposed, which have 16 and 19 layers, respectively. The VGG nets organize their convolutional

layers in blocks that are seperated by pooling layers. Earlier networks like the LeNet or Alexnet used large filter masks in the beginning of the network. The increased receptive field was supposedly beneficial to performances [ZF14]. The VGG networks did not follow this design principle and the authors argued that stacking multile 3×3 kernels serve the same purpose. By using convolutions with a kernel size of 3×3, a stride of one, and zero padding throughout the entire network, the size of the feature maps only changes at the respective pooling layers, (see Equation 13). This design principle can also be found in later networks.

In [SGS15] and [He+16], it was shown that simply increasing the number of layers beyond the size of a VGG-19 network leads to saturation and later to a decrease in performance. It is argued that this is not an effect of overftting due to the high number of parameters. In order to overcome this limitation, He et al. proposed the concept of residual learning [He+16]. One or multiple layers of a traditional feedforward network can be summarized by a function H that constitutes a non-linear mapping of the input data $\mathbf{x}$. In this case, the network layers are considered to be an approximation of a target function $F(\mathbf{x})$. In the case of residual learning, the goal is to not to approximate the goal function $F(\mathbf{x})$ directly, but to approximate the residual function $F(\mathbf{x}) - \mathbf{x}$. With $\mathbf{x}_\ell$ being the output of layer ℓ, we can summarize this principle by

$$\mathbf{x}_\ell = H(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1} \approx F(\mathbf{x}_{\ell-1}), \tag{28}$$

with

$$H(\mathbf{x}_{\ell-1}) \approx F(\mathbf{x}_{\ell-1}) - \mathbf{x}_{\ell-1}. \tag{29}$$

On an architectural level, residual learning can be easily implemented by adding identity connections to a network that simply bypass one or more layers. In [He+16] several architectures were presented with high layer counts ranging up to 152 layers. The additional skip connections that were introduced by so-called residual blocks were a main reason why networks with such high layer counts could be trained efficiently. The different ResNet variants outperformed the state of the art and became one of the most popular architectures in computer vision. It is argued that the skip connections result in a better preconditioning of the network as they facilitates the learning of identity mappings at different layers. Skip connections have become a very common design principle and have also been used in other CNN architectures [SGS15; Hua+17].

In sum, a wide variety of different convolutional architectures exist. For many applications, relying on a proven convolutional backbone architecture such as a VGG or ResNet variant has been a successful strategy. Commonly, the CNN realizes the feature extraction part of the traditional computer vision pipeline and is then followed by more task-specific mechanisms and layers.

2.3 TRAINING

As described in the previous sections, a diverse set of layers, mechanisms and architectures exist to build neural networks for varying applications. After defining the architecture and the respective configuration of a network, the resulting model can, essentially, be considered a large function approximator with potentially billions of weights θ. The process of finding an optimal set of weights for a given problem is called *training*. The most common principle is to learn an optimal set of

weights from annotated exemplary data. Therefore, an annotated training dataset $\mathbf{D} = \{(\mathbf{x}^{(0)}, y_0), \ldots, (\mathbf{x}^{(N)}, y_N)\}$ with datapoints $\mathbf{x}^{(i)}$ and class labels y_i needs to be created manually. Given the labeled data, optimization with *Gradient Descent* can be used to optimize the set of weights with respect to a predefined goal function. This principle is known as *supervised learning*. A major drawback of supervised learning is the creation of the representative, labeled training dataset $\mathbf{D}$. To alleviate the training data problem, various principles exist to include different data sources into the training procedure.

2.3.1 *Gradient Descent*

The main idea behind gradient-based optimization is to use the gradient of a function and follow its direction up to an optimum. This is performed by iteratively taking steps in the direction of the steepest descent, which is known to converge to a local minimum [Cur44]. Let $\mathcal{L}$ be a differientiable function, with respect to a set of parameter θ. The general idea of gradient descent can be summarized by the so called *update rule* that describes the iterative weight updates with respect to the gradient of a function:

$$\theta_{i+1} \leftarrow \theta_i - \eta \cdot \nabla_\theta \mathcal{L}. \tag{30}$$

In this case, $\eta \in \mathbb{R}$ is a hyperparameter that defines the size of the steps taken in the direction of the gradient. Commonly, η is a small positive number. Using gradient descent to find optimal weights for a neural network poses two main problems. First, the so-called *loss function* $\mathcal{L}$ needs to be defined in a way that it measures the quality of how well the network solves the desired task. Furthermore, the gradient with respect to each weight is needed, which requires a special algorithm for neural networks with a high number of weights, several layers and potentially recurrent components.

Loss Function

The goal of gradient descent is to adjust the weights of a neural network in order to minimize a goal function [GBC16, p. 79]. A classic application of neural networks is, for example, the approximation of a classification function. In this case, the output layer usually has the same number of neurons as possible classes. The activation of a neuron can then be considered an estimate for the probability of data point $\mathbf{x}$ belonging to class y, with $\hat{y} = p(y|\mathbf{x}, \theta)$. Let $\mathbf{D} = \{(\mathbf{x}^{(0)}, \mathbf{y}_0), \ldots, (\mathbf{x}^{(N)}, \mathbf{y}_N)\}$ be an annotated training set with N samples. A common way to represent class information is to use one-hot encoded vectors $\mathbf{y}$, which are equal to one at the position that correspond to the respective class and zero everywhere else. In order to train a network with gradient descent, a loss function is required that quantifies the difference between the network prediction $\hat{\mathbf{y}}$ and the label represented as a one-hot encoded vector $\mathbf{y}$. Such a loss function is, for example, given by the L2-loss, which is based on the L2-norm between network output and annotation:

$$\mathcal{L}(\theta, \mathbf{D}) = \frac{1}{|\mathbf{D}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathbf{D}} \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2. \tag{31}$$

The L2-loss measures the divergence between network output and label and can be used as a goal function for optimizing classification or regression networks. In

classification, the *Cross-Entropy-Loss* became more popular as it directly models the symbolic characteristics of distinct classes. Essentially, the pseudo-probability of a class predicted by the network neuron is compared to whether this neuron is supposed to be active ($y_i = 1$) or not ($y_i = 0$). Let M be the number of classes while y_c is an indicator of whether the desired output of the network is one or zero. The Cross-Entropy-Loss over the training set $\mathbf{D}$ is then defined as:

$$\mathcal{L}(\theta, \mathbf{D}) = -\frac{1}{|\mathbf{D}|} \sum_{(\mathbf{x},\mathbf{y})\in\mathbf{D}} \sum_{c=1}^{M} y_c \log(\hat{y}_c). \tag{32}$$

In the aforementioned formulations, the overall loss is calculated over the entire training dataset $\mathbf{D}$. This means the network predictions need to be calculated for all samples in the training set before a weight update takes place. The computation of the gradient with respect to all available samples poses several practical issues, especially as training datasets tend to be very large. Alternatively, weight updates could be performed after each individual sample. As the algorithm draws samples randomly from the training set, this method is known as *stochastic gradient descent* [GBC16, p. 286]. Due to the consideration of only one single sample at a time, the respective gradient tends to be noisy which results in a high variance of the optimization procedure. In order to compromise between computation speed and optimization behavior, *batch stochastic gradient descent* is the most commonly used training approach for neural networks. In this case,the gradient is computed over a small number of samples, which is called a batch. The classical gradient descent method is often extended by an added momentum or an adapted learning rate for specific weights. Prominent methods to this end are AdaGrad [DHS11], RMSProp [TH12] or ADAM [KB15].

Error Back Propagation

After defining a loss function for optimizing a neural network with gradient descent, the question remains how to compute the gradients with respect to the networks weights. Efficiently computing the respective gradients and training a network is still possible using an algorithm known as *Error Back Propagation* that was proposed in [RHW86]. In order to understand the basic principle behind the back propagation algorithm, consider a fully-connected multilayer perceptron. Each layer l contains a number of neurons with weights $w_{jk}^{(l)}$ connecting neuron j of the previous layer and neuron k of layer l. The bias of each neuron is given by $b_j^{(l)}$. The main goal of back propagation is to compute the partial derivatives $\frac{\partial \mathcal{L}}{\partial w_{jk}^{(l)}}$ and $\frac{\partial \mathcal{L}}{b_j^{(l)}}$.

Essentially, back propagation includes three main steps. First, the considered sample is propagated through the network and the respective output is computed. This step is also known as the forward pass. Next, the output error is computed by a previously defined loss function $\mathcal{L}$. The main idea of back propagation is then to define the error of a specific neuron and link it to the errors of the succeeding layer. This way the error given by the loss function can be propagated through the entire network starting from the output layer. By linking the local error of a neuron to its gradient, the required partial derivatives of the loss function with respect to each weight can be computed.

The entire algorithm evolves around four main equations [Nie15]. Let δ_j^l define the local error of neuron j in layer l. The first equation describes the error in the output layer L with z^L being the input of the activation functions of the output layer and $\nabla_a \mathcal{L}$ being the gradient of the loss function with respect to the outputs after computing the respective activation function:

$$\delta^L = \nabla_a \mathcal{L} \odot \phi'(z^L). \tag{33}$$

After computing the local errors δ_j^L of the output layer, the next step of the backpropagation algorithm defines the computation of the local errors based on the local errors of the succeeding layer:

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \phi'(z^l). \tag{34}$$

Therefore, the local errors of layer l are directly linked to the local errors of layer $l+1$. The computation can be interpreted as moving the error backwards through the network by multiplying the local errors of layer $l+1$ with the transposed weight matrix $(w^{l+1})^T$. The Hadamard product with the derivation of the activation function $\phi'(y^l)$ can then be considered as the inversion of the application of the activation function. By computing the errors layer by layer, the local error of each neuron in the network can be derived.

Finally, it is straight forward to define the partial derivatives based on the local errors which allows the weights to be updated accordingly:

$$\frac{\partial \mathcal{L}}{b_j^{(l)}} = \delta_j^l, \tag{35}$$

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{(l)}} = a_k^{l-1} \delta_j^l. \tag{36}$$

Usually, the gradients are averaged over multiple samples, which smooths the optimization behavior. This is feasible if the overall loss is derived from the simple summation of the loss of multiple samples, which then directly translates to the gradient computation.

The concept of backpropagation can be easily generalized for more complex layers such as convolutional layers. In the case of recurrent components such as LSTMs or GRUs, an extended algorithm called *Backpropagation Through Time* can be applied [Moz89; RF87]. Conceptually, the algorithm then considers an unfolded representation of the RNN as depicted in Figure 7 and follows the standard backpropagation approach. As the same parameter then occurs multiple times in the unfolded network, the respective gradient is accumulated. This constitutes a form of weight tying.

2.3.2 *Levels of Supervision*

The traditional approach to derive a learning-based model relies on sample data. After selecting and defining the classifier structure, a dataset needs to be gathered that is of sufficient size and representative of the problem at hand. The dataset is then manually annotated and commonly split into a training, validation and test set. The training set is used to optimize the classifier's weights, for example, by using gradient descent. Hyperparameter choices can be evaluated on the validation

set, while the test set serves as the final evaluation of the method on supposedly unseen data. This entire process is called *supervised learning* as the learning process is steered by the manually created annotations [GBC16, p. 102].

On the other hand, *unsupervised learning* can be used to derive meaningful representations from data [Mur22, p. 14]. This category of learning algorithms includes, for example, clustering approaches that group data points. As this process does not rely on any manually created annotations, usually no semantic meaningful labels can be derived.

Another paradigm that is closely linked to unsupervised learning is the idea of *self-supervised learning* [Mur22, p. 630]. Similar to unsupervised learning no manually created annotations are required. Algorithms commonly rely on proxy tasks or augmentation strategies to create artificial labels as training targets. Training against these artificial targets by using, for example, contrastive approaches allow for learning powerful representations from large amounts of unlabeled data [Jai+20]. Nowadays, self-supervised learning has become a common strategy to pretrain neural networks.

The creation of manual annotations is usually a cumbersome and costly process. Therefore, several learning paradigms try to reduce the labeling effort. *Weakly-supervised learning* reduces the degree of detail in the requested annotation. An example for this paradigm is trying to train an image segmentation network solely based on image level object labels.

A very popular branch of machine learning aims at combining learning from labeled and unlabeled data. This is known as *semi-supervised learning* [CSZ06]. Usually it is assumed that a large unlabeled dataset and a small number of annotations are available. The underlying self-training method in this work is a common technique in semi-supervised learning. In this case, a model is trained on the labeled portion of the available data and is then improved by training on pseudo-labels that are generated for the unlabeled data. For a detailed introduction to self-training see Chapter 4.

While all aforementioned learning paradigms assume that the collected data is representative of the considered problem, the idea of *transfer learning* does not make this assumption. Here, the model is trained on data for one task in order to boost performance with respect to a related task.

3 HANDWRITTEN DOCUMENT ANALYSIS

After establishing the fundamentals of deep learning methodology for computer vision and pattern recognition tasks in Chapter 2, this chapter will present an overview of their applications in the field of document analysis. In this regard, a clear focus lies on offline handwriting recognition as well as word spotting, as these tasks represent the applications of interest with respect to this thesis.

Document analysis has a long-standing tradition in the pattern recognition community and has been an active field of research. Documents in general pose a number of challenges with respect to retrieval and recognition systems. Their complexity emerges from the fact that various factors influence the appearance of a document image. Furthermore, the purpose of a document is usually to carry information in an efficient and compressed way, making them a quite information rich domain.

In order to extract information from a document, several problems have to be solved. Traditionally, this can be broken down into a series of processing steps. Layout analysis tries to identify image regions, such as text areas, and allows them to be split into meaningful entities. Considering a classical text recognition pipeline, systems often first try to segment the written text into text lines or individual words. This can be considered a special case of layout analysis, producing distinct image regions that can then be processed by a recognition or retrieval system. The segmentation problem illustrates well the complexity of the domain. Whereas from a human perspective breaking down a text line into individual words while reading is easily performed due to the exploitation of context and language information, for an automatic system this is a demanding task. This can be summarized with Sayre's paradox [Say73]. Producing a word segmentation is usually infeasible without context information that requires a recognition step. Conversely, recognition traditionally requires a segmentation step. Overcoming this dilemma in combination with the huge variability of complex layouts is usually the first step in a document image analysis pipeline.

In this work, the focus lies on the recognition and retrieval part of the processing pipeline. Given a solution to the word segmentation problem, recognition and retrieval can be considered special forms of information extraction. While for printed text the recognition problem is largely considered to be solved, handwriting recognition is still an open problem. Due to the large variability across and inside of writing styles, error rates are still not satisfactory, especially in the absence of representative training data. Section 3.1 presents an overview of the field of handwriting recognition.

In parallel to solving the recognition problem, retrieval has received some strong interest from the research community. Retrieving a query word from a document collection is commonly known as word spotting. In contrast to recognition, the user is not presented with a fixed, potentially wrong result but a ranked list of similar image regions. This arguably simplifies the problem which allows for robust and interpretable results for challenging datasets where recognition fails. Section 3.2 introduces the essential taxonomy and established techniques for word spotting.

3.1 HANDWRITING RECOGNITION

The field of Handwritten Text Recognition (HTR) has seen an evolution of methodology similar to the general field of pattern recognition. As discussed in Section 3.1.1, early methods relied on designed feature representations usually combined with Hidden Markov Models (HMMs). While HMMs still include structural modeling, the uptick in deep neural networks has led to the use of more data driven approaches, which only account for the sequential nature of the domain. Section 3.1.2 reviews one of the most influential works for text recognition, as Connectionist Temporal Classification (CTC) proposes the use of a designated loss function allowing for the efficient training recurrent neural networks. Inspired by works from machine translation, several works explore the direct mapping from one sequence to another using recurrent neural networks which include an attention mechanism. These so-called sequence-to-sequence models (see Section 3.1.3) offer a competitive alternative to CTC and build the basis for the recognition model used in this thesis. Finally, Section 3.1.4 discusses some recent trends in the field of text recognition, with models successfully using transformer architectures and eliminating the need for explicit text line segmentation.

3.1.1 *Early Methods*

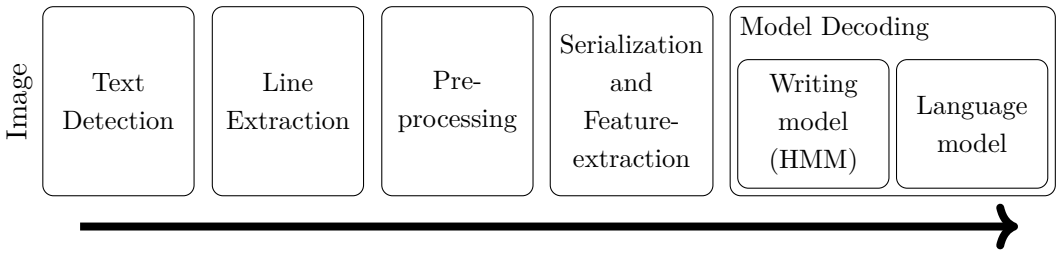


Figure 11: Structure of a classical handwriting recognition system based on HMMs. The image is traditionally transformed into a sequence of feature vectors extracted from preprocessed textlines. Decoding is then performed by the explicit combination of a writing and language model.

In order to solve the problem of HTR, methods need to model a function which maps a text image to a sequence of characters. Motivated by their successful application in the domain of speech recognition [You96], HMMs were among the first models to solve the recognition problem from a statistical perspective [PF09]. One of the appealing benefits of HMMs is that they are capable of jointly solving the segmentation and recognition problem. Generally, HMMs operate on input sequences which made them a natural choice for speech recognition where the input signal is inherently a sequence. Applying the same methodology to HTR requires the serialization of the input image.

Figure 11 summarizes the traditional processing pipeline that was state-of-the art before neural networks became the model of choice. First, the text area is detected and segmented into individual text lines [LZT07]. Following the established pattern recognition approach, preprocessing is performed to remove unwanted variance from

the data. In the context of HTR with HMMs, preprocessing typically considers the normalization of the baseline orientation, the slant angle and the size of the handwriting. Additionally, some systems perform a binarization step.

In order to apply an HMM based system, the input image needs to be transformed into a sequence. Therefore, a serialization step is performed. The most popular approach relies on a sliding-window, where at each window position a feature vector is extracted from the local image region [Kal+93; Sch+96]. For feature extraction, a vast amount of different approaches have been investigated covering both analytic as well as heuristic feature extraction [PF09]. Features either based on the statistics of pixel intensities or geometrical properties have been proven to be effective for an HMM based recognizer. After serialization, the input image is transformed into a sequence of feature vectors allowing modeling and prediction with HMMs.

Decoding the feature vector sequence $\boldsymbol{X}$ to a sequence of symbols $\boldsymbol{w}$ makes use of two distinct models. On one hand an HMM constitutes the writing model that describes the realization $P(\boldsymbol{X}|\boldsymbol{w})$ in terms of the observed feature vector sequence given, for example, a sequence of characters. Additionally, languages are defined by distinctive rules introducing a number of constraints on possible symbol sequences. This is commonly modeled by a languange model describing the probability $P(\boldsymbol{w})$.

The application of HMMs poses three essential problems [Rab89]. Given a model λ and a sequence of observed feature vectors $\boldsymbol{X}$, the *evaluation problem* refers to computing the probability $P(\boldsymbol{X}|\lambda)$, which is the probability that the model has generated the given sequence of feature vectors. An HMM consists of a set of hidden states $\mathbf{S}$ that generate state-specific outputs following the output probability distributions $p(\boldsymbol{x}|s_t = j)$. Finding the state sequence $s_1, \dots s_T$ that explains the observation sequence $\boldsymbol{X}$ is commonly known as the *decoding problem*. Formally, an HMM is defined as $\lambda = (\pi, \boldsymbol{A}, \boldsymbol{B})$ with π and $\boldsymbol{A}$ being the start and state transition probabilities, respectively, and $\boldsymbol{B}$ being the state-specific output probability distributions. Given the model λ and a set of annotated samples, the *training problem* is posed by finding a set of parameters that maximizes the probability of observing the training data. Over the years, algorithms such as the Viterbi-Algorithm [Vit67] for decoding or Baum-Welch-Algorithm [BP66] for training have established themselves and allowed for efficient solutions to the previously described problems [Fin14].

Performing recognition with HMMs boils down to finding the symbol sequence $\hat{\boldsymbol{w}}$ that maximizes the posterior probability of $P(\boldsymbol{w}|\boldsymbol{X})$, given the observed feature vector sequence. Using Bayes' rule, this can be reformulated as follows

$$\begin{aligned}\hat{\boldsymbol{w}} &= \operatorname*{argmax}_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{X}) = \operatorname*{argmax}_{\boldsymbol{w}} \frac{P(\boldsymbol{w})P(\boldsymbol{X}|\boldsymbol{w})}{P(\boldsymbol{X})} \\ &= \operatorname*{argmax}_{\boldsymbol{w}} P(\boldsymbol{w})P(\boldsymbol{X}|\boldsymbol{w}).\end{aligned} \tag{37}$$

Rewriting the posterior probability again illustrates the role of the writing $P(\boldsymbol{X}|\boldsymbol{w})$ and the language model $P(\boldsymbol{w})$.

The recognition problem is commonly solved by modeling each character as a single HMM. Therefore, the recognition problem can be solved by infering the sequence of hidden states $\boldsymbol{s}^*$ with maximal production probability $P(\boldsymbol{X}, \boldsymbol{s}|\lambda)$ given the model λ:

$$\boldsymbol{s}^* = \operatorname*{argmax}_{\boldsymbol{s}} P(\boldsymbol{X}, \boldsymbol{s}|\lambda). \tag{38}$$

Solving the optimization problem, for example with Viterbi decoding, yields a sequence of states usually associated with characters or blank symbols for a given sequence of feature vectors. This has the benefit of jointly providing a solution to both the recognition as well as segmentation problem.

With the availability of a mature methodology, HMMs rapidly became the state-of-the-art in HTR and several systems emerged in the early 2000s [Nat+01; MB01; WFS03]. All of these systems follow the previously described general recognition process with HMMs and only differ in the choice of features, model architecture, topologies and ouput modeling. Severe disadvantages of the classic HMM-based recognition process are the handcrafted feature representation and that, by applying Baum-Welch training, the model is optimized in a non-discriminative way [SR06].

Therefore, parameters of an HMM are only optimized with respect to the corresponding class and are trained independently. Several works aim at overcoming these limitation by combining HMMs with neural networks. The literature distinguishes hybrid [BM94] and tandem combinations [HES00]. For hybrid HMMs, the goal is to replace the output model with a neural network. This is achieved by training a neural network to model the posterior state probability $p(\boldsymbol{s}|\boldsymbol{X})$. The observation probability $p(\boldsymbol{X}|\boldsymbol{s})$ is then approximated by

$$p(\boldsymbol{X}) \approx \frac{p(\boldsymbol{s}|\boldsymbol{X})}{p(\boldsymbol{s})}, \tag{39}$$

with $p(\boldsymbol{s})$ being the prior state probability [BM94].

Tandem approaches also estimate the posterior probability $p(\boldsymbol{s}|\boldsymbol{X})$ using a neural network and use the outputs as features for the standard HMM procedure. This is usually combined with a Principal Component Analysis (PCA) on the logarithmized outputs [SH02].

Before HMM-based systems were largely replaced by models exclusively relying on neural networks, they dominated the field of HTR. High performances were achieved by pure HMM-based systems that relied on Bernoulli distributions for modeling emission probabilities [Gim+14]. Nonetheless, a clear trend towards combined systems and especially tandem systems could be observed [Boq+11; BNK; KDN13].

3.1.2 *Connectionist Temporal Classification*

Due to their success in a wide range of computer vision tasks, neural networks became the natural competitor to HMM based systems. One of the first works, proposing Convolutional Neural Networks (CNNs) for an image recognition task was conducted by LeCun et al. [LeC+98]. The task at hand was handwritten digit classification that was benchmarked with the MNIST dataset, essentially constituting an HTR problem. The success of the method and the large performance gains, heavily influenced neural network research not just for HTR but for recognition problems in general. In this case the HTR task is a simple classification task mapping an image to one out of ten classes. Generalizing this, for example, to word recognition, yields two fundamental problems. A previous word segmentation is required to follow the holistic approach and, due to the large size of vocabularies, an infeasible large number of classes needs to be considered. Several works tried to circumvent the problem of too many output classes, by following nearest neighbour approaches. This idea

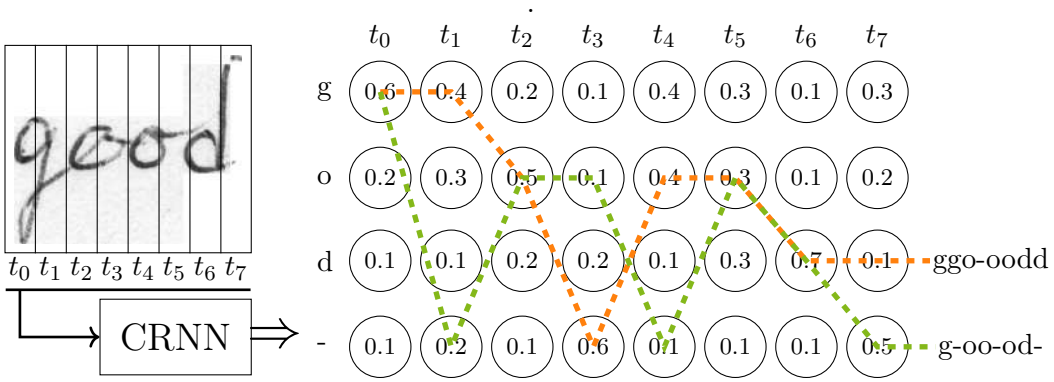


Figure 12: Computation of the CTC-loss for a given input image. A CRNN generates predictions for a sequence of feature vectors t_i. The CTC-loss describes the pseudo-probability of a correct prediction. In the given example, two exemplary paths that result in the final correct prediction of the word "good" are visualized.

predates the application of neural networks and conceptionally tries to map strings and images into a common embedding space [Alm+14]. Mapping images and strings into a common embedding space allows to compute a similarity measure based on common vector distances. Given a lexicon, recognition is then performed by a nearest neighbour search between the predicted embedding for a word image and the embeddings derived from a lexicon. While [Alm+14] learns the mapping from word image to embedding with Support Vector Machines (SVMs), they were later replaced with more effective CNNs [KDJ16; PW16]. With the limitation of requiring a word segmentation and a predefined lexicon, nearest neighbour approaches achieve low error rates and are relevant up to today [KDJ23].

Compared to HMMs these holistic approaches do not model any sequential dependencies. Therefore, neural networks only started to actually replace HMM systems with the introduction of recurrent components such as Long Short-Term Memorys (LSTMs) [HS97] or Gated Recurrent Units (GRUs) [Cho+14]. HMM based recognizer consider a first-order Markov chain, which limits the influence of sequential dependencies. Recurrent Neural Networks (RNNs) do have a higher modeling capacity with regard to long term dependencies. The problem that at first hindered using RNNs for HTR was that, for training the network, individual labels for each time step were required. This results in training a series of independent classifications and also introduces the need for a segmentation on character level of the training data.

In [Gra+06], the concept of CTC was introduced overcoming this limitation and effectively enabling RNNs to replace HMMs for HTR. CTC offers solutions to two main problems. First, a differentiable loss function is defined that allows to train a network with a single label per image. Second, it provides a solution to the decoding problem giving an effective way of predicting a sequence of characters for an input image. CTC is commonly used with neural networks combining convolutional and recurrent layers. The convolutional part of the network operates as a feature extractor producing a set of feature maps for the given input image. By considering the resulting feature maps as a sequence of feature vectors, recurrent components allow for sequential modeling. Finally, the output layers predict a pseudo-probability

distribution over the set of characters given the feature vector at each time step. In order to avoid a predefined alignment between character and feature vector sequence, CTC introduces an additional symbol. This blank symbol (-), not to be mistaken with a white space, allows the network to predict no symbol for a given time step.

See Figure 12, for an exemplary input image with only three possible character outputs and the added blank symbol. At each time step the network predicts pseudo-probabilities for each character given the feature vector. Usually, the feature vector sequence is longer than the number of characters, resulting in a high number of duplicate predictions. Decoding is then performed by finding the path with the highest overvall probability, computed from the product of pseudo-probabilities at each time step. A simple approximation is possible, for example, by using best path decoding, where at each time step the character with highest pseudo-probability is considered (see orange path in Figure 12). The final prediction sequence is obtained by first removing all repeating characters, followed by removing the blank symbols. By introducing the blank symbol, the prediction of repeating characters is feasible.

For training a network with the standard optimization approaches such as gradient descent, a differentiable objective function is required. Given an input sequence $\boldsymbol{X}$, CTC defines the objective as the maximization of the conditional probability $p(\boldsymbol{w}|\boldsymbol{X})$, which is the overall probability of the network predicting the correct character sequence $\boldsymbol{w}$ for a given training sample. Due to the potential prediction of blank and repeating symbols, a high number paths exist that result in a correct prediction sequence. The overall probability results from the sum over all paths resulting in the correct prediction. Considering all potentially correct paths appears computationally expensive, but can be solved efficiently by an adopted version of the Forward-Backward algorithm known from HMMs [Rab89].

The target sequence is modified by adding a blank symbol to the beginning, the end and in between each duplicate character. Computing the conditional probability $p(\boldsymbol{w}|\boldsymbol{X})$ can be performed recursively with dynamic programming. The core idea is that at each time step only those paths need to be considered that correspond to a correct prefix of the final sequence. The goal of training is to maximize the log probabilities for all training samples. As this function can be differentiated, it can be used as a loss function to optimize the parameters of the convolutional RNN.

As shown in [Gra+09], significant performance gains for HTR result from the application of CTC. The approach rapidly became the state-of-the art and outperformed HMM based systems. Methods benefited from the developments in the general research on neural networks and efficient architectures and proven techniques such as dropout [Pha+14] or data augmentation [Wig+17] were rapidly adopted. Further improvements were achieved by pretraining with synthetic data [Dut+18] and multi-task learning [Che+17a].

[GS08] propose replacing the classic LSTM layers with multi-dimensional LSTMs. In the two-dimensional case such as offline handwriting recognition, the multi-dimensional recurrent component considers dependencies in either of the two spatial dimensions. The technique established itself and was part of most systems reporting highest performances in several handwriting recognition competitions [Ton+14; Bru+14; Sán+16]. Despite their successful application, it remains an open discussion whether multi-dimensional LSTMs are actually required [Pui17; MM19], especially with respect to their high computational demand.

3.1.3 *Sequence-to-Sequence*

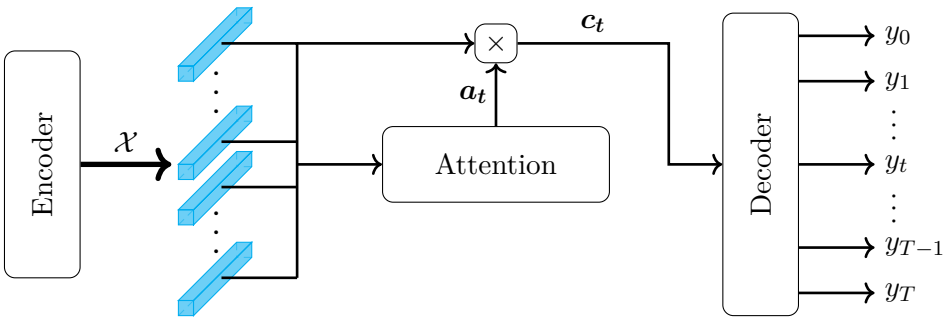


Figure 13: Structure of an encoder-decoder model for handwriting recognition. The encoder povides a sequence of feature vectors. The attention module provides a context vector $\boldsymbol{c_t}$ by attending to different feature vectors. Finally, the recurrent decoder generates a sequence of character predictions y_i.

Even though offline HTR operates on a single input image, most methods traditionally account for the sequential nature of how the data has been produced. Transforming the input image to a sequence has been an established technique, for example, using sliding window approaches in combination with HMMs (see Section 3.1.1) or CNN based feature extraction with slicing for RNNs (see Section 3.1.2). Considering a previous transformation to a sequence, the problem of HTR consists of mapping the sequence of features to a sequence of characters. In most cases, the two sequences are of different length.

Learning the mapping from one sequence to another is a reappearing problem for a broad range of applications besides HTR. In machine translation, a sequence of words needs to be mapped to a sequence of words in another language [BCB15; SVL14]. Image captioning can be considered to have a similar structure. By extracting a set of feature vectors, generating a caption boils down to mapping the feature vectors to a sequence of descriptive words [Xu+15]. Another field closely related to HTR is speech recognition. Again, the speech signal is usually transformed into a sequence of feature vectors resulting in the same structural problem of mapping one sequence to another [Bah+16; Cho+15].

Due to the tremendous success of attention-based models in Natural Language Processing (NLP) [Vas+17] and the structural similarities of the sequential problem, all these fields have been successfully addressed with attention-based models [SVL14; Xu+15; Bah+16]. Most of these so called Sequence-to-Sequence (seq2seq) approaches rely on a general encoder-decoder architecture as depicted in Figure 13. In general, an encoder transforms the input into a sequence of feature vectors $\mathcal{X} = \{x_0, \dots, x_M\}$. Depending on the problem, the decoder generates a sequence of predictions $\mathcal{Y} = \{y_0, \dots, y_T\}$. As in most cases input and output sequence are of different size, a direct mapping between feature vector and prediction is not feasible. Therefore, an attention mechanism is introduced. At every decoder step t, a context vector $\boldsymbol{c_t}$ is computed. The context vector is derived from the weighted sum of feature vectors $\mathcal{X}$ and attention weights $\boldsymbol{a_t}$. Weighting the feature vector sequence by attention at

every time step and encoding that in the context vector allows the decoder to focus on different feature vectors at every time step.

Inspired by other fields, seq2seq models were rapidly explored for scene text [Shi+16; Che+17b] and handwriting recognition [BLM17; Sue+18]. In [BLM17], the first attention based model for HTR was proposed. Based on a model relying on multi-dimensional LSTMs, the authors introduce an attention mechanism. Instead of collapsing the feature maps provided by the encoder, applying softmax activation and training based on the CTC loss, the model relies on a multi-dimensional LSTM network to compute a set of attention weights. Finally, a decoder combining an LSTM layer and a Multi Layer Perceptron (MLP) provide the final predictions. The reported results proved that, in general, the seq2seq approach is applicable to HTR. Nonetheless, the proposed system relied on an encoder pretrained with CTC and only performed on par with the equivalent CTC model.

Sueiras et al. [Sue+18], proposed an architecture that became the de-facto standard seq2seq architecture for HTR. In contrast to [BLM17], the encoder does not rely on multi-dimensional LSTMs, but is purely based on a CNN. The input image is transformed to a sequence of patches by a sliding window approach and, for each patch, a LeNet [LeC+98] inspired CNN extracts a feature vector. An additional LSTM layer is applied to gather context information, before the resulting feature vector sequence is decoded by a LSTM based decoder in combination with an attention mechanism as proposed in [Cho+15]. As shown in [CV18], further improvements can be achieved with techniques such as training with focal loss or decoding using beam search.

While Sueiras et al. [Sue+18] explicitly transformed the image into a sequence of patches, Kang et al. [Kan+18] proposed a different approach to sequentializing the input data. A VGG [SZ15] based architecture is used to compute a set of feature maps for a given input image. By slicing and concatenating, a sequence of feature vectors is generated that correspond to respective image areas. Furthermore, the authors experimented with content [BCB15] and location-based [Cho+15] attention. Overall, Kang et al. were the first to report word level recognition results that are competitive to CTC based models. In another work, the same author showed that it is also feasible to integrate a language model in the decoding process [Kan+21].

Michael et al. provide an extensive study on different design choices for seq2seq models for HTR in [Mic+19], especially with respect to various attention mechanisms.

Summarizing, seq2seq approaches consitute a powerful class of models overcoming some of the drawbacks inherent to CTC based methods. Therefore, the recognition model considered in this thesis follows the previously described methodology. For a detailed discusssion of a possible realization of a seq2seq model similar to [Kan+18], see Section 4.2.1.

3.1.4 *Discussion and Recent Developments*

Reviewing the field of HTR shows a clear trend away from heuristic design and modeling towards learning from exemplary data. As discussed in Section 3.1.1, HMMs followed a mainly handcrafted pipeline from segmentation to decoding. Features were designed based on knowledge of the visual characteristics of handwriting. Characters

were explicitly modeled by individual HMMs and additional language models provided general language information. The introduction of CNNs with RNNs, however, allowed for learning feature representations. On an architectural level, the recurrent components explicitly model the sequential nature of handwriting and layers such as bidirectional LSTMs or GRUs model context information for each prediction. By adding attention mechanisms and relying on seq2seq models, the model is capable of learning the context on which to focus. The stepwise removal of designed components and their replacement with trainable neural network components resulted in a steady increase in performance. Nonetheless, with less structural modeling more information hast to be learned from data, increasing the need for manually annotated training data.

Looking at the most recent developments in HTR and computer vision in general, the previously discussed trend seems to continue. Despite being the dominant architecture in almost all areas of computer vision, transformer architectures are gaining more popularity. The idea of a transformer architecture was first proposed in [Vas+17] and originated from the field of NLP. The model itself is entirely based on fully connected feed forward networks in combination with attention layers. In general, the architecture follows a common encoder decoder structure. Due to the removal of all recurrent components, the sequence of input features, as well as the outputs, are enriched by a positional encoding that essentially provides information on the position of an embedding in the sequence. Transformers were rapidly adapted to all kinds of one-dimensional sequence problems such as language understanding [Dev+19] or speech recognition [DXX18]. The success of transformer architectures led to an increasing interest in generalizing the approach to vision problems. For example, [SCX19] and [Lu+21] proposed scene text models which rely on a transformer architecture. In order to adapt the approach, the two-dimensional image needs to be transformed into a one-dimensional sequence. For both works, convolutional layers are employed at the decoding stage for efficient feature extraction. [SCX19] proposes a so-called *Modality Transform Block* that essentially is a CNN. The extracted features are then combined with a positional encoding and later decoded following the transformer approach of [Vas+17]. In [Lu+21], convolutional layers are directly combined with attention mechanisms at the encoding stage.

The first use of a transformer architecture for HTR was proposed in [Kan+22]. Similar to [SCX19], a CNN, in this case a ResNet50, serves as a feature extractor. The extracted feature maps are then reshaped resulting in a sequence of feature vectors, which is then combined with a standard positional encoding. Additionally, the encoder includes several self-attention modules to further refine the extracted representations. The decoder only relies on attention modules to make the entire model recurrence-free.

Transformers have also been applied to other vision problems with [Dos+21] being arguably the most influential in the area of image recognition. The proposal of the Vision Transformer showed that a purely Transformer-based architecture is competitive to CNNs. In order to apply a transformer architecture to image recognition, the authors propose splitting the input image into patches of fixed size. Following the exact same approach as [Vas+17], each patch is embedded in a feature vector by an MLP and then combined with a one-dimensional positional encoding. It was shown that the approach, despite its simplicity, is highly competitive and scalable. Inspired by the Vision Transformer, [Li+23] proposed a text recognizer following

the same ideas. A text line is split into patches followed by an embedding stage. The experimental evaluation in [Li+23] showed that, even without the use of any recurrent or convolutional components, highly competitive results can be achieved.

This increase in performance may be explained by removing the inductive bias that is introduced by the use of convolutional layers. Nonetheless, this comes at the cost of requiring a higher amount of training data as the model is not regularized by these biases introduced in the model design. In order to effectively train the transformer architecture in [Li+23], multiple stages of pretraining are required. Therefore, the model relies heavily on synthetic data and transfer learning to achieve high performances. Transformer architectures constitute a promising alternative to the established models as they benefit from the architectural simplicity. As investigated in [PP23], an application to low resource domains may be feasible with more elaborate techniques to fine-tune large scale models.

Apart from architectural developments towards more general Transformer architectures with less inductive biases, a trend to a less explicit segmentation can be observed. As of today, processing pipelines commonly rely on word- or line-level-segmentations. Most state-of-the-art models consider the segmentation problem independently and solve it with methods based on Fully Convolutional Networks (FCNs) [Ren+18; OSK18; Grü+19]. Recently, several works aim at combining segmentation and an explicit recognition. For example, [Car+19] and [Car+20] use a Region Proposal Network (RPN) to predict word bounding boxes which are then processed by a recognition model. The authors propose optimizing the combined model in an end-to-end fashion using multi-task learning. Instead of predicting bounding boxes for document regions, the prediction of starting coordinates of lines is a viable alternative. This can be performed, for example, by multidimensional LSTMs [MM19], or by a combination of a CNN and a bidirectional LSTM architecture [Wig+18; TW19]. All of these methods have in common that the segmentation step is explicitly modeled as a task which is then optimized with exemplary data.

3.2 WORD SPOTTING

Word spotting describes the task of locating a query word in a large document collection. As discussed in Section 3.1, character and word recognition have a long-standing tradition in computer vision but come with a severe drawback. When exploring a new document collection, representative training data is usually not available and, therefore, recognition performance is poor. A recognition system provides a full, potentially wrong transcription of the document collection, which can lead to unsatisfactory results when used for a pure syntactical word search. In this case, word spotting offers a viable alternative. Instead of generating a full transcription, the user searches for a designated word inside the collection. By providing the query word, the problem is considerably simplified, thus increasing robustness. The word spotting system then retrieves image regions that are similar to the given query. A user can then evaluate the most similar regions with respect to the query. As the provided result is a ranked list of potential occurrences, the final interpretation is on the side of the user, which is often desired in the context of exploring historical document collections.

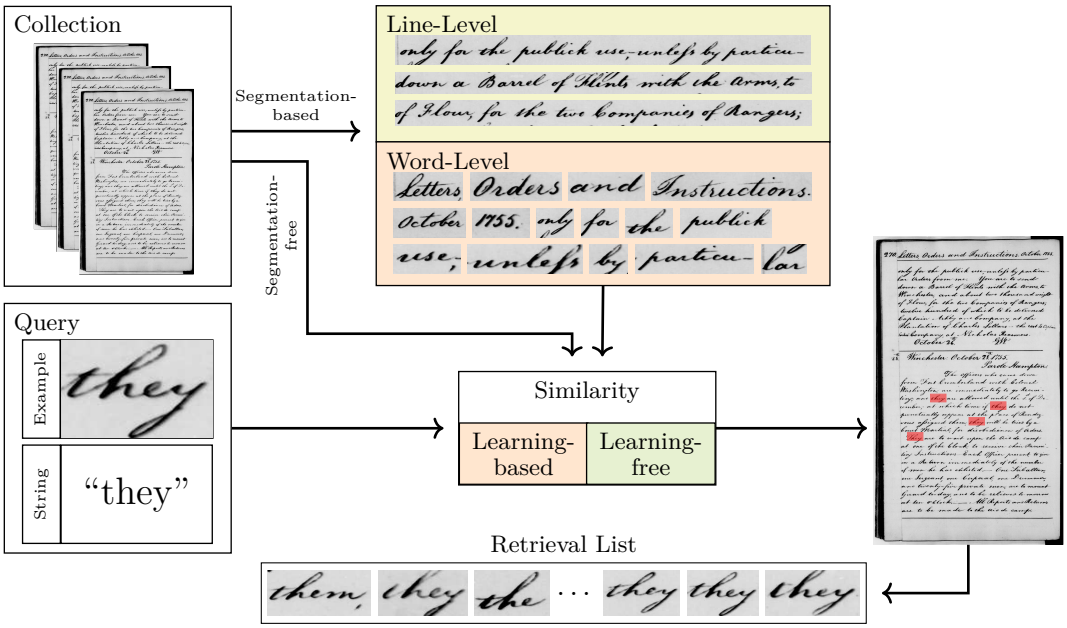


Figure 14: General architecture of a word spotting system. Models are usually distinguished in terms of the query representation (example/string), independent segmentation (segmentation-based/-free) and use of machine learning (learning-based/-free).

The idea of word spotting originated from the domain of speech recognition [Roh+89] and was later adopted to document analysis problems [KH93]. In an influential work [MHR96], Manmatha et al. established the concept of word spotting for handwritten documents. Even though recognition systems have become increasingly more powerful throughout the last decades, the problem of word spotting has received significant attention from the document analysis community and is still an active topic [KDJ23; RSN23]. For an extensive survey on traditional word spotting methodology, see [Gio+17].

In this chapter, the fundamental concepts of word spotting will be established. Traditionally, the classification of spotting systems follows the taxonomy presented in Section 3.2.1. A main consideration when designing a word spotting system is the availability of training data. In the following, the most relevant word spotting methods will be reviewed. Therefore, established methods will be differentiated into *learning-free* (Section 3.2.2) and *learning-based* (Section 3.2.3) approaches according to the common taxonomy. According to the literature, the distinction commonly coincides with the prerequisite of training data. As shown by the method in this thesis, learning does not inherently introduce the need for training data. In the closing section of this chapter (Section 3.2.4), the shortcoming of the current taxonomy is discussed and the use of the alternative term *annotation-free* will be supported.

3.2.1 *Taxonomy*

Due to significant interest within the document-analysis community, a plethora of methods have been developed over the years. Although a diverse set of techniques have been proposed, most methods follow common paradigms leading to quite an es-

tablished taxonomy which distinguishes word spotting systems. The most significant distinction between different approaches can be made based on the query representation, the requirement for an independent segmentation step and the use of machine learning.

Figure 14 presents a conceptionalized visualization of the approach that most word spotting systems follow. The goal is to locate a word in a given document collection. Two main paradigms to represent the query word are predominantly used [Gio+17]. In case of *query-by-example* (QbE), the query is provided as an exemplary image. The availability of an example image allows for a pure visual similarity comparison that is especially beneficial for rather homogeneous document collections. Nonetheless, it comes with a disadvantage for the user. In order to search for a given query word, the user first has to find an example image of the word. In the application scenario, searching for rare words is a common case which makes the provision of an exemplary word image a demanding task. Additional to QbE, *query-by-string* (QbS) is the most common symbolic query representation. Here, the query is represented as a string which avoids the requirement of providing an example. However, using the symbolic representation requires the introduction of a model that goes beyond a pure visual comparison of image regions. Besides QbE and QbS, other query representation where proposed for specific domains such as *query-by-online* for online handwriting [SRF17] or specific scripts such as *query-by-expression* for cuneiform [Rus+20].

Early methods for word spotting started with a preprocessing step that segments the documents into individual words [RM07; KWD14; Alm+14]. The generation of a ranked retrieval list then boils down to the computation of the visual similarity of word images. Methods that require such a segmentation step, which is entirely independent from the retrieval process, have been dubbed as *segmentation-based*, but this does not necessarily correspond to a segmentation into word images. As HMMs and RNNs have been used for word spotting, line segmentation is also a popular preprocesing step [Fri+12; Fis+13; RV13]. Similar to HTR, considering segmentation independently from the retrieval or recognition step may be infeasible and may introduce errors at an early processing step. Therefore, methods that are deemed as *segmentation-free* avoid this explicit segmentation at the preprocessing stage. Several works try to jointly solve the segmentation and retrieval problem by using word hypothesis [Rot+17; RWF21] or detector networks [WLB17].

Given the query representation and a method to extract image regions, the final step in a word spotting system is to compute their similarity. The main distinction that is made in the literature is whether the system applies machine learning techniques or not. *Learning-free* methods typically do not use a trained model thereby avoiding the need for annotated training data. These systems usually rely on using designed feature representations that allow for a similarity comparison based on simple vector distances [Ret+19; VHF19]. See Section 3.2.2 for a detailed review on *learning-free* approaches. *Learning-based* methods exploit the availability of labeled data commonly leading to higher overall performances. Additionally, *learning-free* systems do not offer the capability to perform QbS, as it requires mapping to a symbolic character representation. Section 3.2.3 discusses the most important approaches for learning-based word spotting.

With respect to the literature, the method presented in this thesis illustrates the shortcoming of the current taxonomy. The concept of *learning-free* and *learning based* implies that annotated data is either required to train a model or not. As

shown in Chapter 4 and Chapter 5 this is not necessarily the case. An alternative terminology with respect to data requirements is discussed in Section 3.2.4.

3.2.2 *Learning-free Methods*

Learning-free methods offer the big advantage of not needing any annotated data to tackle the task of word spotting which is desirable in the classical application scenario of a word spotting system. This comes with the disadvantage that usually only QbE is feasible, as without training a model only visual similarity can be compared.

The first approaches that were proposed to solve the problem of word spotting were traditionally *learning-free* and *segmentation-based*. When assuming a given word segmentation, the problem of word spotting boils down to computing a similarity measure between two word images. Therefore, in an early work, [RM07] proposes the use of Dynamic Time Warping (DTW) that established itself as a common similarity measure. In order to use DTW to compare two word images, a feature representation is computed first. Common representations in this regard use column based features such as the word profile [RM07; Abi+12; Mon+13] that lead to a representation of variable length. DTW allows for the computation of a distance between two time series of different lengths [SC78]. The resulting word profiles can be considered to be two time series that may include a temporal shift due to the actual realization of the given word. Simply computing the Euclidean distance between both time series would include the temporal shift in the distance measure. Therefore, DTW aims at matching time indices of the two series and therefore computes the minimal distance over all possible temporal alignments. This can be implemented efficiently via dynamic programming and has been applied to *segmentation-based* word spotting in numerous works [Gio+17]. State-of-the art results for *learning-free* and *segmentation-based* word spotting are still achieved by using a variant of DTW for similarity comparison [Ret+19]. In this case, a zoning scheme is used that allows the main zone of a word image that is then followed by different preprocessing steps to be detected. The word image is split into overlapping patches for which a gradient based feature representation is computed [Ret+16]. A final descriptor is then derived from the coefficients of a Discrete Fourier Transform. The combination of zoning and gradient-based features leads to a sequence of descriptors per word image that are then matched using a variant of DTW that considers the spatial consistency between zones.

While DTW offers the possibility of comparing representations of variable length, many other works propose the use of of fixed-length vector representations [Gio+17]. This simplifies the similarity computation as vector distances such as the Euclidean or Cosine distance can be applied directly. In general, most methods focus on designing feature representations that capture the characteristics of handwriting. Similar to the general field of computer vision, gradient-based features have shown promising performances and several features such as SIFT [Rus+11; Lla+12; Ald+15; Rus+15a], HoG [KWD14; Alm+12; Alm+14] or LBP [KWD14] have been popular for word spotting.

Instead of directly designing a feature representation of fixed length for a word image, extracting local image descriptors and pooling them in a fixed length representation is a well-researched alternative [Ald+15; Rus+15a; Alm+12; Alm+14;

SB12]. One example in this regard is the use of Bag-of-Features (BoF) [Csu+04] representations that were first proposed in the context of object categorization. First, local image descriptors are sampled over a dense grid. This step is performed over the entire document collection capturing the distribution of local image features. By clustering the set of features, typical realizations are represented by the cluster centroids. These so-called visual words constitute a code book of typical features that occur in the given document collection. In order to derive a fixed-length representation for a given word image, the corresponding set of features is extracted and then mapped to the closest visual word. This quantization step allows each word image to be represented by a set of visual words. A global representation can then easily be derived by summarizing the occurring visual words in a histogram with a fixed length equal to the number of codebook entries. BoF representations have been popular for *segmentation-based* [Ald+15; Rus+11; Lla+12] and *segmentation-free* [Dov+13; Rus+15a] word spotting. While BoF is a powerful representation in itself, a drawback of the approach lies in the fact that the spatial information of the local descriptors is neglected. This is especially disadvantageous when applied to word images as characters naturally reoccur and only their spatial distribution differs from word to word. Adding spatial information to BoF representations is feasible by using spatial pyramids [LSP06]. Therefore, an image is split into multiple regions in a pyramidal fashion and BoF histograms are computed for each region individually. A global descriptor is then derived by the concatenation of the individual BoF representations. Introducing Spatial Pyramid Matching (SPM) was proven to also increase word spotting performances [Rus+15a].

An alternative to the concept of BoF, which was established to combine descriptors for word spotting, is the use of Fisher Vectors (FVs) [JH98]. Instead of computing a histogram over quantized descriptors, an FV relies on a Gaussian Mixture Model (GMM) that is fitted to the data. The general idea is to first fit a GMM to the local image descriptors extracted from all documents, thereby minimizing the log-likelihood. The resulting GMM is defined by a fixed number of parameters. In order to derive a global descriptor for a word image, the set of local descriptors is computed. Then, the FV results from the gradient of the log-likelihood function given the word image descriptor and the GMM. In other words, the FV describes how the GMM parameters need to be adjusted to better fit the considered image. As the number of parameters is independent from the number descriptors, a fixed length representation is derived. [PR09] and [Alm+14] use this representation in combination with vector distances to perform QbE word spotting. FVs were also used as an underlying representation for *learning-based* approaches as proposed in [Alm+14]. For a more detailed discussion, see Section 3.2.3.

Most methods for *learning-free* word spotting rely on statistics from gradient-based local image descriptors. As these approaches do not consider the structural properties of handwriting, several works propose the use of graph representations. Handwriting can be considered to have an inherent graph structure that can be reconstructed from an image by heuristic approaches [SFR16]. Mapping word images to a corresponding graph allows for a similarity comparison in the graph domain. Computing the similarity between two graphs is usually done by computing an approximated edit distance. As neither the extraction of a graph structure nor the computation of an edit distance requires labeled data, methods that follow this approach are inherently *learning-free* [RLF15; SFR18a; SFR18b; Ame+19].

In terms of segmentation, many of the aforementioned methods rely on an independent word segmentation followed by feature extraction and similarity computation. Nonetheless, several *learning-free* methods extend this pipeline to a *segmentation-free* manner [Dov+13; KWD14; Alm+14; Rus+11]. The most common approach is to use a sliding window to extract overlapping image regions which can then be compared to a query based on the same feature extraction methods as in *segmentation-based* approaches. Due to the high number of overlapping patches, the approach is generally computationally demanding and requires a non-maximum suppression step.

Considering query modalities, QbS is not usually targeted by *learning-free* methods due to the lack of a trained model that can predict symbolic representations. Only few works try to solve the problem of QbS word spotting in a *learning-free* manner. [BJG08] propose the use of templates. By creating a query word image from the string inputted by the user, the rest of the method can operate following a QbE based approach. In [BLT09], QbS spotting is realized by a specific set of features, so-called word shape codings, that can be directly extracted from strings and images.

In summary, the area of *learning-free* word spotting has a long standing tradition and a set of elaborate methods have evolved. Nonetheless, this family of word spotting techniques comes with several drawbacks such as limited performances when confronted with large variability. Furthermore, without a trained model, QbS is usually not feasible. Despite usually being outperformed by *learning-based* methods (Section 3.2.3), *learning-free* methods are still relevant due to their easy applicability without the requirement of annotated data.

3.2.3 *Learning-based Methods*

While *learning-free* methods are a natural way to tackle the retrieval problem of word spotting, including annotated data and training a model offers several potential benefits. Similar to HTR, early *learning-based* methods for word spotting employed HMMs for model-based similarity comparison [CZF06; RP12; Fis+12; Ros+16]. Most methods for word spotting using HMMs rely on a textline segmentation step. One of the first problems that was addressed with HMMs was *learning-based* QbS word spotting. An influential approach which was later adopted and refined was proposed in [Fis+12]. The goal being to find text lines that include a given query word under the assumption that annotated text lines are available for training a model. In other words, the model needs to quantify the relevance of a text line to the provided query. First, all text lines are preprocessed and a sequence of feature vectors is extracted, commonly by using a sliding window approach. Because HMMs are a generative model, they allow a probability that the given HMM produces a certain feature vector sequence to be computed. In order to use that for word spotting, character HMMs are first trained on the labeled text lines of the training set. Training character HMMs allows for the definition of the so-called *filler* model. The *filler* HMM models the likelihood of the sequence of feature vectors given that they are produced by an arbitrary character sequence. A text line that includes the query word is modeled by a compound HMM that consists of the *filler*, a *space* and the *query* model. The *query* model is built by concatenating the respective character

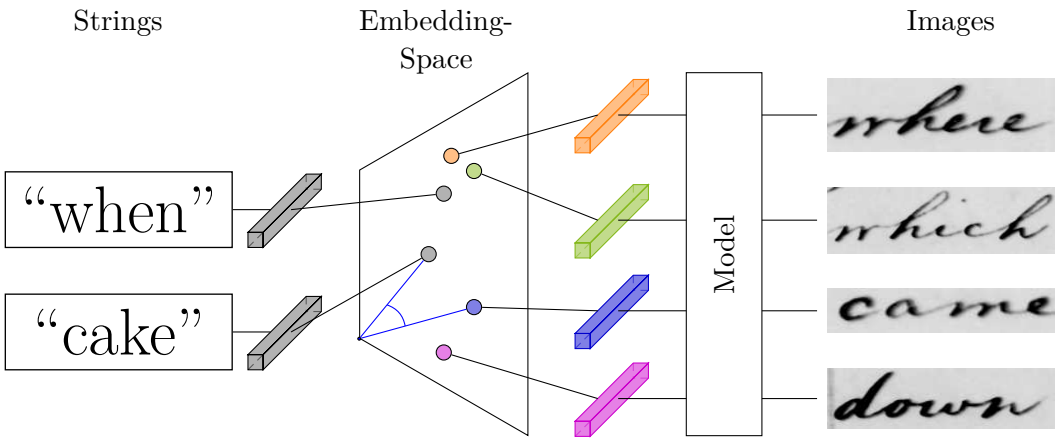


Figure 15: Visualization of the embedded attribute concept. Strings and word images are mapped into a common embedding space, in which similarity can be measured by simple vector distances.

HMMs according to the query string. The *space* model represents whitespaces and is directly derived from a model training on the annotated data. All three models can then be combined in one HMM that represents the sequence of arbitrary characters, whitespace and query word, followed by a sequence of arbitrary characters. If the production probability of the compound HMM is close to the filler HMM, the occurrence of the query word is likely. In practice, the final score computation requires a normalization step with respect to the length of the text line. The final score is given by the quotient of the two production probabilities. Even though no explicit transcription is performed, the final score can be interpreted as the confidence in predicting any transcription opposed to a transcription including the query word [PTV15b]. Later methods improved on the approach by including language models [Fis+13; RPV15; RPV16], lexicon information [PRV14; Ros+16] or by using hybrid models combining HMMs and neural networks [Bid+15; Tho+15].

While HMM-based systems traditionally operate on text lines, other works considered the problem from a holistic perspective assuming a given word-level segmentation. In this regard, SVMs were among the first models to be explored. An SVM constitutes a discriminative model that describes a hyperplane in the feature space. As proposed in [PR09; Alm+12], SVMs can be used for word spotting by training them to discriminate between relevant and non-relevant document image regions. One possible way to train such an exemplary SVM [MGE11] is to generate multiple examples of the query word by just introducing small variations to it. Non-relevant image regions can be extracted by randomly sampling from the documents. After optimization, the distance of a sample to the hyperplane is considered for scoring the retrieval list.

With the uprise of neural networks, a growing interest for applying them to word spotting can be observed. First approaches focused on extracting characteristic word image descriptors that allowed for similarity comparisons based on vector distances. In [KDJ16], the authors propose training a CNN following a traditional classification architecture. The target classes correspond to a lexicon of ten thousand words and the network is optimized following the standard approach using softmax. In order to

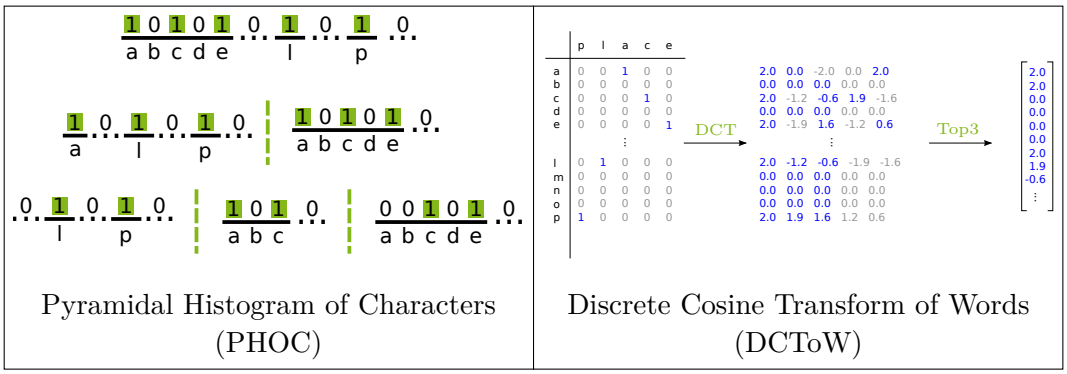


Figure 16: Popular attribute representations for word strings.

use the network for spotting, the output of the penultimate layer is used as a word image descriptor. Similarly, [SRG16] used a pretrained, off-the-shelf CNN [JVZ14] for feature extraction that was later combined with a designated zoning scheme.

Using CNNs for extracting word image descriptors or using exemplary SVMs, does generally not allow for QbS word spotting. An influential concept that integrates QbE and QbS into a single framework is the idea of *embedded attributes* that was first proposed in [Alm+14]. See Figure 15 for a visualization of the general concept. Fundamentally, word images and strings are mapped to vector representations in a shared embedding space. By establishing this transformation, similarities between word images as well as similarities between strings and images can be easily computed via vector distances.

As proposed by [Alm+14], attribute representations are suitable for defining a common embedding space. The concept of attributes originated from the field of object recognition and is of special interest for zero-shot learning [LNH09; Far+09]. An attribute is defined as a semantic property of an object. Usually, it is considered to be binary and shared among different classes. The first attribute representation for words was the Pyramidal Histogram of Characters (PHOC) proposed in [Alm+14], see Figure 16 (left). Here, an attribute is the occurrence of a character in a specific split of a word. Therefore, the word is split in a pyramidal fashion and, for each split, a binary histogram describes character occurrences. The final attribute representation is derived by concatenating the different histograms. Using such an attribute representation based on characters has the benefit that its derivation from a string is trivial. With respect to the embedded attribute approach, mapping word images and strings in the common embedding space boils down to finding a model that is capable of predicting the attribute representation for a word image. [Alm+14] uses a set of SVMs to classify individual attributes as being present or absent.

The SVM-based approach was rapidly replaced with an CNN as an underlying model. In [SF16], a VGG-like architecture was proposed that allowed for the prediction of PHOC vectors. The network is trained in an end-to-end fashion with sigmoid activations. Similar to [SF16], [WB16] aimed at predicting attribute representations with a CNN. Instead of training the network in an end-to-end fashion, they follow a two-stage approach and first train a triplet CNN. A triplet CNN is a network architecture that is based on a three time replication of a network sharing the same weights. During training, the CNN sees two examples of the same class, in this case

two word images of the same word. Additionally, a negative example of a word image showing a different word is provided. By using a contrastive loss function, the network is optimized in such a fashion that the distance between the features predicted for the positive examples is minimized while the distance between a positive and a negative example is maximized. The final network then predicts a discriminative descriptor for word images that can already be used for QbE word spotting. In order to also perform QbS word spotting in the embedded attribute framework, the authors of [WB16] train another MLP to map the intermediate descriptors to a final attribute representation.

While the PHOC representation became the defacto standard for word spotting, other attribute representations were also investigated in [SF17; SF18; WB16]. The Spatial Pyramid of Characters (SPOC) is a PHOC variant with actual character counts replacing the binary occurrence histograms. [WB16] proposes the Discrete Cosine Transform of Words (DCToW), see Figure 16 (right). In this case, the word is represented by a matrix with each row representing a character of the alphabet. Each column corresponds to a character of the respective word. In order to derive the final representation, the discrete cosine transformation is applied to each row and the three highest coefficients of each row are concatenated. As discussed in [SF17; SF18], performances of different attribute representations in combination with various loss functions differ only marginally. Therefore, the PHOC representation has been by far the most popular embedding.

Another line of research that also followed the idea of common subspace representation builds upon the so called HWNet [KDJ16]. Their works relied heavily on using synthetic data (see Section 3.3) in combination with PHOC representations. At first, the HWNet that had been trained as a word classifier was only used as a feature extractor [KDJ16]. To perform QbS word spotting, a set of SVMs was used to predict PHOC attributes. In later works, the approach was refined and the SVM part of the model was replaced by an end-to-end trainable neural network architecture [KJ19; KDJ23]. The general idea is to combine features of different domains in a single embedding. Therefore, a CNN is used as a feature extractor for a real word image and a synthetic word image. During training, the PHOC representation of the annotation is provided. Using an MLP, the features extracted from the real word images and the concatenation of synthetic features and PHOC are mapped into a common embedding space. While most works on holistic *segmentation-based* word spotting exploit attribute representations, [Ret+19] showed that seq2seq-based recognizers are also able to produce discriminative features that can be used for retrieval.

Even though *learning-based* approaches have shown high performances for *segmentation-based* word spotting, the assumption of a word-level segmentation is a severe drawback. As shown in [Rot+17], it is feasible to adapt a *segmentation-based* approach such as an AttributeCNN to a *segmentation-free* scenario by the use of word hypotheses. Instead of generating a fixed segmentation, competing hypotheses are created based on a text detector in combination with a connected component approach. The hypotheses are later ranked by attribute prediction using an AttributeCNN. Alternatively, [WLB17] showed that a CNN architecture similar to an object detection network can be used to predict attribute representations and bounding boxes simultaneously.

3.2.4 *Discussion*

Having a closer look at the area of word spotting and its development shows that it is, to a large extent, motivated by the limitations of HTR. One main application area is the exploration of a novel document collection often from a historic context. In this case, creating labeled data for a recognition model is cumbersome as, in many cases, this needs be done by a domain expert. Without labeled data, recognition usually renders a erroneous full transcription, which is often unusable. Tackling this scenario with a retrieval-based system allows for first exploration. As shown by many works on *learning-free* approaches (see Section 3.2.2), satisfactory performances can be achieved for QbE without introducing any manual labeling effort. On the other hand, using machine learning usually improves performances and gives the option for further capabilities such as QbS. The demand for annotated training data is the main determinant against the applicability of a word spotting system. This motivates the common distinction between *learning-free* and *learning-based* systems found in the literature. While this taxonomy was well suited for traditional methods, it does not address the actual problem of applicability.

Consider, for example, that training on synthetic data has become a common technique [GSF18; KDJ23]. In this case, a model is trained on a generated dataset where labels are created automatically. A word spotting system with a model trained on a synthetic dataset is technically *learning-based*, as it is optimized by learning from data. Nonetheless, in terms of the application scenario, no manual annotation process is involved, which is an essential question with respect to the required effort to apply such a system. As also shown by the method presented in this thesis, using machine learning techniques does not necessarily introduce the demand for manually annotated data.

Therefore, we propose the classification of systems based on their requirement for manually labeled data and use the term of *annotation-free* or *annotation-based*. In contrast to a label that can be created automatically, we consider an annotation to be created manually. An *annotation-free* system can therefore be applied to a novel document collection without the need for human intervention.

3.3 HANDWRITING SYNTHESIS

Training models generally requires labeled data. For deep neural networks the data demand can be tremendous and a major limitation. In this case, synthesizing data is an appealing idea because by designing a generation process, the label of a synthetic sample is known and can be used for training a model. In the context of document analysis and especially written text, designing simple generation processes is easily feasible as text and language follow a distinct set of rules. Using synthetic data to pretrain recognition or retrieval models has become a standard technique. While most works rely on data that is generated by rendering certain fonts, more recent approaches aim at learning the generation process from training data with Generative Adversarial Networks (GANs) or diffusion models. Table 1 gives an overview of works for handwriting generation that follow the Font-, GAN- or Diffusion Model-based approaches which will be discussed in more detail in the following.

One of the first works using synthetic data for a text analysis task was published in [Jad+14]. The proposed model for scene text recognition includes a CNN-based architecture that performs holistic word recognition. Therefore, the authors use a synthetic dataset that is created by rendering word images. For this purpose, a collection of different fonts is used to generate a large variety of word images based on a lexicon.

Krishnan et al, followed a similar approach by selecting fonts that resemble handwriting [KJ16]. The resulting HWSynth dataset was the foundation for their works on word recognition and retrieval in which they train neural networks on the combination of synthetic and real data [KJ19; KDJ23].

It is a general consensus that training a neural network on synthetically generated handwriting images results in beneficial initialization. Many works initialize their recognition or retrieval model by training on synthetic word images [PW16; GSF18; Dut+18; Kan+20b] or text lines [WZG22; Kan+22]. This seems to be increasingly more important with recent transformer architectures that usually rely on a high number of parameters [Bar+22; Li+23].

From a different perspective, training a model on synthetic data can alleviate the data demand for training a model. Gurjar et al. showed in [GSF18] that first training an AttributeCNN for word spotting on synthetic data and finetuneing the model with small numbers of real world samples is feasible. Comparing models trained on all available manually-annotated samples with models first trained on the synthetic dataset proposed in [KJ16] showed that similar performances can be achieved with a significant reduction in labeling effort. Nonetheless, training models only on synthetic data usually does not yield satisfactory performances.

While font-based approaches follow a completely designed, heuristic generation procedure, *learning-based* methods for handwriting generation were also explored in the literature. In this regard, GANs are a popular class of models that has received increasing interest from the research community [AMM19; Fog+20; Kan+20a; Mat+21b]. The underlying idea of a GAN for generating images is to employ two models, a generator and a disciminator, that compete with each other. The generator synthesizes images while the discriminator classifies them into real or fake images. By building the model on neural networks, it is feasible to design a loss function that optimizes the discriminator to better distinguish between real and fake images while the generator tries to generate images that fool the discriminator. This general approach has been applied to handwriting in [AMM19] allowing for the generation of word images of a fixed size. In this work, the generator and the discriminator were implemented by CNNs. The generation process was further conditioned on the word string that was embedded by a recurrent network. In order to ensure the generation of an image showing the correct word, an auxiliary recognition network was added. This allowed for the consideration of the CTC-loss during optimization. Fogel et al. propose a similar model in [Fog+20] using fully convolutional networks to generate images of different sizes. Other works include style embeddings that control the generated writing style [Kan+20a; Mat+21b]. A major drawback of GAN-based handwriting generation is that labeled data is required for training the generation model. Most works report only marginal performance gains when the generated images are used in the sense of data augmentation.

Recently, diffusion models have become increasingly popular for image generation. In order to train a neural network for generation, the model learns a stepwise denois-

Table 1: Overview on different approaches on handwriting synthesis.

Fonts	HWSynth [KJ16; KDJ16; KJ19; KDJ23], Gurjar et al. [GSF18], Kang et al. [Kan+18; Kan+20b], TrOCR [Li+23], Poznanski et al. [Jad+14; PW16], Wick et al. [WZG22], Dutta et al. [Dut+18], Barrere et al. [Bar+22].
Generative Adversarial Networks	Alonso et al. [AMM19], Fogel et al. [Fog+20], Kang et al. [Kan+20a], Mattick et al. [Mat+21b]
Diffusion Models	Zhu et al. [Zhu+23], Ding et al. [Din+23], Nikolaidou [Nik+23]

ing process. By conditioning the process, a designated image can be generated from an image of random noise. Several works also investigate this method for handwriting generation [Zhu+23; Din+23; Nik+23]. Even though visually convincing samples are generated, the influence on model performance after adding synthetic data to the training process is limited.

4 SELF-TRAINING

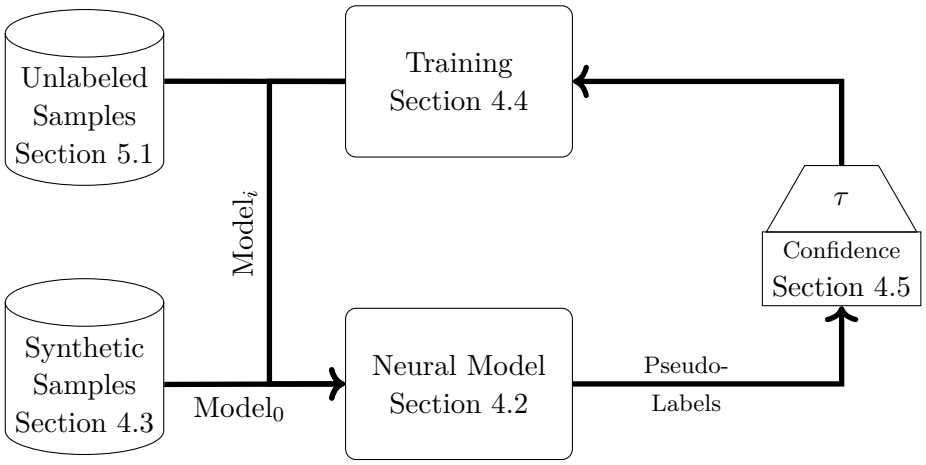


Figure 17: Schematic visualization of the self-training scheme presented in this thesis. Neural model are trained based on a confidence-based selection of pseudo-labels. An initial model (Model_0) is derived from synthetic data and later iteratively adapted with self-training on unlabeled data.

The underlying method of this work which allows for training handwriting recognition and retrieval models without the need for manually annotated samples is self-training. Self-training generally relies on the idea of training on pseudo-labels that are predicted by the model itself. After a short discussion of the theoretical background and works from the literature on self-training, this chapter introduces the main models and methods that are used in this work.

Figure 17 presents the conceptual overview of the proposed approach. At its core lies the fundamental self-training idea of using pseudo-labels to learn from unlabeled data. Section 4.2 introduces two models relying on neural networks that are typical realizations for the tasks of handwritten word retrieval and recognition. Their main methodological difference is that for retrieval the prediction of image attributes from a holistic image representation is used, opposed to a sequential approach for handwritten word recognition. In both cases, an initial model is required that serves as the starting point for self-training. In contrast to other works in the literature, this initial model is not trained on a manually-labeled training set. We explore the use of synthetically generated word images to avoid the labeling effort connected to an annotated dataset. See Section 4.3 for a detailed discussion of the generation pipeline. Section 4.4 discusses the training procedure that integrates unlabeled samples through pseudo-labeling. Depending on the evaluation scenario, the set of unlabeled samples comes from benchmark datasets commonly used in the literature. A detailed description of benchmarks and evaluation protocols is presented in Section 5.1. Models solely trained on synthetic data tend to generalize poorly to real world samples. Therefore, performances and especially the quality of predicted pseudo-labels is also poor. To circumvent training on a high number of falsely

pseudo-labeled samples, we investigate the concept of confidence estimation. Traditional neural networks for classification often consider the activation of the output layer as a form of pseudo-probability for the respective class. This can also be interpreted as a quantitative measure of how confident the network is in a certain prediction. Section 4.5 presents multiple approaches on how to generalize this idea to the attribute and sequence models considered in this work and how to exploit the confidence estimate to identify pseudo-labels that are more likely to be correct.

4.1 BACKGROUND

While many works in different domains show the effectiveness of self-training approaches and semi-supervised learning in general, it is not intuitive how unlabeled data may benefit the training process of a machine learning model. This question is discussed in the following, giving a theoretical motivation of how pseudo-labeling may be understood as a form of entropy regularization. After establishing the theoretical motivation, relevant works on self-training neural networks in general and in the context of document analysis are reviewed.

4.1.1 *Why could learning from unlabeled data help prediction?*

An obvious question that arises when looking at semi-supervised learning literature is why unlabeled data is supposed to support learning a classification function. Regarding a classification problem, the classifier needs to approximate a function $f : \mathcal{X} \to \mathcal{Y}$, that maps an instance $\mathbf{x}$ to a class label y. In semi-supervised learning, a number of unlabeled samples are available, which, in a mathematical sense, allows for the estimation of the distribution $P(\mathbf{x})$. Many works show classification performances can be improved by considering the set of unlabeled samples. Therefore, it can be concluded that there is a meaningful relation between the input distribution $P(\mathbf{x})$ and the conditional probability $p(y|\mathbf{x})$.

With respect to Bayesian decision theory, statistical classifiers minimize the risk of a wrong classification [DHS01, p. 24]. Assuming that all wrong classification decisions carry the same cost, the maximum a posteriori decision rule can be derived:

$$\operatorname*{argmax}_y p(y|\mathbf{x}) = \operatorname*{argmax}_y \frac{p(y)p(\mathbf{x}|y)}{p(\mathbf{x})} = \operatorname*{argmax}_y p(y)p(\mathbf{x}|y). \tag{40}$$

In the classical maximum-a-posteriori (MAP) framework, observing $p(\mathbf{x})$ does not contribute to the classification decision. Therefore, including unlabeled data which effectively boils down to observing $p(\mathbf{x})$ would have no benefit for a classification task. Regardless, empirical results in a plethora of works [EH20; Yan+22] in different application areas show that this is not the case.

From a theoretical perspective, there are two main assumptions that intuitively motivate how unlabeled data may benefit learning a classification function [CSZ06, pp. 5–7]. If two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are close to each other in the respective feature space, then the corresponding outputs should be also close or, in case of classification, the label should be the same. This assumption is commonly known as the *semi-supervised smoothness assumption*. Especially if the path between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ lies in a high density region, this assumption is required to be true for semi-supervised learning to work.

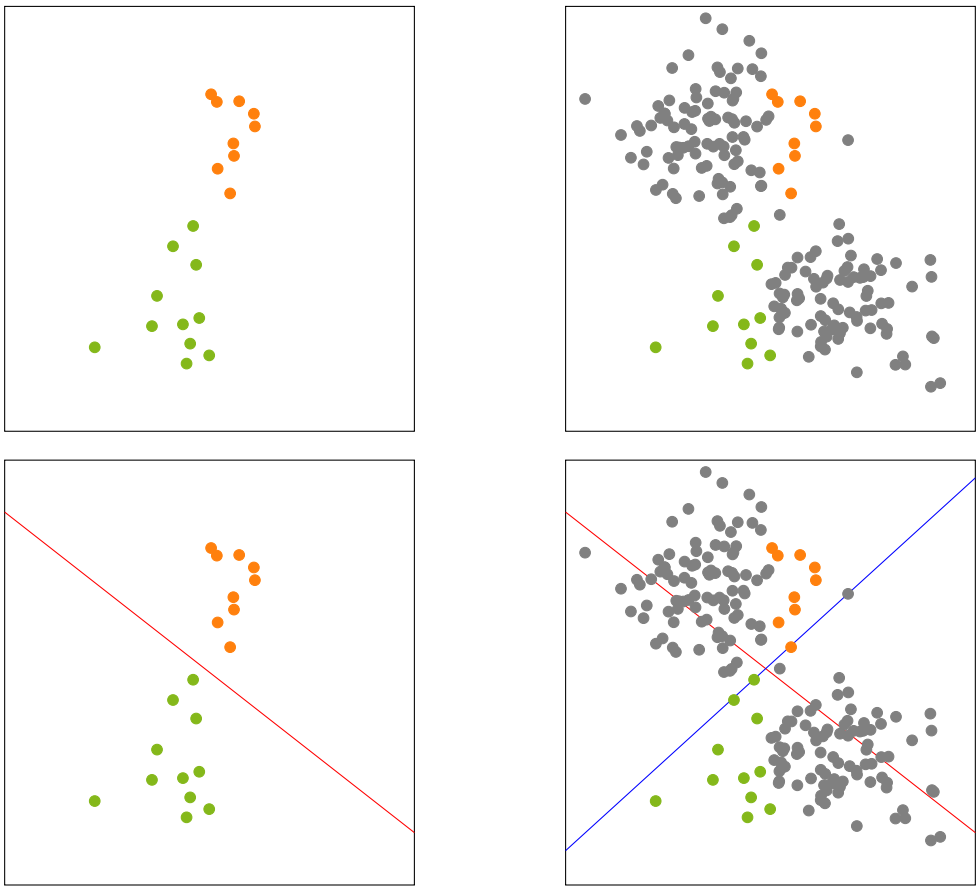


Figure 18: Toy example for the choice of a decision boundary under the availability of unlabeled data. Samples are drawn from two Gaussian distributions representing two different classes. Labeled samples are indicated by respective colors, unlabeled samples are represented by gray dots. If the smoothness assumption holds true for both clusters, optimizing the decision boundary only based on labeled samples (red line) is clearly inferior to the optimal decision boundary (blue line). Including unlabeled data has the potential to better optimize the decision boundary in a low density area.

This assumption is closely related to the *cluster assumption* that also underlines the general motivation behind semi-supervised learning [CSZ06, pp. 5–7]. In this case, it is assumed that feature vectors belonging to the same class form clusters in the feature space. Furthermore, this directly implies that if two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ fall in the same cluster, they share the same class label. The cluster assumption also implies a clear indication on how to choose a decision boundary. As clusters may conceptually be interpreted as high-density regions, a decision boundary should lie in an area of low-density. The idea of low-density separation is directly connected to the smoothness assumption [EH20]. A decision boundary in a high density area, would intersect the path between two points $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ resulting in different classification outputs y_1 and y_2. As the decision boundary is in a high-density area, the path between $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ would be too, resulting in a direct violation of the smoothness assumption. This example shows that even though smoothness and cluster assumption are often individually considered, they are highly interrelated.

Figure 18 visualizes a toy example of how unlabeled data supports the choice of a decision boundary if the previously introduced assumptions hold true. As labeled samples are drawn randomly from the data distribution $p(\mathbf{x}, y)$, the decision boundary derived from a supervised learning algorithm is likely to be suboptimal. This is increasingly likely for smaller numbers of labeled training samples. In the case of semi-supervision, adding a high number of unlabeled samples allows for an implicit model of $p(\mathbf{x})$ which is then exploited to fit the decision boundaries to areas of low-density. If smoothness and cluster assumption hold true, the decision boundaries are likely to be better approximations of the optimal decision boundary.

In general, this leads to the conclusion that the knowledge of the distribution $p(\mathbf{x})$ allows an induction bias conveying information on $P(y|\mathbf{x})$ [GB04] to be derived. Self-training is considered to minimize the overlap of classes and, therefore, the empirically observed performance gains can be explained by the facilitated learning of decision boundaries in low density areas. From a theoretical perspective, self-training is often considered a form of entropy minimization [GB04; CSZ06; Lee13]. In this case, the conditional entropy is a measure of class overlap. By minimizing the entropy of the unlabeled data, the cluster assumption is enforced and decision boundaries in low density areas are encouraged. With respect to neural networks, entropy is minimized in an implicit fashion. As class information is usually encoded in a one-hot fashion, the network is trained to predict a high pseudo-probability for a single class and zero elsewhere. Independent from the quality of the predicted pseudo-labels, this characteristic is enforced by supervised training on pseudo-labeled samples. Therefore, the network implicitly minimizes the entropy on the unlabeled data samples as the training procedure encourages condensed clusters in the feature space.

4.1.2 *Self-Training for Deep Neural Networks*

Self-training and pseudo-labeling are long-standing techniques that have been investigated in many different domains and for different types of classifiers. The first work that uses this training method in the context of neural networks was published in [Lee13]. In their work, the authors essentially trained a Multi Layer Perceptron (MLP) to recognize the single digits of the MNIST dataset. The model was trained on the combination of a small amount of labeled data and the unlabeled dataset.

Pseudo-labels are generated with respect to the class with highest softmax activation and are recalculated at every weight update. In order to train the model on labeled and on pseudo-labeled data simultaneously, a supervised loss is computed for both types of samples and combined with a weighting factor. The model is then simply optimized with stochastic gradient descent. Compared to the model only trained on the small fraction of labeled samples, the authors were able to show that significant performance gains can be achieved by self-training.

Self-training later gained a lot of popularity in the context image classification with the line of research published in [Ber+19; Ber+20; Soh+20]. In their works, the authors use pseudo-labeling for semi-supervised learning for object recognition datasets such as CIFAR-10, CIFAR-100 [KH+09] or STL-10 [CNL11]. Similar to [Lee13], the model is trained on a small number of labeled samples that are combined with pseudo-labeled examples.

In [Ber+19], the authors introduced two extensions of the basic self-training approach and proposed a method called MixMatch. Firstly, their pseudo-label generation relies heavily on augmentations which introduce a so called consistency regularization. The main idea is that a sample does not change its label by applying standard augmentation techniques. Instead of predicting the pseudo-label from a single sample directly, multiple versions are created by augmentations. The prediction is then averaged over the different augmented instances. Secondly, the authors use a sharpening technique to process the output distribution resulting from the average prediction. Instead of using just the class with maximal activation which corresponds to a one-hot encoded vector, a temperature parameter is introduced resulting in a target vector of lower entropy. During training, the batches contain labeled and pseudo-labeled samples. Separate loss terms are computed and combined by a weighting factor. Over the course of training, the influence of the unlabeled samples is increased. In their results, the authors show that pseudo-labeling allows models to be effectively trained for object recognition in a semi-supervised setting.

MixMatch was later extended through the introduction of distribution alignment and augmentation anchoring in [Ber+20]. In the case of a semi-supervised setting, a small number of labeled samples is considered to be available. The assumption is that this small labeled set is representative of the problem and, therefore, the predicted pseudo-labels should follow a similar distribution. This idea was first published in [BHM91]. In [Ber+20], the author integrated this approach into a self-training process of a neural network by computing a running average over the predicted pseudo-labels. This average is then used to normalize the respective predictions encouraging the distributions to be close.

Further improvements were made by augmentation anchoring which relies on differently strong augmentations. The pseudo-label is predicted given a weakly augmented image. During training, the model is trained with the same image underlying stronger image augmentations.

The use of differently intense augmentation strategies was further explored in [Soh+20]. The author showed that MixMatch [Ber+19] as well as ReMixMatch [Ber+20] can be simplified through the use of just two augmented versions of the same image. Again the pseudo-label is predicted from a weakly-augmented one, but in this case only a single strongly augmented instance is used during training. Furthermore, the authors introduce thresholding with the goal of only retaining pseudo-labeled samples with high quality. Therefore, a pseudo-label is only retained if the

output activation of the model is above a certain threshold. This can be considered a form of confidence estimation with the model activation being a numeric estimate for the model's confidence prediction. The threshold hyperparameter allows pseudo-label quality to be balanced against the quantity of available pseudo-labeled samples. One of the authors' finding with regards to tuning the threshold parameter was that, in general, high threshold values consistently give higher performances.

The very influential works on self-training in the area of semi-supervised learning in object classification illustrate well the potential of learning from unlabeled data via pseudo-labeling. For document analysis problems, this idea is only rarely applied even though, in many application, the availability of annotated sample data hinders the application of machine learning models. Similar to other domains, several works explore training a recognition model first on a manually labeled dataset which is then adapted using self-training [SCP17; DJ20; KBH21; RSN21]. If a well-performing initial model is available, performance gains can be observed. As documents and especially language are highly regularized domains, some works aim at including this prior knowledge in the training process. In [SCP17], a handwriting recognition model for French is trained in a semi-supervised setting. The authors use an externally derived lexicon to verify pseudo-labels and enhance their prediction quality. [Ten+18] also trains a recognition model, but aims at transferring the model from one language to another using pseudo-labeling. Here, a language model for the target language provides further language information which supports the adaptation process.

4.2 NEURAL NETWORKS FOR WORD RECOGNITION AND RETRIEVAL

This chapter introduces the fundamental models that build the core of the recognition and retrieval systems and are later trained by the proposed self-training approach. Despite being conceptionally different problems, modern retrieval and recognition models share a common structure that is closely related to the classical pipeline of a computer vision system. In this case, neural networks and especially Convolutional Neural Networks (CNNs) allow multiple steps, such as feature extraction and prediction, to be integrated into a single model. Nonetheless, different architectural components reoccur in almost any successful network for handwriting recognition and retrieval and can be summarized by a three step architecture. First, a CNN serves as a feature extractor and computes a set of feature maps for a given input image. As the resulting features are still extremely high dimensional despite the use of pooling layers inside of the CNN, extraction is commonly followed by an aggregation step. Here, the goal is to compute a compact and characteristic representation based on the extracted feature maps. This can either be achieved in a holistic manner, for example by spatial pooling, or alternatively by slicing and sequentializing the set of feature maps into a sequence of feature vectors. Finally, a decoder computes the desired outputs. In the recognition case, common decoders based on CTC or based on a sequence-to-sequence approach directly compute the recognition sequence given the sequence of feature vectors. For retrieval, a holistic view is more common. Given a single feature vector representing the entire image, a fully connected network allows commonly used attribute representations to be predicted. This holistic and often redundant prediction of image attributes promises a higher robustness which is often of high importance for a retrieval model.

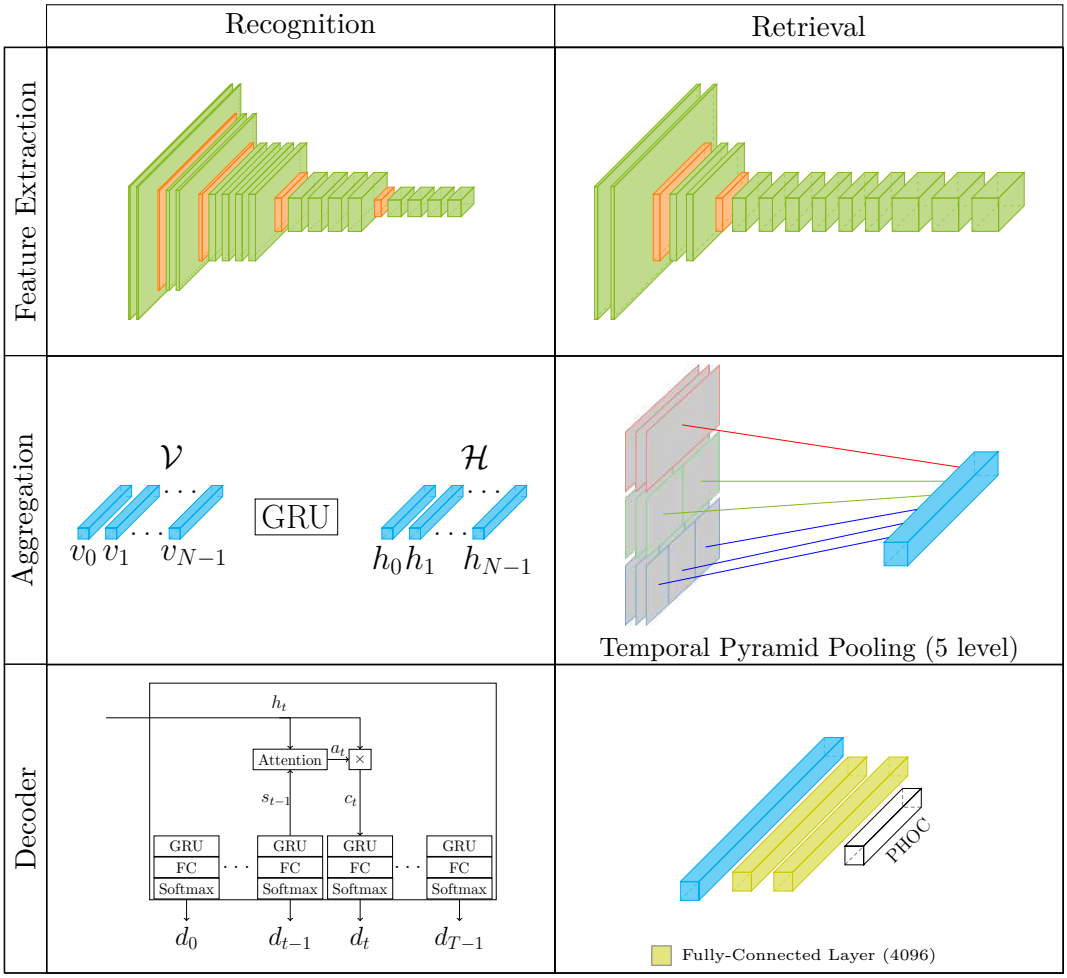


Figure 19: Comparison of the architectural components of the recognition and retrieval models. Both architectures follow the general three step procedure of feature extraction, aggregation and decoding. Details on the recognition and retrieval model are discussed in Section 4.2.1 and Section 4.2.2, respectively.

This work mainly relies on two different models that are discussed in detail in the following. The recognition model utilizes a sequence-to-sequence approach, heavily inspired by [Kan+18]. The retrieval model follows a holistic approach based on an AttributeCNN, which was introduced and extensively investigated in [SF16; SF17; SF18]. Figure 19 visualizes the architectural structure of both models. Individual model components can be distinguished according to the general three step procedure of feature extraction, aggregation and decoding.

4.2.1 *Sequence-to-Sequence Handwriting Recognition*

Sequence-to-sequence handwriting recognition models avoid the use of the commonly used CTC-loss where the prediction of the blank symbol is required without any visual evidence. Attention-based decoding was shown to be a promising alternative [BLM17; Kan+18; Sue+18; Abe+21]. We adapt most of the design choices from [Kan+18] to allow for a fair comparison of training and adaptation methods, which is the main focus of this work.

Training and test images are preprocessed to a fixed height. This allows for the extraction of a sequence of feature vectors of the same length. Therefore the model architecture does not have to account for changing input sizes except for different sequence lengths. A VGG19 network [SZ15] serves as a feature extractor and computes a set of feature maps $\mathbf{X}$. The aggregation stage consists of two steps. First, the computed feature maps are transformed into a sequence of feature vectors. The feature maps can be interpreted as a volume where a slice corresponds to a column of the input image. By reshaping the slice to a one-dimensional vector, the volume spanned by the feature maps is transformed into a sequence of feature vectors. Next, two layers of Bi-directional Gated Recurrent Units (BRGU) allow for recurrent connections inside the network, resulting in an enriched feature representation $\mathbf{H}$ that encodes context information:

$$f_{VGG19}(\mathbf{I}) = \mathbf{X} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N), \tag{41}$$

$$\mathbf{H} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_N) = \mathrm{BGRU}((\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N)). \tag{42}$$

The VGG19 backbone of the network can be considered a function $f_{VGG19}(\mathbf{I})$ that computes a set of feature maps $\mathbf{X}$ given the input image $\mathbf{I}$. By slicing and reshaping, the feature maps X are then transformed into a sequence of feature vectors $(\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N)$. This sequence serves as an input to the BGRU layers which compute the output sequence $(\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_N)$.

For recognition, decoding constitutes the task of predicting a sequence of characters from the given sequence of contextualized feature vectors. Here, a one-directional GRU is combined with an attention mechanism to effectively predict the character sequence. The ultimate goal of the attention is to predict a vector $\mathbf{c_t}$ that encodes all required context information. The literature distinguishes two main variants of attention, namely content [BCB15] and location-based [Cho+15] attention. The idea of attention boils down to computing an attention mask vector $\mathbf{a_t}$ that is then used to compute a weighted sum of the feature vector sequence:

$$\mathbf{c_t} = \sum_{i=0}^{N} a_{ti} \mathbf{h_i}. \tag{43}$$

First, a scoring function e is defined that computes a score e_{ti} for feature vector i at decoder step t. Given the current decoder state $\mathbf{s_t}$, content based attention is defined as

$$e_{ti} = \mathbf{w}^T \tanh(\mathbf{W}\mathbf{h_i} + \mathbf{V}\mathbf{s}_{t-1} + \mathbf{b}). \tag{44}$$

By defining $\mathbf{w}, \mathbf{W}, \mathbf{V}, \mathbf{b}$ as learnable weights, the attention mask allows the model to give different importance to individual parts of the feature vector sequence $\mathbf{H}$ based on the input and the current decoder state. In the case of content-based attention, the score and also the resulting attention weights depend only on the decoder state $\mathbf{s_{t-1}}$ and the respective feature vector $\mathbf{h}_i$. In order to derive the attention scores $\mathbf{a}_t$, the scores are normalized by applying the softmax function:

$$\mathbf{a_t} = \text{softmax}(\mathbf{e}_t). \tag{45}$$

A drawback of content-based attention is that the position of the considered feature vector is irrelevant for the score computation. Therefore, similar feature vectors $\mathbf{h}_i$ are scored similarly. For many sequential domains such as handwriting or speech recognition, it is beneficial to take the positional information into account. The attention mechanism may be extended by explicitly encoding positional information. Location-aware [Cho+15] attention defines a matrix $\mathbf{F}$ that is convolved with the previous attention mask $\mathbf{a_{t-1}}$. This results in a set of k position dependent vectors $\mathbf{f}_t$ that contribute to the attention score computation

$$e_{ti} = w^T \tanh(\mathbf{W}\mathbf{h_i} + \mathbf{V}s_{t-1} + \mathbf{U}\mathbf{f}_{t,i} + b), \tag{46}$$

where $\mathbf{U}$ again constitutes a matrix with trainable parameters. Here, the attention weights of the previous position contribute to the score computation which makes the attention mechanism location-aware. To ensure a broader attention mask that does not focus on narrow image areas, attention smoothing is commonly applied [Cho+15].

Conventionally, a decoder predicts an output symbol y_t based on the concatenation, denoted as $[\cdot, \cdot]$, of the context vector $\mathbf{c_t}$ and the decoder state $\mathbf{s_t}$. This can be realized using recurrent components such as GRUs in combination with a linear prediction layer $w(\cdot)$:

$$y_t = \text{argmax}(w([\mathbf{c_t}, \mathbf{s_t}])). \tag{47}$$

As shown in [Kan20], the decoder can be efficiently simplified to rely only on the decoder state $\mathbf{s_t}$:

$$y_t = \text{argmax}(w(\mathbf{s_t})). \tag{48}$$

Due to the complexity of the neural decoder components, an implicit encoding of the context vector in the decoder state does not degrade performances. Therefore, the decoder state $\mathbf{s_t}$ is defined as

$$\mathbf{s_t} = \text{Decoder}([\tilde{y}_{t-1}, \mathbf{c_t}], \mathbf{s_{t-1}}), \tag{49}$$

with $\tilde{y}_{t-1}$ being a precomputed embedding of the symbol predicted at the previous time step.

In this work, the decoder is implemented in analogy to [Kan+18; Kan20; Kan+20b] as a one-directional multi-layered GRU. Each prediction sequence starts with a <GO> signal. Then, the decoder predicts character symbols until an end of signal

symbol is predicted. This enables the prediction of sequences with different lengths compared to the input feature vector sequences. Therefore, it is sufficient to resize the input images to the same height, leaving the aspect ratios unaffected.

The entire recognition model may be summarized by an encoder $\mathcal{G}_e$ and a decoder $\mathcal{G}_d$ function. The encoder $\mathcal{G}_e : \mathbf{I} \rightarrow \mathcal{H}$ maps an input image $I \in \mathcal{I}$ to a feature representation $\mathbf{H} \in \mathcal{H}$ which is then sequentialized $\mathbf{H} = (\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_N)$. The decoder $\mathcal{G}_d : \mathcal{H} \rightarrow \mathcal{Y}$ predicts the final recognition sequence $(\hat{y}_0, \hat{y}_1, \ldots, \hat{y}_N) \in \mathcal{Y}$. Both functions are realized by neural networks with a set of parameters θ_e and θ_d. To optimize the model parameters θ based on a set of annotated training data $\mathbf{X}$, we define the recognition loss $\mathcal{L}_r$ as

$$\mathcal{L}_r(\theta|\mathbf{X}) = \sum_{i=1}^{N} l_r(\mathcal{G}_d(\mathcal{G}_e(\mathbf{x}^{(i)})), y_i), \tag{50}$$

with $\mathbf{x}^{(i)}$ being a training sample with the corresponding annotation y_i. The recognition loss l_r results from the binary cross entropy between the decoder prediction and the symbol annotation at each decoder step t. Finally, the entire model is optimized in a fully-supervised manner by stochastic gradient descent or techniques like ADAM.

4.2.2 *Attribute-based Word Retrieval*

In the segmentation-based scenario, i.e., when a word segmentation is computed independently, retrieval can be considered the task of computing the similarity between a query and all elements in a dataset. The popular embedded attributes framework [Alm+14] tackles this task by mapping word images to attribute representations. These attribute representations effectively constitute a high dimensional vector space. By designing character based attribute representations, syntactic similarity between word images corresponds to an easy to compute vector distance in the attribute space.

The success of this approach is heavily determined by the accuracy of the attribute prediction. As the task of predicting attributes from a word image can be considered a multi-label classification problem, convolutional neural networks rapidly became the model of choice [Gio+17]. In this work, we follow the AttributeCNN approach presented and researched in [SF16; SF17; SF18]. Even though the model integrates the prediction of attributes into a single network architecture, the structure of feature extraction, aggregation and decoding can be distinguished. The model considered in this work is essentially a TPP-PHOCNet, first presented in [SF17].

Feature extraction uses a variant of the successful VGG16 architecture [SZ15], which was designed for object recognition in natural images. A significant difference between natural images and word images is that word images show considerable differences in sizes and aspect ratios. The design of the PHOCNet architecture was therefore heavily motivated by the goal of avoiding rescaling. Because word images are processed as they are captured in the different benchmarks, the number of pooling layers in the feature extractor is reduced to two. This allows images of smaller sizes to be processed compared to the original architecture.

The main architectural change with respect to the original VGG architecture, was made at the aggregation stage. At this point, an holistic vector representation for

the entire word image is computed. Doing so with a simple pooling layer makes the size of the image representation depend on the dimensions of the input image. This makes the use of a fully-connected MLP for attribute prediction unfeasible. In order to produce an image representation of fixed size that is independent from the input image dimensions, [SF16; SF17] proposes the use of a spatial pooling approach inspired by [He+15b]. The convolutional architecture computes a set of k feature maps. At aggregation, this set of feature maps shall be summarized in a single characteristic feature vector with a fixed size independent of the input image dimensions. Therefore, each feature map is divided in a quadtree-like structure [Sze10]. Depending on the predefined granularity, this results in $1, 4, 16, ...$ regions per level. For each of these regions, a simple pooling operation is performed. The final feature vector results from the concatenation of all pooling outputs. This approach is known in the literature as Spatial Pyramid Pooling (SPP) [He+15b] and has been successfully applied in the form of an SPP-layer in an AttributeCNN in [SF16]. Several works in the literature that rely on spatial pyramids in combination with classical descriptor-based approaches indicate that spatial division along the horizontal axis is less important and may be neglected [Rus+11; SF15; Rus+15a; WRF16]. This idea has been adopted in [SF17], and the SPP layer is commonly replaced by a so-called Temporal Pyramid Pooling layer, that only divides the feature maps along their horizontal axis. An additional effect of the removal of a vertical division is that it also drastically reduces the dimension D of the feature vector at the same level L:

$$D_{\text{SPP}} = \sum_{i=0}^{L-1} 4^i \cdot k \quad > \quad D_{\text{TPP}} = \sum_{i=0}^{L-1} 2^i \cdot k. \tag{51}$$

This is of special consideration, as common architectures use fully-connected layers directly after the spatial pooling layer. In this case, the number of network parameters is directly determined by the feature vector dimension.

The ultimate goal of the AttributeCNN is the prediction of an attribute vector $\mathbf{y}$. A common choice that is adopted for the used retrieval model is the Pyramidal Histogram of Characters (PHOC) [Alm+14]. This embedding constitutes a binary vector that encodes the occurence of a character in a specific split of the string. Therefore, the annotated string is split in a pyramidal fashion. For each split, an occurrence vector is computed whose length is equal to the number of predefined unigrams. Whenever a unigram is present in the corresponding split, the PHOC vector holds a one, otherwise it holds a zero. Even though this representation is highly redundant, it has been shown that it is a robust and efficient choice for word spotting systems. The similarity between word images is then computed by the cosine similarity between the predicted PHOC representations. As an additional benefit, query-by-string is natively possible, as a PHOC representation can be directly derived from any given string. Therefore, strings and word images are mapped into the same high-dimensional embedding space allowing the similarity between string and image to be directly computed based on the vector distance of their PHOC representations.

Architectually, the prediction of the attribute representation is implemented by a fully-connected attribute decoder. The proposed system follows the design of [SF18] and uses an MLP with two fully-connected layers and 4096 neurons each. Ultimately, the output layer corresponds to the size of the target PHOC vector dimension and computes a pseudo-probability for each attribute by using sigmoid activations.

In analogy to the sequence-to-sequence model presented in Section 4.2.1, the AttributeCNN can be summarized by an encoder and a decoder function, $\mathcal{G}_e$ and $\mathcal{G}_d$, that are connected via a TPP layer:

$$\hat{\mathbf{y}}_{\text{PHOC}} = \mathcal{G}_d(\text{TPP}(\mathcal{G}_e(\mathbf{x})). \tag{52}$$

A common assumption for binary attribute representations is that the attributes are independent and follow a Bernoulli distribution [SF18]. Therefore, the model parameters of the AttributeCNN can be optimized using a Binary Cross Entropy l_{BCE} loss between the prediction $\hat{\mathbf{y}}$ for each training sample and its corresponding PHOC annotation $\mathbf{y}$:

$$l_{\text{BCE}} = -\sum_{i=1}^{n}\sum_{j=1}^{d} y_j \log(\hat{y}_j) + (1-y_j)\log(1-\hat{y}_j), \tag{53}$$

$$\mathcal{L}_{Att}(\theta|\mathbf{X}) = \sum_{\mathbf{X}} l_{\text{BCE}}(\mathcal{G}_d(\text{TPP}(\mathcal{G}_e(\mathbf{x}))), \hat{\mathbf{y}}). \tag{54}$$

4.3 SYNTHESIS OF HANDWRITTEN WORD IMAGES

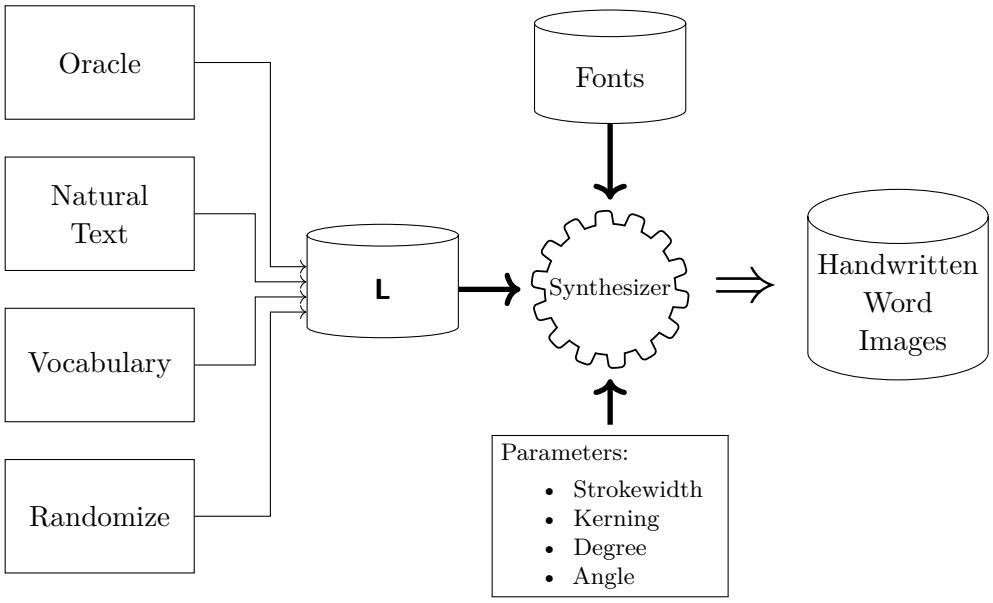


Figure 20: Overview of the proposed synthesis procedure. A labelset **L** can be extracted from different sources. The synthesizer generates handwritten word images by randomly sampling from fonts and style parameters.

The goal of data synthesis is to generate training samples that model the variance occurring in real-world data. Essentially, we need to design a function $f_{syn} : \mathcal{Y} \rightarrow \mathcal{I}$ that maps a given label y to a handwritten word image $\mathbf{I}$. Handwriting is a domain where it is natural to consider synthetic data generation as it is highly regularized. Languages provide a predefined set of rules, such as character sets or words that define which data is reasonable. Based on this general knowledge, it is feasible to design a synthesis pipeline that allows datasets of arbitrary size to be generated.

Designing such a pipeline requires several modeling decisions. On the one hand, we require a vocabulary that defines which strings and also which characters shall be included in the dataset. Furthermore, it is an open question whether the distribution of the vocabulary is relevant for dataset generation. On the other hand, a realistic rendering of the provided string is required. As the models are applied to handwritten documents, we aim at generating word images that also visually resemble a handwritten style. Figure 20 summarizes the synthesis approach and its key components.

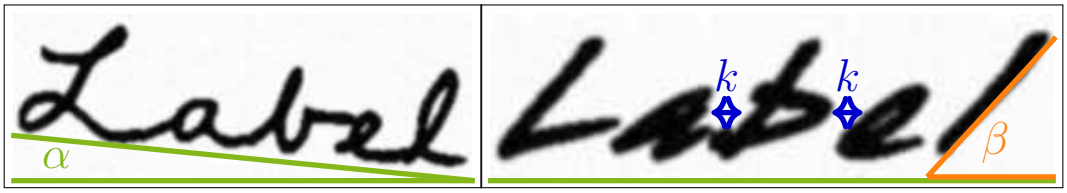
Labelset

The choice of a set of labels $\mathbf{L}$ and a corresponding distribution $p(y)$ can be considered a form of language modeling. As the information on language may only be learned implicitly during training, it is an open question if it is actually encoded and learned in a synthetic pretraining stage. To investigate this, we propose the use of different labelsets, that are later rendered into handwritten word images.

An obvious option is to gather a collection of written text and simply generate corresponding word images. In this case the distribution of words and characters as well as the use of punctuation and capitalization is determined by the selection of the document corpus. More specifically, using, i.e., a literature corpus as the source for the labelset, results in a training dataset where the number of training samples per word depends on its frequency of occurrence in the natural use of a language. In order to gain insight into whether the distribution of words influences the models' performances, we remove the natural distribution from the generation process. Therefore, we use the most frequent English words and generate a fixed number of samples for each. In contrast to the text corpus extracted from natural text, the samples per word follow a uniform distribution.

To further remove language characteristics at the stage of synthetic pretraining, we investigate whether and how performances degrade when the model is trained on randomized strings. In this case, not only is the word distribution removed from the labelset but also any information on word level. If the model learns independent character information, the influence of removing word level information should be limited. If performances degrade strongly, it is a clear indication that synthetic pretraining constitutes a form of language modeling. In order to generate strings that at least resemble words, we introduce the following constraint. The distribution of length follows the same distribution as occurs in the natural text corpus, which also holds true for the occurrence of capital letters and special characters.

Randomizing strings aims at removing any information on language from the synthetic dataset. On the other extreme, it is an open question whether the model benefits from more specific language information from the target domain. Therefore, we create an oracle dataset. For the oracle dataset, the synthetic data is based on the closed dictionary of the evaluation dataset. This means that only words that are part of either the training or test set are generated. From a practical application perspective this closed dictionary is not available.



Parameter:
- Font: t = Popsies
- Strokewidth: $w = 0$
- Kerning: $k = 0$
- Degree: $\alpha = 5°$
- Slant: $\beta = 0°$

Parameter:
- Font: t = QEKevinShirley
- Strokewidth: $w = 1$
- Kerning: $k = 1$
- Degree: $\alpha = 0°$
- Slant: $\beta = 40°$

Figure 21: Exemplary renderings of the label y = label for different parameter configurations.

Image Synthesis

After modeling language and word information by extracting the underlying labelset, the next step in a synthesis pipeline is the design of the function $f_{\text{syn}} : \mathcal{Y} \to \mathcal{I}$ that maps a given label $y \in \mathbf{L}$ to a handwritten word image $\mathbf{I} \in I$. This mapping can easily be realized by off-the-shelf solutions that render strings to a word image. In this work, we use the open-source software ImageMagick[1] for rendering. The framework allows word images from a given string to be generated. Image appearance and thereby the resulting writing style is defined by a TrueType[2] font and a set of style parameters. TrueType is a data format that defines the appearance of a set of glyphs and is the defacto standard approach to encode computer generateable writing styles. Each glyph, for example each character in a given alphabet, is defined by a set of points that allow for the interpolation of corresponding Bezier curves. Combining the individual letterforms into a string allows for the generation of any given string that is built upon the predefined alphabet. As we aim to generate word images that resemble handwriting, the proposed synthesizer relies on a publicly available collection of fonts $\mathcal{F}$ with similar appearances.

While fonts define the general appearance, the writing style can be further modified by different parameters. The strokewidth w describes the thickness of the line. Kerning k allows for the adjustment of the distance between characters. Furthermore, an angle α is defined which defines if and to what degree the baseline of a word is rotated. Lastly, each character follows a slant angle β. Figure 21 shows the different style parameters and their effect on the rendered word image. In sum, the word image $\mathbf{I}$ for a given label y results from a selected true type font t and a given set of style parameters:

$$\mathbf{I} = f(t, y, w, k, \alpha, \beta). \tag{55}$$

During synthesis, a word image is generated for each label in the labelset. A font is drawn randomly following a uniform distribution. Strokewidth and kerning are set to either 0 or 1 with equal probability. The baseline angle α is drawn randomly from the integer interval of $[-2, 2]$. The slant angle is randomly chosen from a set of

1 https://imagemagick.org/
2 https://developer.apple.com/fonts/TrueType-Reference-Manual

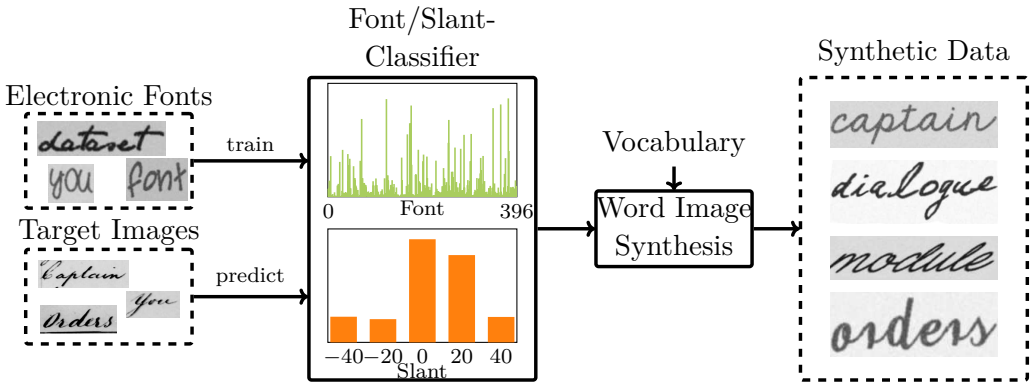


Figure 22: Architecture of the synthesis calibration system. Font and slant classification networks are trained on synthetic images. Respective histograms can be derived from predictions on the target images. An adapted word images synthesis includes the style information encoded by the histograms.

predefined options, specifically $[-40, -20, 0, 20, 40]$. After generating a random set of style parameters and selecting a font, a word image of height 80 pixels is generated with ImageMagick. As the input label is known the annotation is trivially available.

4.3.1 *Synthesis Calibration*

Considering the synthesis parameters discussed in Section 4.3, the selection of a font and slant angle strongly defines what can be considered the style of a writer or document. Instead of randomly choosing these parameters following a uniform distribution, we propose predicting them for a target dataset. Firstly, a synthetic dataset is generated and fonts as well as slant angles are uniformly distributed. This dataset set is then used to train a font and slant angle classifier. Training these classifiers solely on synthetic data does not introduce any additional labeling effort and allows distributions on fonts and slant angles to be derived for a given target dataset. The assumption is that, by classifying each word image from the target dataset, we can estimate a distribution that allows a dataset that is visually more similar to the target set to be synthesized.

Font classification has been of interest for the document image analysis community, especially for optical character recognition, as it allows designated models to be selected depending on the predicted font. We adapt the approaches presented in [TSM17; SB19] and use a convolutional neural network to classify the font of a given word image. Each font used in the synthesis procedure is considered a class. As a backbone architecture, we use the residual network ResNet50 [He+16] with the number of output neurons equal to the number of available fonts. In order to not rescale the differently sized word images, we remove the batch normalization layers and we train the network with pseudo batches, meaning that each sample is processed individually but weights are updated after a batch of training images. The network is trained for $40,000$ iterations with a batch size of 64 using ADAM optimization and cross-entropy loss. The same approach is taken for predicting slant angles. Each angle used in the synthesis procedure is considered a class, resulting in a ResNet50

Algorithm 1 : Basic Self-Training Scheme

Input : Synthetic data $\mathbf{S} = \{(\mathbf{s}^{(0)}, y_0), \ldots, (\mathbf{s}^{(N)}, y_N)\}$; unlabeled training images $\mathbf{X} = \{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(M)}\}$; number of training cycles K; Model $\phi(\cdot, \theta)$;

Train initial model $\phi(\cdot, \theta^0)$ on $\mathbf{S}$;
for $k \leftarrow 0$ **to** K **do**
  Derive pseudo-labels for $\mathbf{X}$: $\bar{y} = \phi(\mathbf{x}, \theta^k)$ for each element $\mathbf{x}$ in $\mathbf{X}$
  $\mathbf{X}_{\text{train}} = \{(\mathbf{x}^{(0)}, \bar{y}_0) \ldots, (\mathbf{x}^{(M)}, \bar{y}_M)\}$
  $\theta^{k+1} \leftarrow$ train $\phi(\cdot, \theta^k)$ on $\mathbf{X}_{\text{train}}$

architecture with five output neurons. Training follows the same hyperparameters as used for training the font predictor network.

Both networks are exclusively trained on synthetically generated word images. In order to adapt the synthesis procedure to an unknown dataset, we predict a font and slant angle for each sample from the target dataset. The histograms of predicted fonts and slant angles constitute an approximate distribution of which fonts and slant angles result in visual similar word images. An adapted synthetic dataset is then generated by replacing the uniform distributions of the synthesis procedure described in Section 4.3 with the distributions approximated by the font and slant angle predictors. Figure 22 presents an overview of the synthesis calibration procedure.

4.4 TRAINING PROCEDURE

A basic implementation of self-training can be realized for the models discussed in Section 4.2 based on a synthetic dataset which followed the generation process described in Section 4.3. See Algorithm 1, for a summary of the training procedure. Let $\mathbf{S}$ be a synthetic dataset containing word images $\mathbf{s}^{(0)}, \ldots, \mathbf{s}^{(N)}$, with corresponding labels $y_0, \ldots y_N$. The goal is to train a model $\phi(\cdot, \theta)$ for a target dataset $\mathbf{X}$ for which no annotations are available. In general, we can use the same self-training approach for both recognition with a sequence-to-sequence HTR model and retrieval with an AttributeCNN. First, an initial model is required. In order to derive an initial set of weights θ^0, we first train the model on the synthetic dataset until convergence. This allows us to use the initial model to make predictions for the unlabeled target dataset. These predictions are then assumed to be correct and, therefore, constitute a pseudo-label. The previously unlabeled dataset in combination with the predicted pseudo-labels can then be considered a training dataset $\mathbf{X}_{\text{train}}$. Training on this dataset results in a new set of weights θ^k. Predicting pseudo-labels and training on the newly generated training set is performed iteratively. This allows the model to improve pseudo-label prediction and adapt from the information learned from synthetic data to the real unlabeled dataset.

Generating a pseudo-label is straight forward for the recognition model discussed in Section 4.2.1. The recognition decoder consists of a linear layer with softmax activation, which gives a prediction vector $\hat{\mathbf{y}}_t$ at every time step t. A final recognition

sequence is then derived by selecting the character with maximum activation, and the resulting prediction sequence may finally serve as a pseudo-label.

In the case of an AttributeCNN using the PHOC encoding as an attribute representation, deriving a pseudo-label offers various options. Due to the high redundancy prevalent in a multi-level PHOC vector, sharpening is not performed as naturally as in the case of a recognition model. One way to still derive a pseudo-label is to simply consider the network activations directly. In this case, the generated pseudo-label is, strictly speaking, not a PHOC vector anymore, as its entries are not binary. The sigmoid activations at the last layer of the AttributeCNN notably encourage an activation of either zero or one.

Instead of directly taking the network activations for pseudo-labeling, sharpening may be performed by binarizing the prediction vector. Consider a PHOC vector $\hat{\mathbf{a}}$ that has been predicted by an AttributeCNN $\phi(\cdot, \theta^k)$ at some point during self-training. The pseudo-label $\bar{y}$ is derived by binarizing at a fixed threshold of 0.5.

$$\bar{\mathbf{y}} = \bar{\mathbf{a}}, \quad with \quad \bar{a}_i = \begin{cases} 0 & if \quad \hat{a}_i < 0.5 \\ 1 & if \quad \hat{a}_i > 0.5 \\ . \end{cases} \tag{56}$$

Using the network predictions as pseudo-labels directly or in a binarized form disregards any knowledge on the structure of a plausible PHOC vector. Attributes that are present in lower levels must reoccur in higher ones. In order to ensure structural consistency and to further include language information in the training procedure, lexicon-based recognition may be used for pseudo-label generation. Even though the AttributeCNN used in this work is not primarily designed to solve word recognition, it is feasible to do so if a lexicon is available. Let $\mathbb{L} = \{l_0, \ldots, l_N\}$ be a lexicon containing a number of N strings. Each string can trivially be represented by its corresponding attribute representation $\mathbb{L} = \{{}^{l_0}\mathbf{a}, \ldots, {}^{l_N}\mathbf{a}\}$. After the prediction of an attribute representation $\hat{\mathbf{a}} = \phi(\mathbf{x}, \theta)$, word recognition is performed in the fashion of a nearest neighbor search over the lexicon. The recognition result ${}^{*}l$ is the lexicon entry with minimal cosine dissimilarity between the respective attribute vectors.

$$l^* = \underset{l \in \mathbb{L}}{\operatorname{argmin}} \, d_{cos}\left({}^{l}\mathbf{a}, \hat{\mathbf{a}}\right). \tag{57}$$

The final pseudo-label is derived as the attribute representation of the recognition result $\bar{\mathbf{y}} = {}^{l^*}\mathbf{a}$.

Obviously, introducing a lexicon at the training stage introduces further requirements on the applicability. To efficiently perform lexicon-based sharpening, it is necessary that an inaccurate lexicon is sufficient. Aiming at the exploration of new datasets where no annotations are available means that the exact lexicon is unknown. In our experiments, we investigate the use of a language-based lexicon, meaning that similar to the synthesis stage, we use the most common $10,000$ English words as the recognition lexicon. This only introduces the assumption that the language of the target dataset is known before training the model.

4.5 CONFIDENCE ESTIMATION

Using pseudo-labels to train a neural network results in a training dataset of poor label quality, as especially early models may be inaccurate. Nonetheless, prediction

quality differs for various parts of the dataset and it is likely that a small portion of labels are at least partially correct. The goal of confidence estimation in the context of self-training is to identify these correct predictions without knowing the actual annotation. Therefore, a numerical value is required that quantifies how likely it is that a given prediction is correct.

At the training stage, the confidence estimate can then be used to negotiate between *novelty* and *quality* of the pseudo-labeled training set. Removing only few samples with the lowest confidence from the training set or training on all offers the highest variance as the number of training samples remains high. Especially in the beginning of the training process, this will also lead to a training set of low quality as predictions are presumably poor. On the other hand, training on samples exclusively with high confidence is supposed to increase the label quality of the training set, but effectively reduces its size, making it prone to overfitting effects. In summary, the confidence measure is a tool to balance the size and variance of the training set and its label quality over the entire training process.

A confidence measure essentially constitutes a function $c : \mathcal{I}, \Theta \rightarrow \mathbb{R}$ that maps an input image $\mathbf{I} \in \mathcal{I}$ to a real valued number based on a set of model parameters $\theta \in \Theta$. In the literature, most self-training methods for training classification models use the activation of the neuron corresponding to the predicted class as a form of confidence estimate. This is not directly feasible for the retrieval and recognition models of this work, as the prediction tasks are either a multi-class classification or sequential problem. We still follow the idea of using the network activations and derive several confidence measures based either on the predicted attribute representation $\hat{\mathbf{a}}$ or the prediction sequence $\hat{y}$.

Sequence-based Confidences

At each time step of the recognition sequence, the decoder of the sequence-to-sequence model outputs a vector of pseudo-probabilities $\mathbf{d}_t$ over the character set. For each character, we consider its corresponding activation as a numeric estimation of how confident the network is in the respective prediction. In the proposed self-training procedure, a sample is either included in the training set or neglected based on a total confidence value. Therefore, the individual character confidences have to be summarized in a single value quantifying the pseudo probability of the prediction sequence to be correct. Summarizing the character confidence is either performed by taking the mean of the character confidences or the minimal confidence estimate:

$$c_{\text{mean}} = \frac{1}{T} \sum_{0}^{T} \max\left(\mathbf{d}_t\right) \qquad c_{\text{min}} = \min_{T}(\max(d_t)). \tag{58}$$

In order to investigate the potential performance loss of the introduced confidence estimate, we further experiment with an oracle confidence measure. A perfect estimate that fulfills the previously discussed requirements of the training procedure can be easily derived if the annotation is taken into account. From a practical perspective, this is infeasible but allows for the performance drop caused by a suboptimal confidence estimation to be quantified. For the sequence-to-sequence model, we use

the Levenshtein distance between the prediction sequence and its annotation as an oracle confidence estimate:

$$c_{\text{oracle}} = \text{Levenshtein}(\hat{y}, y). \tag{59}$$

Attribute-based Confidences

Retrieval relies on the prediction of an attribute representation $\hat{\mathbf{a}}$. We model each attribute A_i as a binary random variable following a Bernoulli distribution. The output of the attribute CNN is given by $\hat{\mathbf{a}} = \phi(\mathbf{x}, \theta)$. Each element of the output vector is considered an estimate $\hat{\mathbf{a}}_i \approx p(A_i = 1|\mathbf{x})$ for the probability of the i-th attribute being present in the word image $\mathbf{x}$. Here, we only consider PHOC vectors as attribute representations as they are most commonly used.

Similar to the sequence model, a confidence measure may be built directly upon the network's outputs. In case of a PHOC vector, an ideal prediction would give a vector with either a zero or a one for every entry. An attribute is considered to be present in the word image if its pseudo-probability $\hat{\mathbf{a}}_i \approx p(A_i = 1|\mathbf{x})$ lies above a value of 0.5. For defining a confidence estimate, we neglect all attributes that are not present according to the prediction. The final value results from taking the average over all present attributes:

$$c = \frac{1}{|\hat{\mathbf{a}}|} \sum_{\hat{a}_i > 0.5} \phi(\mathbf{x}, \theta)_i \approx \frac{1}{|\hat{\mathbf{a}}|} \sum_{\hat{a}_i > 0.5} p(A_i = 1|\mathbf{x}). \tag{60}$$

While only considering present attributes is a straight forward generalization of using a class activation as a confidence estimate, it neglects most of the information of the predicted PHOC vector. In most cases PHOC vectors and therefore also their estimations are rather sparse. Most of the entries are usually zeros and won't contribute to the previously introduced confidence measure. For an estimate of high confidence it is not just desired that present attribute activations are close to one, but also that entries of absent attributes are zero. This characteristic can be quantified by using entropy as a confidence measure. From the perspective of information theory, entropy measures the amount of information received by observing a random variable [Bis06]. The observation of a random variable with minimal entropy does not hold any information. Therefore, there is no uncertainty about the realization of the random variable. In this case, low entropy corresponds to high confidence in the network's predictions. Following the interpretation of an attribute as a Bernoulli distributed random variable A_i, its entropy is given by

$$H(A_i) = -\hat{a}_i \log(\hat{a}_i) - (1 - \hat{a}_i) \log(1 - \hat{a}_i). \tag{61}$$

To model the confidence of an entire attribute vector, we compute the negative joint entropy over all attributes. We assume conditional independence among attributes. The joint entropy over all attributes is then computed by the sum over the entropies of the individual random variables.

$$\begin{aligned} c &= -H(A_1, \ldots, A_D) = -\sum_{i=1}^{D} H(A_i) \\ &= \sum_{i=1}^{D} \hat{a}_i \log \hat{a}_i + (1 - \hat{a}_i) \log(1 - \hat{a}_i). \end{aligned} \tag{62}$$

Algorithm 2 : Self-Training with Confidence selection

Input : Synthetic data $\mathbf{S} = \{(\mathbf{s}^{(0)}, y_0), \ldots, (\mathbf{s}^{(N)}, y_N)\}$; unlabeled training images $\mathbf{X} = \{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(M)}\}$; number of training cycles K; Model $\phi(\cdot, \theta)$; confidence measure $c(\cdot, \theta)$;

Train initial model $\phi(\cdot, \theta^0)$ on $\mathbf{S}$
for $k \leftarrow 0$ **to** K **do**
  Derive pseudo-labels for $\mathbf{X}$: $\bar{y} = \phi(\mathbf{x}, \theta^k)$ for each element $\mathbf{x}$ in $\mathbf{X}$
  Sort $\mathbf{X}$ w.r.t confidence: $\mathbf{X} = \{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(j)}, \ldots, \mathbf{x}^{(M)}\}$ with $c(\mathbf{x}^{(i)}, \theta^k) > c(\mathbf{x}^{(i+1)}, \theta^k)$;
  Either:
  Select j most confident samples $\mathbf{X}_{conf} = \{\mathbf{x}^{(0)}, \ldots, \mathbf{x}^{(j)}\}$;
  Or:
  Select all samples with $c(\mathbf{x}^{(i)}, \theta^k) > \tau$
  $\mathbf{X}_{\text{train}} = \{(\mathbf{x}^{(0)}, \hat{y}_0) \ldots, (\mathbf{x}^{(M)}, \hat{y}_M)\}$
  $\theta^{k+1} \leftarrow$ train $\phi(\cdot, \theta^k)$ on $\mathbf{X}_{\text{train}}$

Additionally, we also introduce an oracle confidence measure for the AttributeCNN exploiting the annotations. Using string-based attribute representations, such as the PHOC encoding, results in mapping both strings and the network predictions into a common embedding space. A confidence estimate may be defined based on the cosine similarity of the predicted attribute representation and the attribute representation of its annotation. Here, a high dissimilarity corresponds to a prediction of low confidence:

$$c_{\text{oracle}} = d_{\cos}({}^{l}\mathbf{a}, \hat{\mathbf{a}}). \tag{63}$$

Confidence for Pseudo-Label selection

The main requirement to successfully integrate confidence estimation in a self-training scheme is that the following assumption holds true. Let $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ be two word images, with corresponding predictions $\hat{y}_1$ and $\hat{y}_2$ and annotations y_1 and y_2. For a given confidence function c the following relation needs to be generally fulfilled:

$$c(\mathbf{x}^{(1)}, \theta) > c(\mathbf{x}^{(2)}, \theta) \quad \Rightarrow \quad P(\hat{y}_1 = y_1) > P(\hat{y}_2 = y_2). \tag{64}$$

If that is the case, pseudo-labels with high confidence are more likely to be correct compared to low-confidence predictions.

As shown in Algorithm 2, Algorithm 1 can be extended as follows. After making predictions for the entire unlabeled dataset $\mathbf{X}$, a confidence value is determined for each sample. This allows the unlabeled dataset to be sorted with respect to its confidence estimations. Under the assumption that low confidence estimations are highly erroneous, we aim to remove inaccurate pseudo-labels at each training cycle. One option is to define a training schedule and always include a fixed percentage p_k of the unlabeled training data. Manually designing such a selection schedule has the drawback of introducing a number of hyperparameters which need to be tuned, potentially without having a validation set available. On the other hand, it offers

full control of the number $j = p_k \cdot |\mathbf{X}|$ of training samples included at each training cycle.

Alternatively, sample selection may be performed introducing a threshold τ. In this case, the number of hyperparameters is reduced to choosing only a single threshold value. The main drawback with introducing a single threshold is that it is possible that only a limited number of samples surpass the threshold over the course of training. This bears the potential that the model overfits on a small number of confident samples and converges without ever including most of the other pseudo-labeled samples. As shown in the later experiments, this is not the case as the network becomes increasingly more confident in its predictions over the course of training. In this case, novel samples are introduced at each training cycle. Therefore, working with a fixed threshold automatically balances label quality and novelty.

5 EXPERIMENTAL EVALUATION

The theoretical background and empirical results on self-training in other areas give a strong motivation for an application to handwritten document analysis. Following the method discussed in Chapter 4, the concept of self-training can be realized for the presented handwritten word recognition and retrieval models. In this thesis, the effectiveness of the approach is evaluated in an empirical fashion. The conducted experiments shall provide empirical evidence for the following questions that imminently arise regarding the proposed method and its core components.

How can an initial model be derived solely by training on synthetic data? Self-training relies on training an initial model that is then adapted with unlabeled data. Most works train such an initial model on a small labeled dataset. As this already introduces manual annotation effort, our goal is to train an initial model solely on synthetic data. The experiments in Section 5.4.1 show that training an initial model on synthetic data with satisfactory performance is feasible, but requires certain consideration with respect to the synthetic dataset.

Does an adaptation of language properties of the synthetic dataset have direct implications on model performances? Only few works consider the properties of a synthetic dataset and their influence on model performances in the context of handwritten word synthesis. As a well performing initial model is crucial for the success of self-training, several experiments are conducted that consider synthetic datasets with different properties. Section Section 5.4.2 presents a series of experiments showing that training on synthetic data constitutes a form of implicit language modeling and, therefore, the language properties of a synthetic dataset have clear implications for the models.

Can classification networks trained on synthetic data capture visual properties of a target dataset and how does the visual adaption of the synthesis approach influence performances? Adapting the visual properties of a synthetic dataset without labels from the target domain is performed by the proposed classification networks and inferred distributions on fonts and slant angles. The experiments in Section 5.4.3 verify that this calibration approach is able to capture style information which leads to a visually more similar synthetic training set. In general, the common assumption that maximizing the visual variance by basing the synthesis approach on as many fonts as possible is empirically confirmed.

Does self-training allow the considered models to adapt to a target dataset without the need for manually labeled data? The core question of this thesis boils down to showing that self-training closes the performance gap between models trained on synthetic data and models trained in a fully-supervised fashion. As the proposed method does not require manually labeled samples, the resulting model is annotation-free. The experiments in Section 5.4.4 show that the basic implementation of self-training already leads to significant performance gains and well performing models result in the absence of labeled data.

Do the proposed confidence measures capture prediction quality? Generalizing the concept of confidence estimation to the AttributeCNN and sequence-to-

sequence model relies on the introduction of quantitative confidence measures. The main assumption that higher confidence values coincides with higher probability for a correct prediction is crucial. Quantitative and qualitative results of the experiments in Section 5.4.5 show that the proposed measures capture prediction quality and result in an interpretable confidence estimation.

Is a confidence-based selection of pseudo-labels beneficial for self-training? Even though a model may be able to quantify its prediction quality by means of a confidence measure, this does not directly imply that selecting pseudo-labels based on high confidence is beneficial in self-training. Selecting pseudo-labels essentially requires a trade-off between prediction quality and variance to avoid overfitting. The proposed integration of confidence estimation by either using fixed schedules or thresholding is shown to effectively increase performance and robustness of the training scheme. The quantitative results in Section 5.4.5 and Section 5.4.6 give a clear answer to whether to include confidence estimation in self-training and verify the applicability even in the absence of a validation set.

Answering the previously raised question on self-training for handwritten word recognition and retrieval relies on empirical experiments. In order to provide comparable and interpretable results, the method is evaluated on four established datasets with respective benchmark protocols for retrieval and recognition. Details on their characteristics and protocols are provided in Section 5.1. Performances are measured by mean average precision and error rates, which are introduced and discussed in Section 5.2. As the conducted experiments are inherently of random nature, significance testing and the consideration of confidence intervals is important and follows the approach presented in Section 5.3. All quantitative and qualitative results providing answers to the key questions are discussed in detail in Section 5.4.

5.1 DATASETS

In order to provide empirical evidence for the different methods presented in this work, experiments are conducted on four different benchmark datasets. All datasets are established benchmarks in the document analysis community. Following the respective evaluation protocols allows for a direct comparison with other works in the literature. While all datasets consist of handwritten word images, their appearance strongly varies across the different benchmarks. See Figure 23 for a selected examples of all datasets. By considering documents of contemporary as well as historic nature that are written by a single or multiple writers, we aim at underlining the general applicability of the proposed approach. Furthermore, the datasets show significant differences in the size of the available training and testing material, which represents the challenge in a real world application. Beside the presented benchmarks for evaluation, a publicly available synthetic dataset serves as an additional baseline.

George Washington Letters

The George Washington letters constitute a historic document collection that is based on images from the George Washington papers at the US Library of Congress [Was54]. It resulted from the digitization of letterbooks capturing the correspondence of George Washington written in English. A first definition of the benchmark in the

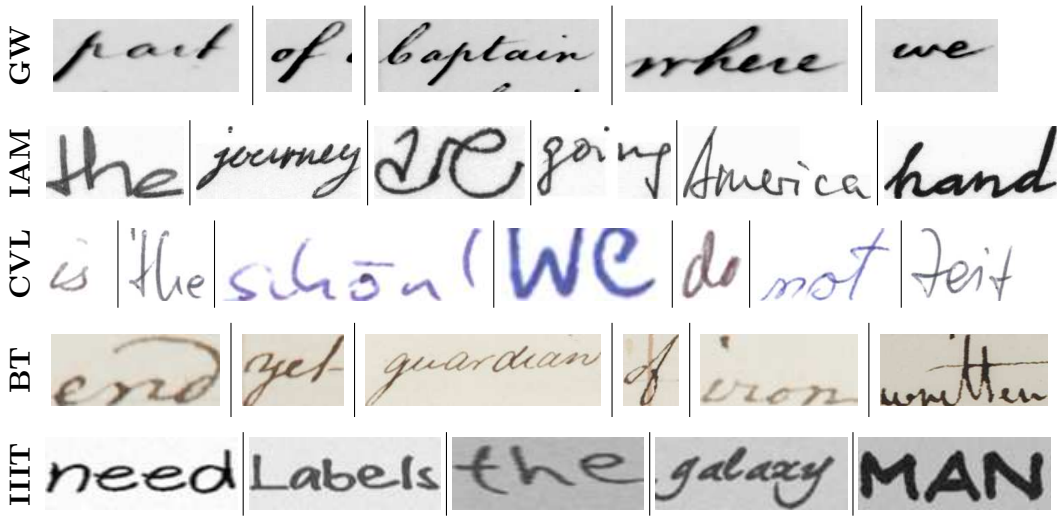


Figure 23: Example word images from the historic (GW, BT), contemporary (CVL, IAM) and synthetic (IIIT) datasets.

context of document image analysis was published in [KLP01]. Later, a small excerpt of the document collection was established as the defacto standard benchmark for evaluating word recognition and retrieval methods [Gio+17]. As proposed in [LRM04; RM07], twenty pages were selected from a designated letterbook. This letterbook is a handwritten copy of the original [SF16] and, therefore, it is commonly agreed that all pages were written by a single writer. The general appearance of words and writing style is rather homogeneous. 4,860 words were manually segmented and annotated, constituting a total of 1,124 unique strings. Due to its long standing tradition as an evaluation benchmark, multiple splits and evaluation protocols exist for word retrieval and recognition.

For recognition, we follow the protocol established in [Tol+17; Dut+18]. The dataset is split into training, validation and test according to the first partition proposed in [Fis+12]. Capitalization and punctuation is considered as annotated.

Retrieval is evaluated following the protocol first described in [Alm+14]. All strings are mapped to their lowercase correspondence and punctuation is removed. The twenty pages are split into fifteen pages for training and five pages for testing. Splitting is performed four times with the same selection as in [Alm+14]. The final performance measure of the four-fold cross validation results from the mean over the four test cases. In the case of query-by-example, each word image of the test set serves as query once. Similarly for query-by-string, each unique string of the test set is considered as a query.

Jeremy Bentham Manuscripts

The Jeremy Bentham Manuscripts is a historic document collection written in the 18th and 19th century [LM81]. The documents contain works on philosophy, law and morality authored by Jeremy Bentham. In a large crowdsourcing effort, the *Tran-*

scribe Bentham project has been creating manual transcriptions for the collection [CW12; CT14; Cau+18]. By March 2023 a total of 30,540 pages had been entirely transcribed. As the documents were not exclusively written by Jeremy Bentham himself but also by a number of secretaries, the writing style changes considerably.

In 2014, an excerpt of 50 pages was established as a keyword spotting benchmark in the *ICFHR2014 Competition on Handwritten KeyWord Spotting* [Pra+14]. The subsequent competition at ICDAR 2015 [PTV15a] was based on the same collection containing a larger number of 70 document images. Both competitions define segmentation-based protocols for query-by-example word spotting. Query words are taken from a predefined list that has been published alongside the competition. The respective datasets are not entirely annotated but only provide a list of words that are relevant with respect to a given query. Therefore, a numeric evaluation of query-by-string is not feasible even though the proposed model has the capability. While [Pra+14] uses a different relevance criteria for computing the mean average precision, we stick to the standard performance measure as described in Section 5.2. A comparison to the literature is still possible as most methods have been reevaluated under the standard approach in [ZPG17].

Even though no classic recognition benchmark exists for the Bentham Manuscripts it still has been used in this fashion in [Mat+21a]. Based on [Vil+16], the authors propose a question answering benchmark and therefore created the BenthamQA dataset with word level annotations. They also report recognition results of a system entirely trained without training material from the target domain to which we can draw a direct comparison.

IAM Handwriting Database

The IAM Handwriting database was specifically created with the goal of establishing a benchmark for handwriting recognition tasks [MB02]. For this purpose, forms were created that contain text from the Lancaster-Oslo/Bergen corpus [JLG78]. These forms were handed out to different writers which were instructed to copy the given text in their own handwriting. A total of 400 different persons took part in the creation of the dataset resulting in 1066 pages of modern handwriting. Due to the artificial creation process, the images are of high quality and contain comparably clean handwriting on a light background. The challenge of the dataset lies especially in the large variation of writing styles resulting from the numerous different writers. The dataset was rapidly adopted by the document analysis community and has served as a handwriting recognition benchmark up to today.

We follow the predominantly used evaluation protocol as described in [MB02]. Evaluation is considered to be case sensitive and the test set also contains punctuation marks. An additional challenge of the dataset is posed by the fact that each writer either contributed to the training or to the test split. The recognition model does not have access to annotated training material for a specific writer and is, therefore, required to be writer independent.

Due to its large scale and challenging characteristics, the IAM database was also adopted as a word spotting benchmark. The most commonly used protocol was defined in [Alm+14]. The authors removed all words that have been marked to have a dubious annotation. These marks were created during the annotation process,

whenever the handwritten text may diverge from the text prompt on the given form. Each word image of the test set serves as a query once, except the official list of stop words. For query-by-string, each unique string from the test set is used as a query.

CVL Database

For the creation of the CVL Database published in [Kle+13], a predefined text was given to different writers which was then copied in handwriting. In contrast to the IAM, the selection of texts is significantly smaller as only seven different texts were used. It is noteworthy that two of these texts are in German while the rest are written in English. For all later experiments, we ignore this fact and follow the exact same procedures as for the datasets entirely written in English. The focus of the dataset was especially the evaluation of writer identification and writer retrieval tasks. Therefore, a total of 311 different writers contributed to the dataset. An additional specificity of the dataset is its uncommon split in training and testing. The CVL training set is smaller than the test set by a factor seven.

Even though it was created for the purpose of writer retrieval, the CVL database may be used for the evaluation of handwriting recognition as word level annotations are available. We follow the same protocol as [Kan20; Abe+21] to allow for a fair comparison. Small and capital letters are considered different symbols and punctuation is included as annotated.

IIIT-HWS

In the literature on handwritten document analysis, using synthetic data to pretrain models is common practice. Nonetheless, the actual synthetic datasets are rarely publicly available. One of the first published synthetic datasets of handwritten word images is the IIIT-HWS dataset first described in [KJ16]. The dataset was the foundation for a line of research concerned with learning word image representations for recognition and retrieval [KDJ18; KJ19; KDJ23]. It was further picked up by the community and was also used for pretraining AttributeCNNs in [GSF18]. The dataset consists of a total of nine million word images which were rendered from a vocabulary consisting of $90,000$ words. Different variants of capitalization were applied with either all, none, or the first characters being in uppercase. A diverse set of TrueType fonts was used that generally resembles handwriting. The authors define two variants of the dataset with IIIT-HWS10K that only uses a subset of $10,000$ words opposed to the IIIT-HWS90K which includes the entire dataset. In this thesis, both versions are used as an example for a generic synthetic dataset from the literature without any model or domain specific considerations.

5.2 PERFORMANCE MEASURES

Comparing different models in order to draw conclusions on the methodological contribution is easiest when model performance can be summarized in a numeric performance measure. Several measures are established in the literature making them a natural choice for the following experimental evaluation and allow for a direct comparison. Retrieval systems are usually evaluated and compared in terms

of *mean average precision*, while text recognition systems rely on *character* and *word error rates*.

Mean Average Precision

In general, a retrieval system returns an ordered list of items with respect to a given query. The elements in this list can either be relevant or irrelevant. Considering a word spotting application, the query represents a word which is supposed to be retrieved from a document collection. Usually, a retrieval list item is only considered relevant if the annotations of the item itself and of the query are equal. The perceived quality of a retrieval list depends on several factors. Recall is the number of retrieved relevant items over all items from the dataset relevant to the query. Mathematically, it can be defined as follows [BR11, pp. 135–136]. Let $\mathbf{D}$ be a dataset of size J and $\mathbf{R_q}$ a retrieval list of size K with respect to a given query q. In general, $\mathbf{R_q} \subset \mathbf{D}$ and $K < J$ hold true. The relevance criteria can be summarized as a function $rel : \mathbf{D} \rightarrow \{0, 1\}$ that maps an element to one if it is relevant with respect to the query and zero elsewise. With $d_k \in \mathbf{D}$ and $r_k \in \mathbf{R_q}$, the recall r is defined as

$$r(\mathbf{R_q}) = \frac{\sum_1^K rel(r_k)}{\sum_1^J rel(d_k)}. \tag{65}$$

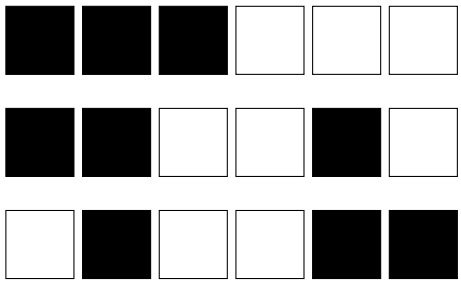
Therefore, the recall describes the fraction of the number of relevant items in the retrieval list to the number of relevant items in the entire dataset. Despite the goal of retrieving all elements from the dataset, the fraction of relevant items and the length of the list is desired to be high. This property is summarized as the precision of a retrieval list [BR11, pp. 135–136]:

$$p(\mathbf{R_q}) = \frac{\sum_1^K rel(r_k)}{K}. \tag{66}$$

In the case of segmentation-based word spotting, the pure computation of recall and precision does not allow for any insights in model performance. Usually, the system returns all elements of the dataset which boils down to a recall of one for any given query. As precision also does not depend on model performance in this case, a numeric measure is required that captures how well the provided list is sorted. The expectation of the user is that relevant items occur in the beginning of the list while irrelevant elements remain at the end. This desired characteristic is captured by the average precision. Considering the retrieval list to be a binary list of either relevant or irrelevant items, the mean average precision relies on the computation of the precision over different portions of the retrieval list. Specifically, the precision is calculated at different positions k whenever the recall level changes. With p_k being the precision of the list up to the k-th element, the average precision is defined as

$$AP(\mathbf{R_q}) = \frac{\sum_1^K p_k \cdot rel(r_k)}{\sum_1^J rel(d_k)}. \tag{67}$$

Figure 24 visualizes a toy example of a retrieval list of size six with three relevant items. A perfect ordering would correspond to an average precision value of 1. More relevant items at the end of the retrieval list result in a lower numeric value of the average precision. This captures the desired characteristic and perceived quality of a



$$AP = \frac{1}{3} \cdot [(\frac{1}{1} \cdot 1) + (\frac{2}{2} \cdot 1) + (\frac{3}{3} \cdot 1)] = 1.0$$

$$AP = \frac{1}{3} \cdot [(\frac{1}{1} \cdot 1) + (\frac{2}{2} \cdot 1) + (\frac{3}{5} \cdot 1)] = 0.87$$

$$AP = \frac{1}{3} \cdot [(\frac{1}{2} \cdot 1) + (\frac{2}{5} \cdot 1) + (\frac{3}{6} \cdot 1)] = 0.47$$

Figure 24: Example of the computation of the *mean average precision*. Each row represents a retrieval list with black squares being relevant with respect to a query.

retrieval list. To compare different retrieval systems, the average precisions over the query set are averaged resulting in the standard performance measure mean average precision [BR11, pp. 140–141].

Character and Word Error Rate

The standard evaluation metrics for text recognition are character and word error rates. A word can be considered a sequence of characters from a predefined alphabet. The *character error rate* is based on the comparison of two sequences with the Levenshtein distance. The Levenshtein distance is an edit distance that is defined by the number of operations that are required to transform one sequence into another [BR11, p. 222]. Possible operations are substitution (S), insertion (I) and deletion (D). For a given word with an annotation of length N, the *character error rate* is computed as

$$CER = \frac{S + I + D}{N}. \tag{68}$$

In the same way, the word error rate can be defined for a sequence of words, for example that which results from a text line-based recognition system. As in this work only individual words are considered, the *word error rate* simplifies to the fraction of wrongly predicted words. With respect to the Levenshtein distance, a wrong prediction can be considered a required substitution. Therefore, we can define the *word error rate* over a number of N words as

$$WER = \frac{S}{N}. \tag{69}$$

5.3 SIGNIFICANCE TESTING AND INTERVALS

The experiments in this work are in most cases the result of combining numerous stochastic processes. Data synthesis relies on randomly sampled parameters. Augmentation methods randomly introduce variations and also the optimization and initialization of neural networks involve stochastic processes. In order to account for the random nature of the presented experiments and in order to extend the interpretability of the results, significance tests are performed for all retrieval experiments. Furthermore, we provide confidence intervals for the recognition experiments.

In the area of word spotting, it is common practice to summarize and compare performances of a system based on the mean average precision. Especially when test benchmarks and the numerical differences in performances are small, it is not obvious if the observed difference results from methodological superiority or mere chance. This is of special interest when different configurations of a method are compared motivating specific design choices. Answering the question whether a change of in a method results in significant performance gains requires more than a single quantitative performance measure. As proposed in [SF18], running a permutation test [SAC07] gives further insights into the statistical significance of a performance difference. Comparing two retrieval systems is based on the average precisions with respect to a set of queries. As the results in the literature only report single mean average precision values, we are only able to run permutation tests for the evaluation of the models developed in this thesis.

Consider two word spotting models that are evaluated on a given benchmark with a predefined set of queries. In this case, both systems produce a retrieval list for each query and performances can be summarized in terms of average precision. As a result of this process, we observe two mean average precisions. Hence, a permutation test is supposed to determine if the two systems are significantly different. Therefore, the null hypothesis is formulated, which is that both systems are actually identical [SAC07]. If both systems are identical any permutation of the resulting set of average precisions is equally likely. Based on this assumption, the *p-value* or *significance level* can be computed. The exact p-value is the number of permutations that resulted in a mAP difference greater than the observed value for the original system divided by all possible permutations. In a practical application, computing all possible permutations is usually not feasible as the number of permutations grows exponentially with the number of queries. This makes the approximation of the p-value necessary. Monte-Carlo sampling of random permutations allows an estimate $\hat{p}$ of the actual *p-value* to be derived. It is desirable to quantify how accurate this estimation is in order to allow for an interpretation of the significance test. In other words, the question is how many permutations are required so that the standard deviation of the approximated *p-value* $s_{\hat{p}}$ is small. As discussed in [SAC07], given a described standard deviation, the required number of random permutations can be derived. With p being the real p-value, the number of random permutations k is directly related to the standard deviation $s_{\hat{p}}$ of the estimate:

$$s_{\hat{p}} = \sqrt{\frac{p(1-p)}{k}} \quad \Rightarrow \quad k = \frac{p(1-p)}{s_{\hat{p}}^2}. \tag{70}$$

Given the desired standard deviation, an upper boundary for the number of required permutations can be computed. With $p = 0.5$, k describes the number of random permutations that result in an estimated p-value with a standard variation equal or smaller than $s_{\hat{p}}$:

$$k = \frac{1}{4s_{\hat{p}}^2}. \tag{71}$$

In the later experiments, we follow the same parameterization as in [SF18; Rot19]. A precise approximation of the *p-value* with a maximal standard deviation of $s_{\hat{p}} = 0.0001$ requires $k = 250,000$ permutations. During random sampling, it is ensured that each permutation is unique. Two models are considered significantly different

if the estimated *p-value* lies below the significance level of $\alpha = 0.05$. If a result is significantly better or worse than its respective baseline, its mAP values will be noted in italics.

All recognition experiments are quantitatively evaluated in terms of error rates. Due to the random nature of the conducted experiment, the provided error rate can be only considered an estimate of the actual error rate. Typically, recognition tasks are considered Bernoulli processes that follow a binomial distribution [BCD01]. An observed word accuracy describes an estimate $\hat{p}$ based on the fraction of correctly classified patterns k over the total number of all patterns n with $\hat{p} = \frac{k}{n}$. If n is large enough, the Wilson interval [BCD01] can be derived from solving the following quadratic equation:

$$(n + c^2)p^2 - (2k + c^2)p + \frac{k^2}{n} = 0. \tag{72}$$

The value c describes the probability with which the actual accuracy lies in the confidence interval. Here, $c = 1.96$ corresponds to a probability of 95%. The final confidence interval is derived by computing the solutions of Equation 72 given by

$$p_{u/o} = \frac{2k + c^2}{2(n + c^2)} \pm \sqrt{\left(\frac{(2k + c^2)}{2(n + c^2)}\right)^2 - \frac{k^2}{n(n + c^2)}}. \tag{73}$$

5.4 RESULTS

In this section, a series of experiments is presented that shows the effectiveness of the proposed self-training method and tries to provide insights into the raised researched questions. Section 5.4.1 to Section 5.4.3 are concerned with training a model on synthetic data which then serves as an initial model for the latter self-training process. Therefore, we first train both models on a generic synthetic dataset generated by the process described in Section 4.3 and compare performances to two variants of a synthetic dataset from the literature. To gain further insights into whether the language properties of the synthesis procedure influences performance, we conduct a set of experiments using different synthesis labelsets in Section 5.4.2. Section 5.4.3 provides evaluations with respect to the visual adaptation of the synthesis procedure.

After deriving an initial model, self-training offers the potential to adapt to a target datatset. Section 5.4.4 provides empirical evidence that this adaptation is feasible from an initial model purely trained on synthetic data. Furthermore, different strategies for deriving pseudo-labels and their influence on performances are compared.

As discussed in Section 4.5, confidence estimation in the context of self-training aims at a beneficial selection of pseudo-labels. In Section 5.4.5, it is shown that the proposed confidence measures quantify prediction quality. The quantitative evaluation motivates the integration of confidence estimation into the self-training process by either using fixed portions of the datasets (see Section 5.4.5) or via thresholding (see Section 5.4.6). Finally, we compare the resulting performances of the retrieval and recognition models to results from the literature in Section 5.4.7.

Table 2: Comparison of retrieval performances resulting from training on different synthetic datasets. All results in mAP [%]. Permutation tests run with respect to our dataset. Significantly different results highlighted in italics.

Dataset	GW		IAM		BT_{14}		BT_{15}	
	QBE	QBS	QBE	QBS	QBE	QBS	QBE	QBS
IIIT-10K	*51.42*	*63.36*	*14.54*	*36.36*	*30.50*	–	*18.58*	–
IIIT-90K	*50.48*	*61.63*	*13.66*	*34.80*	*25.58*	–	*15.35*	–
Ours	**74.69**	**80.44**	**57.78**	**74.50**	**79.77**	–	**70.09**	–

Table 3: Comparison of recognition performances resulting from training on different synthetic datasets. All error rates in [%].

Dataset	GW			IAM			CVL			BT_{QA}		
	CER	WER	±	CER	WER	±	CER	WER	±	CER	WER	±
IIIT-10K	42.61	82.26	2.2	36.10	74.59	0.6	37.43	77.23	0.3	56.38	94.96	0.2
IIIT-90K	44.45	81.19	2.2	32.80	70.25	0.6	35.51	74.53	0.3	54.34	94.00	0.2
Ours	**17.77**	**45.02**	2.9	**21.09**	**48.39**	0.7	**28.19**	**58.13**	0.3	**23.53**	**52.29**	0.4

5.4.1 *Training on synthetic data*

In order to derive an initial model as a starting point for self-training, we train the retrieval and recognition models on synthetic datasets. Pretraining on synthetic data has been investigated in several works in the literature. Nonetheless, only few datasets are published due to the commonly large size of synthesized datasets and their random and possibly infinite nature. One exception is the previously introduced IIIT-HWS dataset. It allows for a direct comparison of models trained on a dataset generated by the proposed synthesis approach.

In the first step, a synthetic dataset is generated using the pipeline described in Section 4.3. A total of three million word images are synthesized using all 396 available fonts and the Gutenberg corpus as a labelset. In this case, the distribution of words follows their natural occurrences. For a quantitative evaluation, we trained both the AttributeCNN and the sequence-to-sequence recognition model on the synthetic data until convergence. In this case, training all models for one epoch on the synthetic data was sufficient due to the number of samples and the comparably low variance in the synthetic dataset. As is common practice, the AttributeCNN considers all annotations to be case insensitive for training and evaluation. The recognition model considers upper and lower case characters as individual symbols and also considers punctuation. Following the same setup, models for two variants of the IIIT-HWS dataset are trained. The 10K version includes one million images and, therefore, considerably less word images than our dataset, while the 90K ver-

sion is larger with nine million images. Table 2 and Table 3 present the resulting performances on all benchmarks.

Considering retrieval performance, mean average precision values up to 80% in the case of query-by-string on George Washington can be achieved, despite the fact that the model has never seen representative samples from the target dataset. In general, considerable performances are achieved on all benchmarks, motivating the suitability of the model as an initial model for self-training. Comparing the different datasets, model performances reflect the presumed difficulty of the tasks, with higher performances on more homogeneous and smaller datasets, such as George Washington and Bentham, compared to the IAM database. Recognition performances paint a similar picture with character error rates around 20% and word error rates around 50%. Naturally, supervised performances exploiting the availability of the training set achieve higher performances.

An noteworthy observation is that the models trained on the proposed dataset strongly outperform their respective counterparts trained on the IIIT-HWS datasets. A closer look at the synthesis procedure of the IIIT-HWS datasets offers potential explanations [KJ19]. The approach differs especially in the derivation of a label set. In case of IIIT-HWS, a unique set of labels from the Hunspell dictionary is used. Rendering a fixed number of images per label leads to a uniform distribution of words. Furthermore, all words are rendered in different capitalization variants with all letters either being capitalized, lower or only the first letter in upper case. As the model presented in [KJ19] only uses the synthetic data in a pretraining stage, the authors motivate this design choice with the argument that the model should be case-insensitive. The observed performance degradation indicates that a more natural word distribution and capitalization seems to improve performances.

In terms of dataset size, only marginal performance differences can be observed comparing the two variants of the IIIT-HWS datasets. Additionally, both of these datasets are clearly outperformed by the proposed dataset of three million word images, leading to the conclusion that the sheer addition of more samples does not improve performances.

5.4.2 *Implicit language modeling*

One of the main questions when designing a synthesis pipeline is which strings shall be used for generation. If the considered models only learn independent visual information for letters, the choice of the labelset should only have marginal influence on performance. This assumption is likely to be false, as it is presumed that highly parameterized neural models also learn language information in the form of an implicit language model [SRN17]. As indicated in the initial experiments with the IIIT-HWS datasets in Section 5.4.1, a careful choice of a labelset is likely to be a crucial factor for performance.

In order to investigate the effect of the choice of labels, the following experiment is conducted. Four datasets with three million word images are synthesized following the exact same process using all available fonts. Each dataset only differs in the labels that represent the available language information.

Table 4: Comparison of retrieval performances resulting from training on synthetic datasets with varying labelsets. All result in mAP [%]. Permutation test were run with respect to the Gutenberg labelset. Significantly different results highlighted in italics.

Labelset	GW		IAM		BT_{14}		BT_{15}	
	QBE	QBS	QBE	QBS	QBE	QBS	QBE	QBS
Oracle	**84.56**	**89.74**	**61.03**	**76.58**	**85.06**	–	**73.19**	–
Gutenberg	74.69	80.44	57.78	74.50	79.77	–	70.09	–
Vocabulary	*68.97*	78.86	*41.02*	*67.45*	*67.40*	–	*59.57*	–
Randomize	*66.45*	76.04	*44.07*	*66.84*	*70.30*	–	*60.93*	–

Table 5: Comparison of recognition performances resulting from training on synthetic datasets with varying labelsets. All error rates in [%].

Labelset	GW			IAM			CVL			BT_{QA}		
	CER	WER	±	CER	WER	±	CER	WER	±	CER	WER	±
Oracle	20.07	58.42	2.8	**17.86**	**38.81**	0.7	**6.74**	**12.72**	0.2	44.83	79.30	0.3
Gutenberg	**17.77**	**45.02**	2.9	21.09	48.39	0.7	28.19	58.13	0.3	**23.53**	**52.29**	0.4
Vocabulary	33.65	83.42	2.1	29.49	63.65	0.7	35.42	73.81	0.3	46.01	87.78	0.2
Randomize	45.02	88.49	1.8	28.89	62.62	0.7	35.08	71.54	0.3	63.47	93.83	0.2

In order to derive a natural text corpus, the top 100 ebooks available from Project Gutenberg [1] were downloaded. This yielded a text corpus of approximately 14 million words. The text corpus includes punctuation, lowercase and capital letters and all distributions follow the occurrence patterns found in natural text from English literature.

The presented experiments aim at removing or inserting additional language information at the synthesis stage. In the experimental setup, it is feasible to define a labelset that perfectly represents the target datasets by using their respective annotations. From a practical application perspective, it is usually not possible to obtain such an oracle labelset.

In order to remove language information, a vocabulary is extracted from the Gutenberg corpus and a fixed number of word images is generated per label. This approach is similar to the generation process of the IIIT-HWS datases as discussed in Section 5.4.1. In this case, models which are trained on the resulting dataset do not have any information on the distribution of word occurrences. The extreme case which tries to remove as much language information as possible uses randomized strings as described in Section 4.3. Generating randomized strings also removes any n-gram distribution from the synthesized dataset. Retrieval and recognition models are trained for one epoch on each of the four datasets. See Table 4 and Table 5 for

1 https://www.gutenberg.org/ Accessed on 14.02.2024.

the resulting performances. For retrieval, all permutation tests are run with respect to the experiment on the Gutenberg dataset.

Focusing the synthesis labels on the annotations of the actual datasets leads to significant performance gains for the retrieval model. Improvements are comparably larger for a small dataset such as George Washington in contrast to the IAM dataset indicating that the choice of the labelset introduces a strong bias to the model. For a homogeneous dataset with a quite limited test set, this bias actually improves performances quite significantly. For the recognition model, the experiments offer a more diverse interpretation. In the case of George Washington and Bentham, a drop in performance can be observed. Opposed to the AttributeCNN, the sequence-to-sequence model seems not to benefit from the introduced language bias. The strong sequential modeling capacity of the recurrent architecture is likely to overfit on the rather limited label set, which yields a drop of performance. This is not the case for the IAM database with a larger and more diverse labelset, for which a performance gain similar to that of the retrieval case can be observed. The CVL database presents an interesting outlier with a striking jump in performance. This is easily explainable by the dataset characteristics. On the one hand, using the annotations includes the occurring German labels which are not part of the Gutenberg corpus. On the other hand, the diversity of text in the CVL database is extremely limited, allowing the model to highly benefit from the introduced bias.

The results show that removing language information generally leads to a drop in performance. Already removing the word distribution by using the vocabulary-based dataset results in significant performance drops on almost all benchmarks for both models. This also offers a potential explanation for the poor performances of the IIIT-HWS dataset in the initial experiments in Section 5.4.1. Considering the randomization approach, performances generally decrease compared to the vocabulary dataset, but the influence is comparably lower. This indicates that information on word distributions is increasingly important compared to n-gram information. Interestingly, in some cases, slight performance gains can be observed when comparing the randomization to vocabulary models. This is especially the case for the pure visual retrieval in case of query-by-example. The respective model seems to benefit more from the visual diversity of the dataset than it is hindered by the lack of language information.

In summation, the experiments clearly show that training a model on synthetic data can be considered a form of language modeling. This is in line with the literature on implicit language modeling [SRN17], but in contrast to training a supervised model, the following problem arises. For supervised training, language information is modeled by the collection of a representative training dataset while for synthetic data it is freely chosen independently from the visual generation process. The experiments indicate that focusing the language information towards a specific dataset may benefit performance but offers the risk of introducing harmful biases. In general, synthesizing from natural text, including word distributions according to their real world occurrences, offers a robust and easily available basis for synthetic data generation.

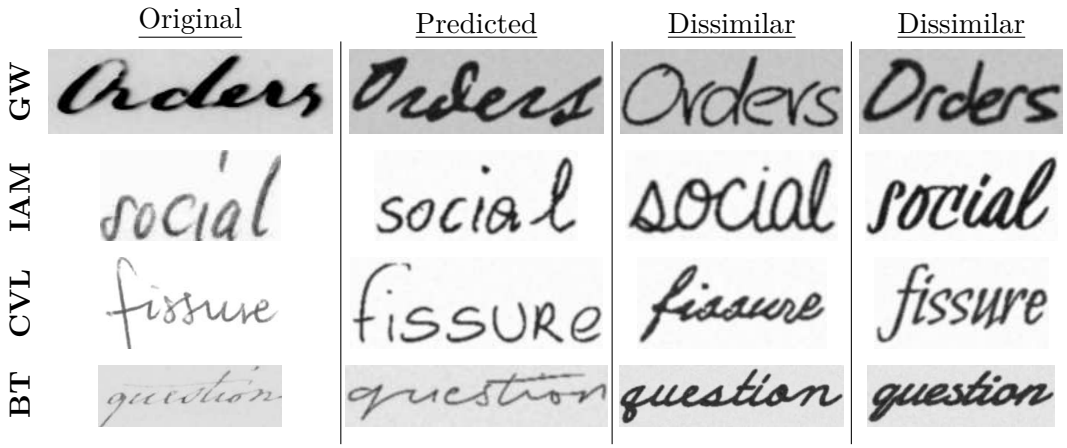


Figure 25: Examples of style adapted word image synthesis. First column shows original images from each dataset. Second column shows images generated using the font predicted by the classification network. Column three and four use the two least frequently predicted fonts for each dataset. The style is likely to be dissimilar to the original image.

5.4.3 *Synthesis Calibration*

Besides the selection of a suitable labelset, a synthesis process is mainly defined by the function $f : \mathcal{Y} \rightarrow \mathcal{I}$ that maps a label y to a word image $\mathbf{I}$. In the literature and also in the previous experiments, word images are often rendered by randomly drawing from a predefined set of fonts. The choice of fonts is entirely arbitrary in most cases and relies on the fact that respective fonts have been labeled at some point as handwritten. Commonly, the set of fonts is chosen to be as big and as diverse as possible. It is likely that a large portion of fonts actually shows a huge style difference with respect to the target dataset.

In the following experiments, the feasibility of determining a beneficial selection of fonts is investigated in addition to whether increasing their numbers benefits performances. Furthermore, the distribution of slant angles is adapted following the same strategy. We follow the adaptation strategy presented in Section 4.3.1 and use font and slant classification networks that have been entirely trained on synthetic data. These style classifiers are then used to derive distributions of fonts and slant angles by inference on the actual target datasets. Here, the main assumption is, if the font classifier predicts a certain synthetic font for a given real word image, that this font is likely to be visual similar. If this assumption holds true, the resulting distributions of fonts and slants should result in a more representative dataset when the distributions are integrated in the synthesis process.

After inferring the distributions of fonts and slants for all four datasets, first conclusions can be drawn from a pure qualitative inspection. Figure 25 shows a selection of samples from the actual datasets with corresponding synthetic word images. In all cases, the synthetic image is generated with the most often predicted slant angle. For each of the real world samples, a synthetic version is generated using the font predicted by the font classifier. Additionally, synthetic images using the two least often predicted fonts for each dataset are shown. The given examples show that

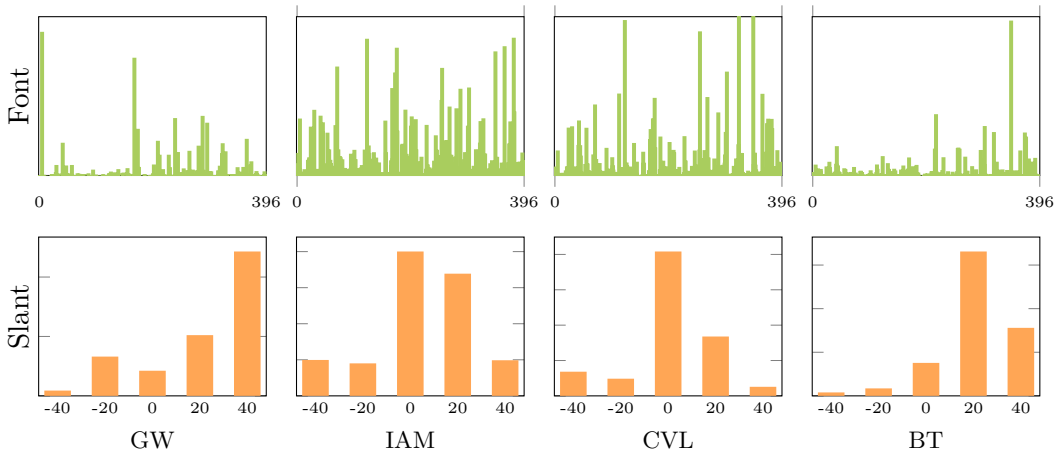


Figure 26: Overview on the histograms derived from predicting fonts and slant angles for the given datasets. For visualization purposes, the absolute frequencies were normalized to the interval $[0, 1]$.

exploiting the style classification networks yield synthetic samples that are visually more similar to an actual training sample. This underlines the assumption that font classification can be used to identify the most similar synthetic font for a given word image.

Besides looking at the adaptation process on the level of individual word images, the distributions can also be considered from a dataset perspective. Figure 26 presents the font and slant angle histograms for all four datasets. The goal of the adaptation process is that the given histograms approximate distributions that encapsulate the visual characteristics of each dataset. As shown in the first row of the figure, it can be argued that this is the case as the font distributions clearly reflect the number of writers in each of the datasets. Different writers in a dataset usually result in highly diverse writing styles. In cases such as the IAM and CVL databases, where a high number of different writers contributed to the collection of data, we observe the frequent prediction of multiple fonts. This stands in a clear contrast to the historic datasets of George Washington and Bentham with one or few contributing writers. In this case, the histograms are clearly dominated by individual fonts.

Considering the slant angle, the dataset characteristics are reflected by the histograms shown in the second row of the figure. Again, a difference between the modern and historic datasets can be observed.

For IAM and CVL, a frequent prediction of the angles of 0° or 20° can be observed, corresponding to a rather upright writing style. With respect to George Washington and Bentham, both datasets contain a writing style with rather strong slant angles. This is perfectly represented by the derived histograms with 20° and 40° as clear peaks. All histograms reflect the tendency of the writers to write upright or with a slant to the right as negative angles are only rarely predicted.

In general, the interpretation of the histograms of fonts and slant angles corresponds to the qualitative perception of the different datasets. The use of the proposed slant and font classifiers allows the visual properties of the datasets to be encoded, motivating an adapted synthesis process.

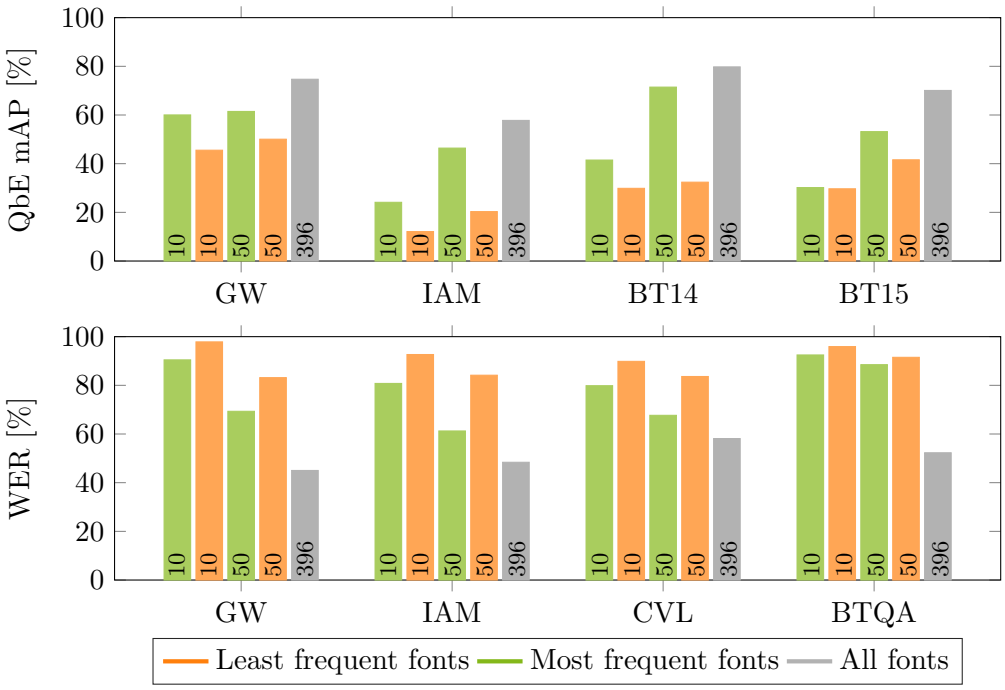


Figure 27: Comparison of query-by example retrieval performances in mAP (upper half) and recognition performances in WER (bottom half) on adapted synthetic datasets. Orange indicates performances derived from datasets using the least frequently predicted fonts. Green represents datasets generated with the most frequently predicted fonts. Gray corresponds to using all fonts. The number on each bar shows the number of fonts used for synthesis

Up to this point, the experiments only indicate that using the style classification networks results in word images with a style similar to the respective dataset. This does not necessarily mean that training a model on an adapted dataset leads to performance gains. The following experiment is conducted in order to quantitatively evaluate the influence on performance using an adapted synthesis process. For each benchmark, multiple datasets are generated, always using the Gutenberg corpus and with a total number of three million word images per synthetic dataset. During synthesis, the slant angle is always drawn from the distribution approximated by the histogram that has been derived by the slant classification network. In order to answer the question whether fonts that are more frequently predicted by the font classifier are better suited for generation, the following approach is proposed. Synthetic datasets that only use the ten or fifty most frequently predicted fonts of the target dataset are created. Using the same synthesis process, corresponding datasets for the least predicted ten and fifty fonts are generated. Additionally, we compare performances to using all fonts with a uniform distribution for synthesis. In all cases, we train the retrieval and recognition models for one epoch on the different synthetic datasets and compare performances in terms of query-by-example and word error rates on the target benchmarks.

See Figure 27 for a comparison of the resulting performances. Considering the models trained on a limited number of fonts, a clear superiority of the adapted

Table 6: Comparison of retrieval performances from self-training on all unlabeled samples using different pseudo-label generation strategies. All result in mAP [%]. Permutation tests were run with respect to the baseline which is the performance of the initial model. Significantly different results highlighted in italics.

Method	GW		IAM		BT_{14}		BT_{15}	
	QBE	QBS	QBE	QBS	QBE	QBS	QBE	QBS
Baseline	74.69	80.44	57.78	74.50	79.77	–	70.09	–
Sigmoid	*86.61*	*70.76*	*45.24*	*32.54*	80.32	–	*59.22*	–
Binarized	*78.44*	*62.43*	*34.94*	*33.19*	*35.64*	–	*37.91*	–
Lexicon	**93.78**	**92.71**	**80.64**	**85.91**	**96.38**	–	**77.56**	–

datasets is shown. In all cases, the models trained on the more frequently predicted fonts outperform the models trained on the least predicted ones. This observation encourages the previously made assumption that the set of handwritten fonts contains writing styles that are better or worse suited for data synthesis. Here, the prediction frequency of the font classifier is an indication of which fonts to use during synthesis. In the experiments, we also observe performance gains when the variance inside the dataset is increased by using more available fonts. This characteristic is less notable for a rather homogeneous dataset such as George Washington. Using all 396 fonts that are considered in this thesis results in the best performing models. This leads to the conclusion that even though visually dissimilar fonts are included during data synthesis, the additional variance benefits performance.

In summary, the conducted experiments show that a visual adaptation of the synthesis approach is feasible by using font and slant classification networks trained on synthetic data. This does not introduce any additional label effort and allows visual properties of a target dataset to be encoded by its inferred distributions. By that, more or less similar fonts from a style perspective can be identified. Nonetheless, simply choosing a large number of fonts and using a uniform distribution seems to be a valid strategy as the inclusion of less frequently predicted fonts does not hurt performances and provides a form of additional regularization.

5.4.4 *Adaptation via self-training*

The previous experiments presented several considerations on how to generate synthetic data for training a word recognition or retrieval model. Even though considerable performances can be achieved, naturally they are still far behind performances of models trained on labeled data. In a given application scenario, especially when challenging datasets are targeted, performance is possibly insufficient. Starting with an initial model trained on synthetic data, self-training offers the potential to adapt to a given target dataset and improve performances without introducing additional manual labeling effort. In this section, the questions of whether this adaptation is feasible and whether self-training is able to improve performances upon a model solely trained on synthetic data are answered.

Table 7: Comparison of recognition performances resulting from self-training on all unlabeled samples using different pseudo-label generation strategies. The baseline corresponds to the performances of the initial models. All error rates in [%].

Method	GW			IAM			CVL			BT_{QA}		
	CER	WER	±	CER	WER	±	CER	WER	±	CER	WER	±
Baseline	17.77	45.02	2.9	21.09	48.39	0.7	28.19	58.13	0.3	23.53	52.29	0.4
Argmax	**14.15**	35.14	2.7	**13.01**	35.88	0.7	**10.52**	**31.08**	0.3	14.67	38.49	0.4
Lexicon	14.41	**33.16**	2.7	13.96	**29.18**	0.6	16.24	33.93	0.3	**12.43**	**28.03**	0.4

Taking the previous experiments on synthetic data generation into consideration, all initial models are trained on a synthetic dataset with three million word images that are rendered using all available fonts and the Gutenberg corpus. Better performing initial models might be achievable by adapting the visual or language properties of the synthetic data, but this would introduce further assumptions on the target dataset. As the main application scenario of the proposed method is the new exploration of a novel dataset without introducing any manual effort, previous knowledge is to be avoided. The respective synthetic dataset relies only on the assumption that the text is handwritten and in the English language.

In the following experiments, self-training is performed according to Algorithm 1. Different strategies for pseudo-label generation are investigated. Model outputs are either directly considered as pseudo-labels, as in the case of sigmoid activation for the AttributeCNN, or they are combined with a sharpening function such as binarization or argmax. Furthermore, additional information is introduced to the training process by using a general lexicon during pseudo-label generation. In the experimental setup and also in a practical application, this does not require any further manual effort or additional knowledge of the target domain, as the utilized lexicon can be directly derived from the synthesis procedure. Here, the lexicon includes the ten thousand most frequent words of the Gutenberg corpus that has already been used for generating the synthetic dataset on which the initial models were trained. Note, that the lexicon is only used during the self-training process. Recognition and retrieval operate in a lexicon-free manner.

For the retrieval model, we perform self-training for $K = 20$ cycles and observe query-by-example and query-by-string performances. In each cycle, all images from the target dataset $\mathbf{X}$ are included in the pseudo-labeled training set $\mathbf{X}_{\text{train}}$. Table 6 presents the performances on all four benchmarks. Permutation tests were performed with respect to the baseline which is the performance of the initial models. Directly training on sigmoid activations as pseudo-labels does not lead to any beneficial adaptation in most cases. Performances are significantly worse, except for query-by-example on George Washington and the Bentham 2014 benchmarks. These datasets show the highest initial performances, indicating that an adaptation is only feasible with a certain overall pseudo-label quality. The fact that query-by-string performances degrade particularly strongly indicates that training on erroneous pseudo-labels heavily affects the prediction of character information encoded as attributes.

In contrast to the expectation, exploiting the a priori knowledge on the PHOC vectors by binarizing the network outputs has a harmful effect on performances. This form of sharpening seems to enforce the influence of faulty predictions, making them even worse training targets. In the case of the retrieval model, one can conclude that the general quality of the initial model is not sufficient to effectively learn from the predicted pseudo-labels. Introducing the lexicon at the stage of pseudo-label generation overcomes this problem by introducing an additional form of regularization. Training on pseudo-labels generated by word recognition based on the general lexicon leads to large performance gains with respect to the initial model. The achieved performances result from only training on synthetic and unlabeled data in combination with an easily accessible generic lexicon.

Self-training of the recognition model is performed with the same strategy. An increased number of cycles with $K = 50$ is performed as preliminary experiments showed that the sequence-to-sequence model needs a longer training period to converge. The results in Table 7 show that adapting the model is feasible as performances generally improve upon the baseline. In contrast to the retrieval model, the integration of a lexicon into the training process is not essential for successful adaptation. Already directly considering the predictions as pseudo-label leads to strong performance gains on all benchmarks. Mapping the network outputs to a character prediction can be seen as a form of sharpening which is a common technique in self-training. Incorporating the lexicon in the training process mostly leads to further performance gains, especially with respect to word error rates. The lexicon apparently helps to regularize the sequence predictions during pseudo-label generation, allowing the model to better learn the sequential information. For the CVL database, the results offer an interesting insight into the influence of the choice of lexicon. Due to the highly limited number of unique words in the dataset and also due to the inclusion of German words, the lexicon extracted from the Gutenberg corpus is quite different from the actual dataset lexicon. Using such a dissimilar lexicon which also lacks parts of the dataset words still leads to an improvement upon the baseline without self-training, while using no lexicon at all is clearly superior.

In conclusion, the conducted experiments show that self-training allows models initially trained on synthetic data to adapt to a target dataset, leading to significant performance gains. A clear difference between the two models can be observed. While the recognition model can be directly adapted via self-training, the retrieval model requires further regularization by a lexicon. This characteristic hints at the stronger capacity of the sequence-to-sequence model to implicitly model language information resulting from training on the synthetic dataset. The recognition model still benefits from directly incorporating the lexicon, but the AttributeCNN adaptation without a lexicon is hardly possible.

5.4.5 *Confidence estimation*

At its core, self-training relies on training on pseudo-labels. It is assumed that a higher pseudo-label quality in terms of prediction accuracy leads to better adaptation and higher performances. Many works try to enhance the quality of the pseudo-labeled dataset by removing samples with inaccurate predictions. For example, in the case of a traditional classification network, samples with lower neuron activation at

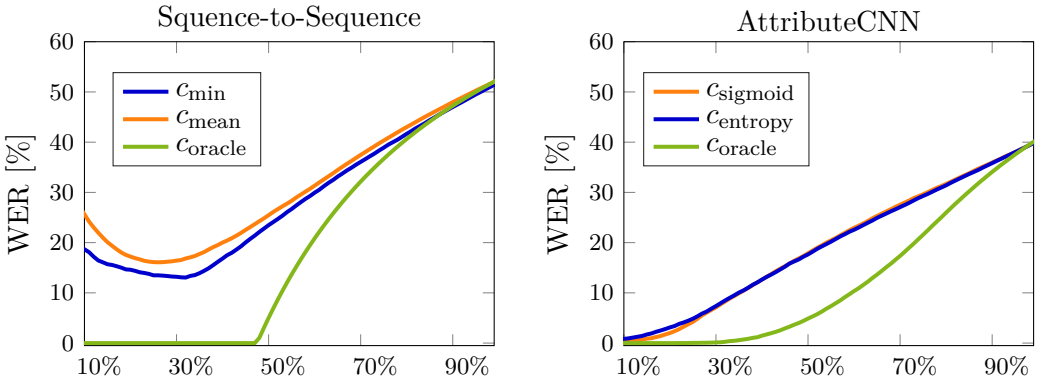


Figure 28: Word error rates over different parts of the IAM test set. The dataset is sorted by its confidence estimates.

the output layer are removed. This technique can be considered a form of confidence estimation with the goal of quantifying the confidence of the network in a given prediction. As discussed in Section 4.5, this work aims at generalizing this principle to the considered AttributeCNN and sequence-to-sequence recognition model.

The first question that needs to be answered is whether the proposed confidence estimates provide a quantitative estimation of the prediction quality in absence of the respective annotation. In general, this boils down to the assumption formulated in Equation 64. The following experiment gives a first indication whether a higher confidence value provided by the proposed methods coincides with higher prediction quality. Consider the models that serve as initial models which have been solely trained on synthetic data. Here, specifically the synthetic dataset using all fonts and the Gutenberg corpus is used as the training set. The corresponding performances on the test set can be found in Table 2 and Table 3. If the proposed confidence measures fulfill the underlying assumption and quantify prediction quality, better performance should be observed for more confident samples. Therefore, we sort the dataset samples according to the predicted confidence values and observe the word error rates of the pseudo-labels for different parts of the dataset. It is noteworthy that for the retrieval model we also report word error rates even though the model is not inherently designed for recognition. Nonetheless, lexicon-based recognition can be performed following the same method as for lexicon-based pseudo-label generation. Figure 28 shows the error rates with respect to the most confident x% of the IAM database. For the other benchmarks, the same plots can be found in the appendix showing the same characteristics. As an upper baseline, the oracle confidence measures are plotted. The oracle confidences exhibit the perfect confidence estimation, giving an error of zero as long as correct pseudo-labels are available and after that a steady decrease in performance is observed. The main observation is that even though the proposed confidence measures do not have access to the actual annotations, they are able to mimic the behavior of the oracle confidence measures. For both models and all confidence measures, lower error rates are observed for those parts of the dataset with more confident pseudo-labels.

Considering the recognition model, both estimates show slightly worse performances for the most confident part of the dataset. This demonstrates that some

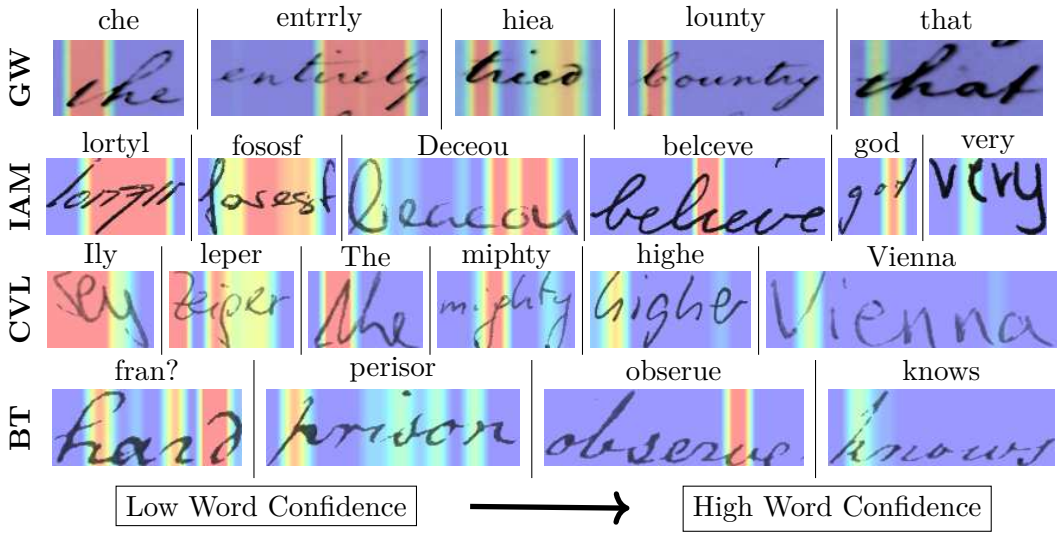


Figure 29: Visualization of local confidence values derived from combining the character confidences c_t and the respective attention weights $\boldsymbol{a}_t$. Red indicates an area of low confidence, blue an area of high confidence. Word images are sorted by the total confidence value c_{mean} with images of lower confidence on the left.

erroneous pseudo-labels have high coinciding confidences. Comparing the two approaches of confidence estimation, taking the mean activation over the prediction sequence seems to be marginally better as its course is slightly closer to the oracle confidence. Using the minimum also means focusing the confidence estimation on a single character prediction seemingly resulting in higher sensitivity.

The results for the retrieval model present a similar characteristic. Both confidence measures coincide with higher prediction performance. Interestingly, including the absence of an attribute in the confidence estimation by computing the entropy makes almost no difference. Whether a sample is deemed to be confident or not is, therefore, mainly determined by the activation of the present attributes. In contrast to the recognition model, the AttributeCNN seems to suffer less from overconfidence as both confidence measures are generally better able to mimic the oracle. Especially in the most confident parts, error rates close to zero can be observed.

In order to further investigate the behavior of the proposed confidence measure, the following analysis of the recognition model is conducted. The attention mechanism integrated in the model directly links a character prediction to a region of the input sequence. This allows a qualitative impression of which image regions lead to unconfident predictions to be obtained. The expectation of a well-working confidence measure is that regions with lower confidence are visually ambiguous and also correspond to prediction errors.

Again, we consider the initial model trained on a synthetic dataset. For a given sample, the prediction relies on the feature sequence $(\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_N)$ that is weighted for a given character at step t by a set of attention weights a_{ti}. Due to the sequentialization procedure, each feature vector corresponds to a series of image columns. Therefore, the attention weights can be interpreted as the numerical relevance of an

image column for a given character prediction. In order to link the local relevance to the proposed confidence estimation, we simply multiply the attention weights by the respective character confidence. By considering each character of the prediction sequence, the entire confidence mass is distributed over the feature vector sequence. This results in a function C that gives a local confidence value depending on the feature vector index i:

$$C(i) = \sum_{t=0}^{N} c_t \cdot a_{it} \tag{74}$$

For visualization purposes, the feature vector index is mapped to the image columns. Figure 29 shows some examples with the resulting confidence mask overlayed on the original image. High confidence values are depicted in blue, while red indicates uncertainty. From a qualitative perspective, the character confidence values align quite well to the perception of the samples. Several characteristics can be observed that show the desired behavior with respect to confidence estimation. For word samples with an overall very low confidence value (i.e. “che”, “lortyl”, “Ily”, “fossof”, “leper”), large parts of the image are linked to low confidence values and severe prediction errors can be observed. Nonetheless, the prediction of individual characters may still be correct and is also reflected by small parts of the image considered as confident. For other samples (i.e. “lounty”, “belceve”, “The”, “mipthy”, “obserue”), the confidence value perfectly quantifies the wrong prediction of individual characters and localizes the error in the image. In general, prediction errors often correspond to ambiguous image regions for which an identification of the actual character without context is mostly unfeasible also from a human point of view. It is important to point out that lower confidence does not inherently result in a prediction error. Consider the following samples i.e. “god”, “that”, “very”, “Vienna”, “knows” with a high general word confidence and correct prediction. Despite the absence of prediction errors, low confidence areas exist and correspond to hard to classify image regions that are qualitatively prone to prediction errors. Overall, the analysis shows that for the recognition model the proposed character confidence numerically captures what would be considered plausible confidence from a human perspective. Additional to an overall evaluation of a sample, the connection of the attention weights allows for localization of potential prediction errors.

After the previous experiment clearly showed that the proposed measures constitute a form of confidence estimation, a natural application is using them to identify correct pseudo-labels. Enhancing the quality of the pseudo-labeled dataset and removing erroneous prediction is assumed to improve self-training performances. The proposed confidence measures allow the unlabeled dataset to be sorted by the respective confidence values. In order to integrate the confidence estimation in the self-training process, we follow Algorithm 2. Even with the availability of a confidence measure $c(\cdot, \theta)$, it remains an open question how many samples shall be part of the training set $\mathbf{X}_{\text{train}}$ at each cycle k. In a first series of experiments, the number of samples was chosen as a portion of the available dataset and a schedule was defined which increased the number of samples over the course of self-training cycles. As a result, a direct comparison can be drawn with respect to the influence of different confidence measures and to training on all pseudo-labeled samples. The total number of self-training cycles is equal to the initial experiments relying on all samples.

For the retrieval model, training is consecutively performed on the upper 30% and 60% confident samples for five cycles each. Then, training is continued for another 10 cycles on all samples. The schedule for the recognition model is defined as 60% and 80% for 10 cycles each followed by 30 cycles of training on all samples. Defining the selection cycles inherently poses a problem in regard to the application scenario. Effectively, the introduction of such a schedule means the introduction of hyperparameters. As the proposed method is designed for scenarios where no annotations are available, tuning said parameter would be unfeasible. The optimization of hyperparameters such as finding an optimal schedule requires a validation set, which might already impose a demanding manual annotation effort. The proposed schedules are mainly motivated by the experienced optimization behavior of the models and their performances on the benchmarks in other experiments. Nonetheless, no empirical optimization of training percentages or cycle counts has been performed, as this would be out of scope for most applications. Therefore, the considered schedules are likely to be suboptimal and primarily serve the purpose of comparing the proposed confidence measures.

With respect to the application of self-training to a novel dataset, the performance of the initial model is unknown. The experiments in this section reflect this fact by performing self-training based on two different variants of initial models. An initial model with comparably high performances is trained based on the proposed synthesis approach using all fonts and the Gutenberg corpus. To avoid that the conclusions on confidence estimation rely only on a well performing initial model, the same evaluation is run starting with initial models trained on the IIIT-HW90K dataset. Performances on the IIIT-HW90K dataset are comparably poor for all benchmarks and for both the recognition and retrieval models.

See Table 8 for the resulting performances of the AttributeCNN including confidence estimation. As discussed in Section 5.4.4, training is performed using a generic lexicon for further regularization. Permutation tests are run with respect to the model trained on all pseudo-labels. Considering performance with respect to the well performing initial model (see Table 8 upper half), the inclusion of confidence estimation leads to performance gains in just a few cases. Only performances for query-by-string on IAM and query-by-example on Bentham 2015 are significantly better than training on all pseudo-labeled samples. This is also the case for the oracle confidence which only marginally improves performance on George Washington. The observation is entirely different when the initial model with poor performances is considered. For all benchmarks, except for George Washington, self-training on all pseudo-labeled samples leads to performance drops and no meaningful adaptation can be observed. It is rather obvious that this results from the extremely poor performances of the initial model on most benchmarks. In the case of George Washington, which by far has the highest initial average precision values, the model converges to a reasonable performance that is still below the performance of the previously considered model resulting from the better performing initial model. Including the confidence based selection of pseudo-labels drastically improves the results. For all benchmarks, self-training allows the model to adapt and performances of training on all pseudo-labels are clearly outperformed. Also for George Washington, where training on all pseudo-labels is successful, selecting pseudo-labels based on confidence is significantly better. The resulting performances are in line with the conclusions

Table 8: Comparison of retrieval performances resulting from self-training with a confidence-based selection under a fixed schedule. All result in mAP [%]. Permutation tests were run with respect to initial models (Initial or IIIT-90K). Significantly different results highlighted in italics.

Method	GW		IAM		BT_{14}		BT_{15}	
	QBE	QBS	QBE	QBS	QBE	QBS	QBE	QBS
Initial	*74.69*	*80.44*	*57.78*	*74.50*	*79.77*	–	*70.09*	–
All	**93.78**	92.71	**80.64**	85.91	96.38	–	77.56	–
$c_{sigmoid}$	93.62	**93.82**	79.72	*87.64*	**96.80**	–	*87.64*	
$c_{entropy}$	93.72	93.70	*79.21*	**87.87**	96.63	–	**87.97**	
c_{oracle}	94.42	93.75	80.25	*88.75*	–	–	–	–
IIIT-90K	*50.48*	*61.63*	*13.66*	*34.80*	*25.58*	–	*15.35*	–
All	90.65	90.07	5.54	11.69	5.84	–	12.89	–
$c_{sigmoid}$	*93.97*	*93.28*	**75.32**	*87.46*	*96.17*	–	**73.10**	–
$c_{entropy}$	**94.60**	**93.80**	*76.85*	**88.77**	**96.34**	–	*58.25*	–
c_{oracle}	*95.09*	*94.22*	*78.10*	*88.84*	–	–	–	–

drawn from Figure 28. Comparing the two measures gives only marginal differences and performances are similar to the results for the oracle confidence.

Table 9 presents the resulting performances for the recognition models. A clear difference between the sequence-to-sequence model and the AttributeCNN can be observed with respect to their sensitivity to the initial model performance. While the AttributeCNN was not able to adapt from the model trained on the IIIT-HW90K dataset, the recognizer is able to do so despite the poor initial performances. Even in the case of the Bentham benchmark with an initial word error rate of 90%, self-training converges and leads to a performance improvement with a word error rate of 70%. The observed differences between the retrieval and recognition models are even more striking, keeping in mind that the AttributeCNN exploits a generic lexicon during pseudo-label generation which provides further information. Focusing training on more confident pseudo-labels already outperforms training on all samples when the better performing initial model is considered (Table 9 upper half). On all benchmarks, a confidence-based selection improves performances in terms of character and word error rates. The impact of the confidence measures and the relative improvement is considerably higher for models trained based on the initial model from the IIIT-HW90K dataset. Taking the mean over the character confidences generally performs better than the minimum-based approach. As already observed in Figure 28, focusing on single character confidences seems to introduce a higher sensitivity. Both confidence measures are outperformed by the oracle con-

Table 9: Comparison of recognition performances resulting from self-training with a confidence-based selection under a fixed schedule. All error rates in [%].

Method	GW			IAM			CVL			BT_{QA}		
	CER	WER	±	CER	WER	±	CER	WER	±	CER	WER	±
Initial	17.77	45.02	2.9	21.09	48.39	0.7	28.19	58.13	0.3	23.53	52.29	0.4
All	14.15	35.14	2.7	13.01	35.88	0.7	10.52	31.08	0.3	14.67	38.49	0.4
c_{mean}	**14.04**	**34.19**	2.7	**11.53**	**32.97**	0.6	**9.97**	**25.62**	0.3	**12.61**	**31.82**	0.3
c_{min}	14.48	34.28	2.7	13.09	35.61	0.7	10.57	24.15	0.3	13.56	34.32	0.4
c_{oracle}	10.40	18.56	2.2	7.45	22.99	0.6	8.63	8.97	0.2	5.52	14.89	0.3
IIIT-90K	44.45	81.19	2.2	32.80	70.25	0.6	35.51	74.53	0.3	54.34	94.00	0.2
All	26.17	60.48	2.8	24.34	64.13	0.7	14.21	46.47	0.3	34.05	69.56	0.3
c_{mean}	**16.69**	**42.87**	2.8	**15.55**	**42.65**	0.7	**9.51**	**31.96**	0.3	**22.83**	**57.67**	0.4
c_{min}	18.35	49.31	2.9	17.36	47.65	0.7	14.69	45.07	0.3	24.56	57.92	0.4
c_{oracle}	15.15	38.49	2.8	10.41	33.11	0.6	9.55	21.38	0.3	12.34	36.05	0.4

fidence, indicating that further improvements are achievable by a better confidence estimation.

In conclusion, the experiments in this section showed that the confidence measures introduced in Section 4.5 allow prediction quality to be quantified. Adding a confidence-based selection to the self-training process generally leads to higher performance and an especially more robust training procedure. Especially when performances are initially poor, which is not quantifiable in the application scenario, the observed impact of the confidence is beneficial. In an extreme case, focusing on confident pseudo-labels may enable an initial model to adapt to a target dataset whereas performance otherwise degrades.

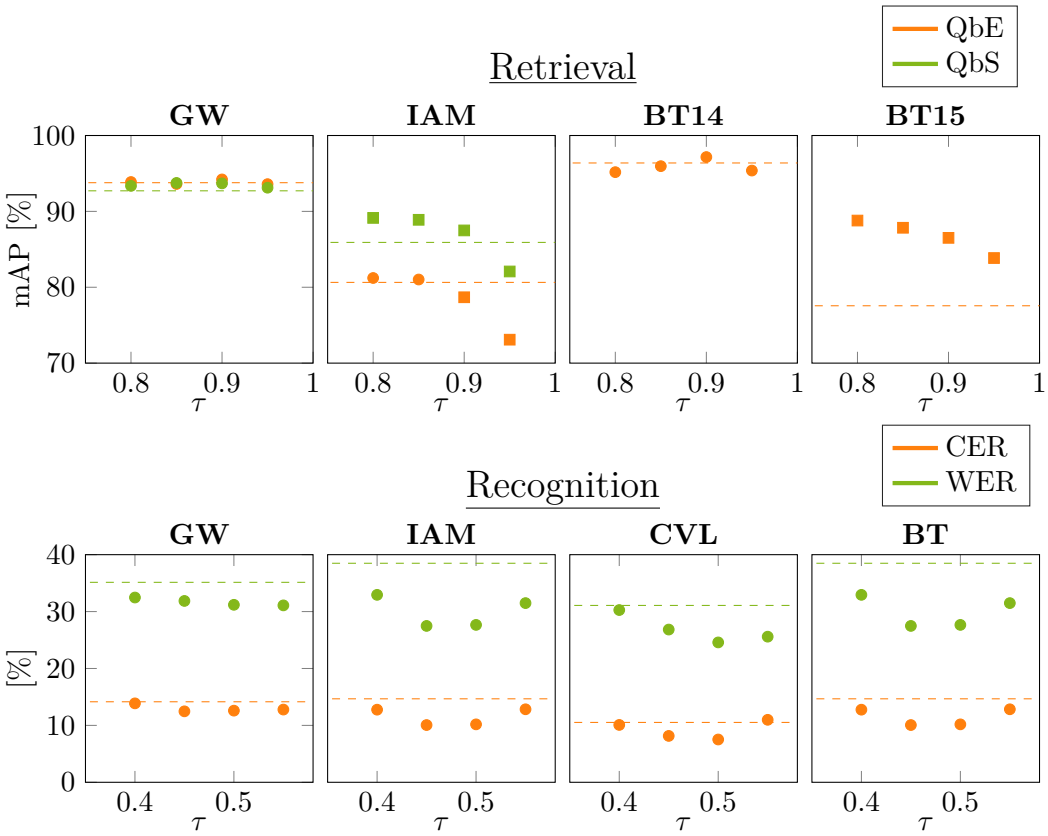


Figure 30: Overview of retrieval and recognition performances under a confidence-based selection (c_{sigmoid} and c_{mean}) with different thresholds. Self-training with no selection is drawn as a dashed line. For retrieval, permutation test are run against self-training on all samples. Significantly different results are marked with squares.

5.4.6 *Thresholding*

The experiments in Section 5.4.5 show that the integration of confidence estimation in the self-training scheme leads to performance gains and a higher robustness in regard to a poor performing initial model. Introducing a predefined schedule-based on fixed portions of the datasets allows different confidence measures to be compared. For the considered benchmarks, it is feasible to derive consistent schedules for both models giving robust results. In how far this approach generalizes in a practical application is questionable, as intuitively the number of selected samples shall not depend on the number of available samples but only on their prediction quality. Therefore, whether a pseudo-labeled sample is used as a training sample should be solely determined by the model and the respective confidence measure.

A possible alternative to choosing fixed numbers of samples is to follow a thresholding approach. As summarized in Algorithm 2, all pseudo-labeled samples with a confidence estimate below a given threshold τ are not considered as training samples. In contrast to the definition of a selection schedule, the number of hyperparameters is reduced to a single threshold value. From the perspective of a practical application, optimizing the threshold value in the absence of a validation set is still problematic,

but has fewer degrees of freedom than using a predefined schedule. For the given benchmarks, an empirical analysis of thresholding with different values is feasible.

In the following experiments, the retrieval and recognition models are trained using self-training with a confidence-based selection strategy and thresholding. For both models, the well performing initial model trained on the synthetic dataset using all fonts and the Gutenberg corpus is used. The retrieval model uses c_{sigmoid} as a confidence measure, while the recognition model relies on c_{mean}. Following the previous training setups, self-training is performed for 20 cycles for the retrieval model and for 50 cycles for the recognition model. The threshold value is varied and performances are compared to training on all available samples. Note that a threshold value above 0.6 for the recognition model results in the selection of almost no samples, due to the label smoothing strategy that is applied during training. Numerically, network activations rarely surpass this value, as the network is not confronted with higher training targets values. Figure 30 summarizes the experimental results of various threshold values. Training on all samples is considered a baseline, indicated by dashed lines. Significantly different performances of the retrieval model according to the permutation tests are represented as squared markers.

The results of the recognition model confirm the general conclusions drawn on the integration of confidence estimation in self-training. On all benchmarks, lower character and word error rates compared to training on all samples can be observed independent of the considered thresholds. Although the highest performances consistently result from using a threshold value of 0.45 or 0.5, it is an important observation that a suboptimal choice of the threshold value does not hurt performance with respect to not including thresholding. The experiments indicate that the addition of a confidence-based selection for self-training of the recognition model is generally beneficial and the choice of a threshold value generalizes to different datasets and initial performances.

Similar conclusions can be drawn for the retrieval model. Performances are either on par or better than training on all samples in most cases. For high threshold values, performances slightly drop, showing that the model is probably more robust towards a drop of pseudo-label quality rather than to a lack of variance in the training set. For George Washington and Bentham 2014, only marginal performance differences can be observed, which is consistent with the results for a predefined selection schedule. Training based on an initial model with poor performance is likely to lead to an increase in performance differences when a confidence estimate is integrated.

Overall, the resulting models either perform on par or better than when the predefined schedules discussed in Section 5.4.5 are used. Thresholding results in an automatic selection of the number of samples that is superior to an unoptimized heuristic schedule. A closer look at the training behavior of the models reveals the characteristics depicted in Figure 31, which shows the relative number of samples that lie above the threshold at each self-training cycle and that are, therefore, added to the training set $\mathbf{X}_{\text{train}}$. Here, threshold values of 0.8 and 0.5 are considered for retrieval and recognition, respectively. No clear relation between the initial performances and the initial number of samples above the threshold can be observed. Over the course of self-training, both models become more confident in their pseudo-label predictions and increasing numbers of samples are added to the pseudo-labeled training set. Thresholding based on a fixed confidence value is able to replace a pre-

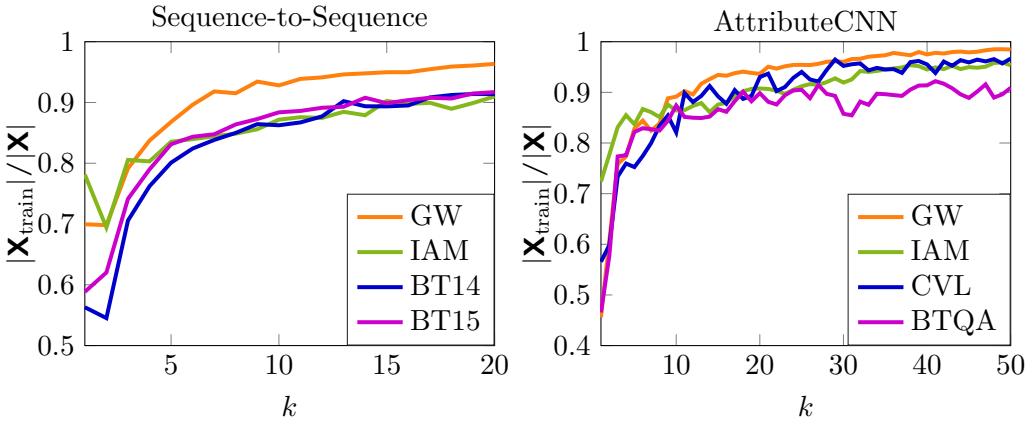


Figure 31: Evolution of the relative number of samples above the threshold τ. The sequence-to-sequence model performs self-training with c_{mean} and $\tau = 0.5$. The AttributeCNN is trained with c_{sigmoid} and $\tau = 0.8$.

defined schedule. The models are capable of successfully balancing the quality of pseudo-labels and the growing variance provided by more samples.

5.4.7 *Annotation-free application*

The main application area of the proposed method is the exploration of a novel document collection. In this case, the requirement for representative training material poses a substantial burden for the application of a deep neural network. Nonetheless, word retrieval or recognition is highly desirable to retrieve information from a respective document collection. This problem is also reflected by the considered benchmarks and different works in the literature that aim at word recognition and retrieval without relying on manual annotations.

In the literature on word retrieval, most works fall in one of two categories. On one hand, a paradigm shift towards deep neural networks has taken place and several works reported tremendous performance gains in the presence of a labeled training set [SF18; Rus+18; WB16; KDJ18; KDJ23]. On another hand, several works avoid the use of any labeled data and, therefore, propose methods that are learning-free and do not rely on a trained model [Alm+12; KWD14; ZPG17; PTV15a; SRG16; Alm+14; Ret+19]. These methods do not impose any label effort but in terms of performance are clearly inferior especially when more visually challenging benchmarks are considered. Works that train models on data that is not directly representative are rare and often solely rely on training on synthetic data which results in non-competitive performances [GSF18; KDJ23].

Table 10 presents retrieval performances for George Washington and the IAM database for annotation-free and annotation-based methods from the literature. With respect to methods that do not include manual labels during training, the presented self-training approach clearly outperforms all learning-free approaches. Performances of the model only trained on synthetic data are similar to other results in the literature [KDJ23] and are, in many cases, already superior to the reported per-

Table 10: Comparison of retrieval performances on GW and IAM to results from the literature. All results are reported in mAP [%]. Methods marked as annotation-free do not rely on manually labeled samples.

Method	Annotation Free	GW		IAM	
		QBE	QBS	QBE	QBS
Synthetic Training	✓	74.7	80.4	57.8	74.5
Self-Training	✓	93.8	92.7	80.6	85.9
Self-Training (Confidence)	✓	**93.9**	**93.4**	**81.2**	**89.1**
Almazan et al. [Alm+12]	✓	49.4	–	–	–
Sfikas et al. [SRG16]	✓	58.3	–	13.2	–
DTW [Alm+14]	✓	60.6	–	12.3	–
FV [Alm+14]	✓	62.7	–	15.6	–
Retsinas et al. [Ret+19]	✓	77.1	–	28.1	–
Gurjar et al. [GSF18]	✓	39.8	48.9	26.2	36.5
HWNet v3 (Synthetic) [KDJ23]	✓	–	–	54.1	69.2
AttributeSVM [Alm+14]		93.0	91.2	55.7	73.7
TPP-PHOCNet [SF18]		97.9	96.7	84.8	92.9
STPP-PHOCNet [Rus+18]		97.7	96.8	89.2	95.4
Triplet-CNN [WB16]		98.0	93.6	81.5	89.4
Deep Embed [KDJ18]		98.0	98.8	90.4	94.0
HWNet v3 [KDJ23]		99.5	99.8	93.2	97.5
Retsinas et al. [Ret+21]		97.9	98.4	92.0	95.9

formances of learning-free approaches. Additionally, training a model allows query-by-string to be performed which is out of scope otherwise. As discussed in the earlier experiments, adapting the proposed model with self-training under a confidence based pseudo-label selection further increases model performances. Of course, other works report higher performances under the supervision of the labeled training set, but the proposed self-training method is able to tremendously reduce the performance gap towards a model trained only on synthetic data. The comparison shows the often proven effectiveness of neural networks for vision tasks, as even with no labeled training data, the method is able to outperform a similar approach based on SVMs under full supervision [Alm+14].

For both benchmarks based on the Bentham document collection, similar observations can be made, see Table 11. Other annotation-free methods are clearly outperformed. Even though it is not possible to report query-by-string performances for the benchmarks due to the lack of annotations, it is noteworthy that the method

Table 11: Comparison of retrieval performances on BT to results from the literature. All results are reported in mAP [%]. Non of the methods rely on manually labeled samples.

Method	Annotation Free	BT_{14} QBE	BT_{15} QBE
Synthetic Trainig	✓	79.8	70.1
Self-Training	✓	**96.4**	77.6
Self-Training (Confidence)	✓	95.2	**88.8**
Aldavert et al. [Ald+15]	✓	46.5	–
Almazan et al. [Alm+14]	✓	51.3	–
Kovalchuk et al. [KWD14]	✓	52.4	–
CVC [PTV15a]	✓	–	30.0
PRG [PTV15a]	✓	–	42.4
Sfikas et al. [SRG16]	✓	53.6	41.5
Zagoris et al. [ZPG17]	✓	60.0	50.1
Retsinas et al. [Ret+19]	✓	71.1	58.4

is capable of performing query-by-string. Again, this is not the case for other works in the literature.

A similar comparison to the state-of-the art can be drawn for the recognition model, see Table 12. Most works on recognition either report performances for models trained in a fully-supervised fashion or for models trained on synthetic data. In this regard, the work of Kang et al. [Kan+20b] provides an interesting comparison in terms of the adaptation efficiency of the proposed self-training approach. The underlying model and, therefore, also fully-supervised and synthetic performances are comparable. In the considered work, the authors adapt the recognition model by introducing an additional loss that is supposed to align the feature vectors of synthetic and real world samples. The reported performance gains are considerably smaller compared to the use of self-training. This clearly indicates the effectiveness of the proposed training scheme in the application area. While the work of Kang et al. [Kan+20b] focuses the adaptation process solely on the visual domain in the feature space, self-training benefits from the language bias and the learned information on the target domain. Overall, training an initial model on synthetic data followed by self-training gives state-of-the-art results for recognition in the absence of a manually labeled training set. In comparison to models trained in a fully-supervised manner, a performance gap still exists which leaves room for further improvements on the adaptation of neural models.

Table 12: Comparison of recognition performances to results from the literature. All error rates are reported in [%]. Methods that do not rely on manually labeled samples are marked as annotation-free (AF).

Method	AF	GW		IAM		CVL		BT_{QA}	
		CER	WER	CER	WER	CER	WER	CER	WER
Synthetic	✓	17.8	45.0	21.1	48.4	28.2	58.1	23.5	52.3
Self-training	✓	14.2	35.1	13.0	35.9	10.5	31.1	14.7	38.5
Self-training (Confidence)	✓	**12.6**	**31.2**	**10.2**	**27.7**	**7.5**	**24.6**	**10.2**	**27.7**
Kang et al. [Kan+20b]	✓	26.1	56.8	26.4	54.6	26.3	55.6	–	–
Kang et al. [Kan+20b]	✓	16.3	39.9	14.1	34.9	19.2	44.3	–	–
Mathew et al. [Mat+21a]	✓	–	–	–	–	–	–	–	76.8
HWNet v3 [KDJ23]	✓	–	–	26.4	37.9	–	–	–	–
Kang et al. [Kan+20b]		4.6	13.5	6.9	17.5	3.6	7.8	–	–
Aberdam et al. [Abe+21]		–	–	9.5	20.1	–	23.0	–	–
Retsinas et al. [Ret+21]		4.2	10.8	5.0	14.7	–	–	–	–
HWNet v3* [KDJ23]		–	–	1.7	3.6	–	–	–	–

6 CONCLUSIONS

6.1 SUMMARY

In this thesis, a method for self-training has been developed and evaluated. The main contribution of this work lies in the application of the self-training scheme to two classical document analysis problems in combination with using synthetic data generation. As the entire training process relies solely on synthetic and unlabeled data, this work shows that powerful models can be learned in the complete absence of manually annotated training material.

In order to show the effectiveness of self-training in the context of document analysis, the two tasks of word recognition and word retrieval were considered. Both tasks are traditionally addressed with models relying on neural networks. For both tasks, state-of-the-art models from the literature and their working principles were introduced. Choosing the AttributeCNN and the sequence-to-sequence recognition model was motivated by these models being established approaches in the community. Nonetheless, it can be argued that the conclusions drawn on the training principles are not limited to the specific choice of models. Similar conclusions can be drawn for both tasks and models, despite their significant structural differences.

The proposed self-training approach requires an initial model to generate pseudo-labels. Since no manually-annotated data is available, synthetic data is generated using a generation pipeline relying on true type fonts. In the experimental evaluation, it is shown that the underlying vocabulary heavily influences the performances of the models trained only on synthetic data. Further, experiments motivate the conclusion that training on a synthetic dataset constitutes a form of implicit language modeling. A severe degeneration of performance can be observed for both models when language information is removed. Interestingly, this is already the case when the naturally occurring word distribution is removed from the synthetic dataset.

Additional to the encoded language information in the synthetic training set, the visual appearance of the generated word images influences performances. As the synthesis procedure is based on true type fonts which provide visual variability in writing styles, an obvious question is how to select suitable fonts. A method based on font classification is proposed that allows for the identification of fonts which are visually more similar to a target dataset. The main conclusion of the conducted experiments is, nonetheless, that the models generally benefit from a high variability in font styles and, therefore, using a high number of fonts for data synthesis is preferable.

After the investigation of data synthesis for training initial models, the proposed method uses self-training to adapt the considered models to the target datasets. It is shown that both models show increased performances after training on pseudo-labels. A significant difference between the models can be observed, as the AttributeCNN for word retrieval requires an approximate lexicon with respect to benchmarks with poorer initial performances. This is not the case for the sequential recognition model which shows improved performances for all benchmarks and does not benefit from

additionally including a lexicon for pseudo-label generation. The conducted experiment therefore indicates that the sequential model learns a more powerful implicit language model that supports pseudo-label generation and the model adaptation via self-training.

A further contribution of this work is the introduction of confidence estimates for each model and their integration into the self-training approach. It is shown that confidence estimation is possible based on the final neural activations of the networks. The introduced numerical measures quantify prediction quality in the absence of an annotation. Two different ways to integrate the confidence measures into self-training are presented, which rely either on a fixed schedule or thresholding. For both approaches, focusing the self-training process on pseudo-labels with high confidences improves final performances, especially when initial model performances are poor.

Finally, the proposed approach is compared to other works in the literature that consider word recognition and retrieval without requiring manually annotated training material. The proposed method outperforms the state-of-the-art. This is mainly to be explained by the fact that the presented self-training approach allows for training established models on synthetic and unlabeled data. Therefore, the power of large neural networks and machine learning can be leveraged without introducing the severe drawback of requiring manually-labeled data.

6.2 OUTLOOK

The results of this work motivate further research on self-training techniques for different tasks and model types. In this regard, document analysis is an obvious application area for several reasons. Automatic document analysis can have significant impact on the digitization and analysis of historic document collections. This area naturally suffers from a lack of annotated material. Nonetheless, most documents are regularized by their inherent structure and their language is often known. Self-training can be expected to give consistent performance gains, if an initial model and a sufficient amount of unlabeled data is available, due to its generality.

A natural evolution of the method presented in this work is its adaptation to recognition systems that do not rely on word segmentation. While word recognition is the problem at hand in many application domains, line-based or full-page recognition systems integrate the segmentation step into the recognition model. In this case, self-training can supposedly follow the principles established in this thesis. It can be expected that the extended context available to a line or full-page system helps the generation and evaluation of pseudo-labels.

The proposed self-training approach can be further extended by exploring the concept of consistency regularization. Many works that use self-training techniques for different computer vision tasks rely heavily on augmentation strategies to generate distorted versions of the same training sample [Ber+19; Ber+20; Soh+20]. Here, the intuition is that the training process is further regularized by the fact that the prediction needs to be consistent over all augmented instances of an image. Independent from the actual pseudo-label, this provides an additional form of supervision. A first step towards integrating consistency-based regularization was presented in [WF22b]

where augmented word images were used to regularize the training of a handwriting recognition model.

Despite the efficiency of the presented approach, there exists a natural performance gap with respect to a model which is trained fully-supervised. If an application requires higher performances than those achievable in an annotation-free manner, manual annotation effort remains necessary. It is an open question how to combine synthetic data generation, self-training and small numbers of manually-annotated samples. In an initial work, [WF22a] explores different strategies on how to select samples for annotation and how to integrate those in the self-training procedure.

Future research in handwritten word recognition and retrieval should focus on the application and training of models with minimal to no manual annotation effort. An interesting development to be observed is the recent introduction of multimodal large language models. Even though this family of models originally stems from the area of natural language processing [PG23], models like BERT [Dev+19] or the GPT [Ope23] family have shown impressive capabilities. The extension of the traditional language modeling capabilities to include multimodal inputs such as images opens up a future paradigm shift for document analysis. Essentially, a multimodal large language model is potentially capable of providing an answer to a question with respect to an input image. The question, or more specifically the prompt, could simply ask for the corresponding transcription. Given the previously unseen scale of recent large language models and their already impressive capabilities, it is likely that this approach will be feasible in the future. One main concern is how to adapt such a model to a designated task. Therefore, principles following the general concept of self-training are again of interest.

BIBLIOGRAPHY

[Abe+21] A. Aberdam, R. Litman, S. Tsiper, O. Anschel, R. Slossberg, S. Mazor, R. Manmatha, and P. Perona. "Sequence-to-Sequence Contrastive Learning for Text Recognition." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Nashville, TN, USA, 2021, pp. 15302–15312.

[Abi+12] A. Abidi, A. Jamil, I. Siddiqi, and K. Khurshid. "Word Spotting Based Retrieval of Urdu Handwritten Documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Bari, Italy, 2012, pp. 331–336.

[Ald+15] D. Aldavert, M. Rusiñol, R. Toledo, and J. Lladós. "A study of bag-of-visual-words representations for handwritten keyword spotting." In: *Int. Journal on Document Analysis and Recognition* 18.3 (2015), pp. 223–234.

[Alm+12] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. "Efficient exemplar word spotting." In: *Proc. of the British Machine Vision Conf.* Surrey, UK, 2012.

[Alm+14] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. "Word spotting and recognition with embedded attributes." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 36.12 (2014), pp. 2552–2566.

[Ame+19] M. R. Ameri, M. Stauffer, K. Riesen, T. D. Bui, and A. Fischer. "Graph-based keyword spotting in historical manuscripts using Hausdorff edit distance." In: *Pattern Recognition Letters* 121 (2019), pp. 61–67.

[AMM19] E. Alonso, B. Moysset, and R. O. Messina. "Adversarial Generation of Handwritten Text Images Conditioned on Sequences." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 481–486.

[Bah+16] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio. "End-to-end attention-based large vocabulary speech recognition." In: *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*. Shanghai, China: IEEE, 2016, pp. 4945–4949.

[Bar+22] K. Barrere, Y. Soullard, A. Lemaitre, and B. Coüasnon. "A Light Transformer-Based Architecture for Handwritten Text Recognition." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vol. 13237. La Rochelle, France, 2022, pp. 275–290.

[BCB15] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." In: *Proc. of the Int. Conf. on Learning Representations*. San Diego, CA, USA, 2015.

[BCD01] L. D. Brown, T. T. Cai, and A. DasGupta. "Interval estimation for a binomial proportion." In: *Statistical science* 16.2 (2001), pp. 101–133.

[Ben09] Y. Bengio. "Learning Deep Architectures for AI." In: *Found. Trends Machine Learning* 2.1 (2009), pp. 1–127.

[Ber+19] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel. "MixMatch: A Holistic Approach to Semi-Supervised Learning." In: *Proc. of the Neural Information Processing Systems Conf.* Vancouver, BC, Canada, 2019, pp. 5050–5060.

[Ber+20] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel. "ReMixMatch: Semi-Supervised Learning with Distribution Matching and Augmentation Anchoring." In: *Proc. of the Int. Conf. on Learning Representations*. Addis Ababa, Ethiopia, 2020.

[BHM91] J. S. Bridle, A. J. R. Heading, and D. J. C. MacKay. "Unsupervised Classifiers, Mutual Information and 'Phantom Targets'." In: *Proc. of the Neural Information Processing Systems Conf.* Denver, CO, USA, 1991, pp. 1096–1101.

[Bid+15] G. Bideault, L. Mioulet, C. Chatelain, and T. Paquet. "Benchmarking discriminative approaches for word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nancy, France, 2015, pp. 201–205.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. ISBN: 0387310738.

[BJG08] A. Bhardwaj, D. Jose, and V. Govindaraju. "Script Independent Word Spotting in Multilingual Documents." In: *Int. Joint Conf. on Natural Language Processing*. Hyderabad, India, 2008, pp. 48–54.

[BLM17] T. Bluche, J. Louradour, and R. O. Messina. "Scan, Attend and Read: End-to-End Handwritten Paragraph Recognition with MDLSTM Attention." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 1050–1055.

[BLT09] S. Bai, L. Li, and C. L. Tan. "Keyword Spotting in Document Images through Word Shape Coding." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Barcelona, Spain, 2009, pp. 331–335.

[BM94] H. A. Bourlard and N. Morgan. *Connectionist speech recognition: a hybrid approach*. Vol. 247. 1994.

[BNK] T. Bluche, H. Ney, and C. Kermorvant. "Tandem HMM with convolutional neural network for handwritten word recognition." In: *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada: IEEE, pp. 2390–2394.

[Boq+11] S. E. Boquera, M. J. C. Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. "Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33.4 (2011), pp. 767–779.

[BP66] L. E. Baum and T. Petrie. "Statistical inference for probabilistic functions of finite state Markov chains." In: *The annals of mathematical statistics* 37.6 (1966), pp. 1554–1563.

[BR11] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval: The Concepts and Technology behind Search*. Vol. 82. 2011.

[Bru+14] S. Brunessaux, P. Giroux, B. Grilhères, M. Manta, M. Bodin, K. Choukri, O. Galibert, and J. Kahn. "The Maurdor Project: Improving Automatic Processing of Digital Documents." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Tours, France, 2014, pp. 349–354.

[BSF94] Y. Bengio, P. Y. Simard, and P. Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[Car+19] M. Carbonell, J. Mas, M. Villegas, A. Fornés, and J. Lladós. "End-to-End Handwritten Text Detection and Transcription in Full Pages." In: *Int. Workshop on Machine Learning (ICDAR)*. Sydney, Australia, 2019, pp. 29–34.

[Car+20] M. Carbonell, A. Fornés, M. Villegas, and J. Lladós. "A neural model for text localization, transcription and named entity recognition in full pages." In: *Pattern Recognition Letters* 136 (2020), pp. 219–227.

[Cau+18] T. Causer, K. Grint, A.-M. Sichani, and M. Terras. "Making such bargain: Transcribe Bentham and the quality and cost-effectiveness of crowdsourced transcription." In: *Digital Scholarship in the Humanities* 33.3 (2018), pp. 467–487.

[Che+17a] Z. Chen, Y. Wu, F. Yin, and C. Liu. "Simultaneous Script Identification and Handwriting Recognition via Multi-Task Learning of Recurrent Neural Networks." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 525–530.

[Che+17b] Z. Cheng, F. Bai, Y. Xu, G. Zheng, S. Pu, and S. Zhou. "Focusing Attention: Towards Accurate Text Recognition in Natural Images." In: *Proc. of the Int. Conf. on Computer Vision*. Venice, Italy: IEEE Computer Society, 2017, pp. 5086–5094.

[Cho+14] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *Conf. on Empirical Methods in Natural Language Processing*. Doha, Qatar, 2014, pp. 1724–1734.

[Cho+15] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. "Attention-Based Models for Speech Recognition." In: *Proc. of the Neural Information Processing Systems Conf.* Montreal, Quebec, Canada, 2015, pp. 577–585.

[Chu+14] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." In: *CoRR* abs/1412.3555 (2014).

[CNL11] A. Coates, A. Y. Ng, and H. Lee. "An Analysis of Single-Layer Networks in Unsupervised Feature Learning." In: *Proc. of the Int. Conf. on Artificial Intelligence and Statistics*. Vol. 15. Fort Lauderdale, FL, USA, 2011, pp. 215–223.

[Csu+04] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. "Visual categorization with bags of keypoints." In: *Workshop on statistical learning in computer vision, ECCV*. Vol. 1. 1-22. Prague, Czech Republic, 2004, pp. 1–2.

[CSZ06] O. Chapelle, B. Schölkopf, and A. Zien, eds. *Semi-Supervised Learning*. The MIT Press, 2006.

[CT14] T. Causer and M. Terras. "Many hands make light work, many hands together make merry work: Transcribe Bentham and crowdsourcing manuscript collections." In: *M. Ridge (ed.), Crowdsourcing Our Cultural Heritage (Ashgate)*. 2014.

[CUH16] D. Clevert, T. Unterthiner, and S. Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *Proc. of the Int. Conf. on Learning Representations*. San Juan, Puerto Rico, 2016.

[Cur44] H. B. Curry. "The method of steepest descent for non-linear minimization problems." In: *Quarterly of Applied Mathematics* 2.3 (1944), pp. 258–261.

[CV18] A. Chowdhury and L. Vig. "An Efficient End-to-End Neural Model for Handwritten Text Recognition." In: *Proc. of the British Machine Vision Conf.* Newcastle, UK, 2018, p. 202.

[CW12] T. Causer and V. Wallace. "Building A Volunteer Community: Results and Findings from Transcribe Bentham." In: *Digital Humanities Quarterly* 6 (Jan. 2012).

[CZF06] J. Chan, C. Ziftci, and D. A. Forsyth. "Searching Off-line Arabic Documents." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. New York, NY, USA, 2006, pp. 1455–1462.

[Dev+19] J. Devlin, M. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Minneapolis, MN, USA, 2019, pp. 4171–4186.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. 2nd. Wiley-Interscience, 2001.

[DHS11] J. C. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal on Machine Learning Research* 12 (2011), pp. 2121–2159.

[Din+23] H. Ding, B. Luan, D. Gui, K. Chen, and Q. Huo. "Improving Handwritten OCR with Training Samples Generated by Glyph Conditional Denoising Diffusion Probabilistic Model." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 14190. San José, CA, USA, 2023, pp. 20–37.

[DJ20] D. Das and C. V. Jawahar. "Adapting OCR with Limited Supervision." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vol. 12116. 2020, pp. 30–44.

[Dos+21] A. Dosovitskiy *et al.* "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *Proc. of the Int. Conf. on Learning Representations*. Virtual Event, Austria, 2021.

[Dov+13] V. Dovgalecs, A. Burnett, P. Tranouez, S. Nicolas, and L. Heutte. "Spot It! Finding Words and Patterns in Historical Documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA, 2013, pp. 1039–1043.

[DT05] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. San Diego, CA, USA, 2005, pp. 886–893.

[Dut+18] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar. "Improving CNN-RNN Hybrid Networks for Handwriting Recognition." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Niagara Falls, NY, USA, 2018, pp. 80–85.

[DXX18] L. Dong, S. Xu, and B. Xu. "Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition." In: *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*. Calgary, AB, Canada, 2018, pp. 5884–5888.

[EH20] J. E. van Engelen and H. H. Hoos. "A survey on semi-supervised learning." In: *Mach. Learn.* 109.2 (2020), pp. 373–440.

[EMH19] T. Elsken, J. H. Metzen, and F. Hutter. "Neural Architecture Search: A Survey." In: *Journal on Machine Learning Research* 20 (2019), 55:1–55:21.

[Far+09] A. Farhadi, I. Endres, D. Hoiem, and D. A. Forsyth. "Describing objects by their attributes." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Miami, Florida, USA, 2009, pp. 1778–1785.

[Fin14] G. A. Fink. *Markov Models for Pattern Recognition - From Theory to Applications*. Advances in Computer Vision and Pattern Recognition. Springer, 2014.

[Fis+12] A. Fischer, A. Keller, V. Frinken, and H. Bunke. "Lexicon-free handwritten word spotting using character HMMs." In: *Pattern Recognition Letters* 33.7 (2012), pp. 934–942.

[Fis+13] A. Fischer, V. Frinken, H. Bunke, and C. Y. Suen. "Improving HMM-Based Keyword Spotting with Character Language Models." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA, 2013, pp. 506–510.

[FJK23] L. Fei-Fei, J. Johnson, and A. Karpathy. *CS231n: Convolutional Neural Networks for Visual Recognition*. 2023. URL: http://cs231n.stanford.edu/.

[Fog+20] S. Fogel, H. Averbuch-Elor, S. Cohen, S. Mazor, and R. Litman. "ScrabbleGAN: Semi-Supervised Varying Length Handwritten Text Generation." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Seattle, WA, USA, 2020, pp. 4323–4332.

[FPZ16] C. Feichtenhofer, A. Pinz, and A. Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, 2016, pp. 1933–1941.

[Fri+12] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. "A Novel Word Spotting Method Based on Recurrent Neural Networks." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 34.2 (2012), pp. 211–224.

[GB04] Y. Grandvalet and Y. Bengio. "Semi-supervised Learning by Entropy Minimization." In: *Proc. of the Neural Information Processing Systems Conf.* Vancouver, British Columbia, Canada, 2004, pp. 529–536.

[GBB11] X. Glorot, A. Bordes, and Y. Bengio. "Deep Sparse Rectifier Neural Networks." In: *Int. Conf. on Artificial Intelligence and Statistics*. Vol. 15. Fort Lauderdale, FL, USA, 2011, pp. 315–323.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Gim+14] A. Giménez, I. Khoury, J. Andrés-Ferrer, and A. Juan. "Handwriting word recognition using windowed Bernoulli HMMs." In: *Pattern Recognition Letters* 35 (2014), pp. 149–156.

[Gio+17] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou. "A survey of document image word spotting techniques." In: *Pattern Recognition* 68 (2017), pp. 310–332.

[Gra+06] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In: *Proc. of the Int. Conf. on Machine Learning*. Vol. 148. Pittsburgh, PA, USA, 2006, pp. 369–376.

[Gra+09] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. "A Novel Connectionist System for Unconstrained Handwriting Recognition." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 31.5 (2009), pp. 855–868.

[Gre+17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. "LSTM: A Search Space Odyssey." In: *Transactions on Neural Networks Learning Systems* 28.10 (2017), pp. 2222–2232.

[Grü+19] T. Grüning, G. Leifert, T. Strauß, J. Michael, and R. Labahn. "A two-stage method for text line detection in historical documents." In: *Int. Journal on Document Analysis and Recognition* 22.3 (2019), pp. 285–302.

[GS00] F. A. Gers and J. Schmidhuber. "Recurrent Nets that Time and Count." In: *Int. Joint Conf. on Neural Networks*. Como, Italy, 2000, pp. 189–194.

[GS08] A. Graves and J. Schmidhuber. "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks." In: *Proc. of the Neural Information Processing Systems Conf.* Vancouver, British Columbia, Canada, 2008, pp. 545–552.

[GSF18] N. Gurjar, S. Sudholt, and G. A. Fink. “Learning Deep Representations for Word Spotting under Weak Supervision.” In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vienna, Austria, 2018, pp. 7–12.

[He+15a] K. He, X. Zhang, S. Ren, and J. Sun. “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.” In: *Proc. of the Int. Conf. on Computer Vision*. Santiago, Chile, 2015, pp. 1026–1034.

[He+15b] K. He, X. Zhang, S. Ren, and J. Sun. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition.” In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916.

[He+16] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition.” In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, 2016, pp. 770–778.

[HES00] H. Hermansky, D. P. W. Ellis, and S. Sharma. “Tandem connectionist feature extraction for conventional HMM systems.” In: *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*. Istanbul, Turkey, 2000, pp. 1635–1638.

[Hoc+01] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, *et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[HS97] S. Hochreiter and J. Schmidhuber. “Long Short-Term Memory.” In: *Neural Comput.* 9.8 (1997), pp. 1735–1780.

[HSW89] K. Hornik, M. B. Stinchcombe, and H. White. “Multilayer feedforward networks are universal approximators.” In: *Neural Networks* 2.5 (1989), pp. 359–366.

[Hua+17] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. “Densely Connected Convolutional Networks.” In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Honolulu, HI, USA, 2017.

[Jad+14] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. “Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition.” In: *CoRR* abs/1406.2227 (2014).

[Jai+20] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. “A Survey on Contrastive Self-supervised Learning.” In: *CoRR* abs/2011.00362 (2020).

[JH98] T. S. Jaakkola and D. Haussler. “Exploiting Generative Models in Discriminative Classifiers.” In: *Proc. of the Neural Information Processing Systems Conf.* Denver, Colorado, USA, 1998, pp. 487–493.

[JLG78] S. Johansson, G. N. Leech, and H. Goodluck. *Manual of information to accompany the Lancaster-Oslo/Bergen corpus of British English, for use with digital computers*. Tech. rep. Oslo, Norway: Department of Computer Science, University of Oslo, 1978.

[JVZ14] M. Jaderberg, A. Vedaldi, and A. Zisserman. "Deep Features for Text Spotting." In: *Proc. of the European Conf. on Computer Vision*. Vol. 8692. Zurich, Switzerland, 2014, pp. 512–528.

[Kal+93] A. Kaltenmeier, T. Caesar, J. M. Gloger, and E. Mandler. "Sophisticated topology of hidden Markov models for cursive script recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Tsukuba City, Japan: IEEE Computer Society, 1993, pp. 139–142.

[Kan+18] L. Kang, J. I. Toledo, P. Riba, M. Villegas, A. Fornés, and M. Rusiñol. "Convolve, Attend and Spell: An Attention-based Sequence-to-Sequence Model for Handwritten Word Recognition." In: *Proc. of the German Conf. on Pattern Recognition*. Vol. 11269. Stuttgart, Germany, 2018, pp. 459–472.

[Kan+20a] L. Kang, P. Riba, Y. Wang, M. Rusiñol, A. Fornés, and M. Villegas. "GANwriting: Content-Conditioned Generation of Styled Handwritten Word Images." In: *Proc. of the European Conf. on Computer Vision*. Vol. 12368. Glasgow, UK, 2020, pp. 273–289.

[Kan+20b] L. Kang, M. Rusinol, A. Fornés, P. Riba, and M. Villegas. "Unsupervised writer adaptation for synthetic-to-real handwritten word recognition." In: *Winter Conference on Applications of Computer Vision*. Snow Mass Village, CO, USA, 2020, pp. 3502–3511.

[Kan+21] L. Kang, P. Riba, M. Villegas, A. Fornés, and M. Rusiñol. "Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture." In: *Pattern Recognition* 112 (2021), p. 107790.

[Kan+22] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas. "Pay attention to what you read: Non-recurrent handwritten text-Line recognition." In: *Pattern Recognition* 129 (2022), p. 108766.

[Kan20] L. Kang. *Robust Handwritten Text Recognition in Scarce Labeling Scenarios: Disentanglement, Adaptation and Generation*. Dissertation. Barcelona, Spain, 2020.

[KB15] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization." In: *Proc. of the Int. Conf. on Learning Representations*. San Diego, CA, USA, 2015.

[KBH21] M. Kiss, K. Benes, and M. Hradis. "AT-ST: Self-training Adaptation Strategy for OCR in Domains with Limited Transcriptions." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 12824. Lausanne, Switzerland, 2021, pp. 463–477.

[KDJ16] P. Krishnan, K. Dutta, and C. V. Jawahar. "Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Shenzhen, China, 2016, pp. 289–294.

[KDJ18] P. Krishnan, K. Dutta, and C. V. Jawahar. "Word Spotting and Recognition Using Deep Embedding." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vienna, Austria, 2018, pp. 1–6.

[KDJ23] P. Krishnan, K. Dutta, and C. Jawahar. "HWNet v3: a joint embedding framework for recognition and retrieval of handwritten text." In: *Int. Journal on Document Analysis and Recognition* (2023), pp. 1–17.

[KDN13] M. Kozielski, P. Doetsch, and H. Ney. "Improvements in RWTH's System for Off-Line Handwriting Recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 935–939.

[KH+09] A. Krizhevsky, G. Hinton, *et al.* "Learning multiple layers of features from tiny images." In: (2009).

[KH93] S. Khoubyari and J. J. Hull. "Keyword location in noisy document image." In: *Annual Symposium on Document Analysis and Information Retrieval*. Las Vegas, NV, USA, 1993, pp. 217–231.

[KJ16] P. Krishnan and C. V. Jawahar. "Generating Synthetic Data for Text Recognition." In: *CoRR* abs/1608.04224 (2016).

[KJ19] P. Krishnan and C. V. Jawahar. "HWNet v2: an efficient word image representation for handwritten documents." In: *Int. Journal on Document Analysis and Recognition* 22.4 (2019), pp. 387–405.

[Kle+13] F. Kleber, S. Fiel, M. Diem, and R. Sablatnig. "CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA, 2013, pp. 560–564.

[KLP01] S. Kane, A. Lehman, and E. Partridge. *Indexing George Washington's handwritten manuscripts*. Tech. rep. MM-34. Center for Intelligent Information Retrieval, University of Massachusetts Amherst, 2001.

[Kon05] A. Konar. *Computational Intelligence*. Springer, 2005.

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Proc. of the Neural Information Processing Systems Conf.* Lake Tahoe, NV, United States, 2012, pp. 1106–1114.

[KWD14] A. Kovalchuk, L. Wolf, and N. Dershowitz. "A simple and fast word spotting method." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2014.

[LeC+98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324.

[Lee13] D. Lee. "Pseudo-Label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks." In: (2013).

[Li+18] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming. "Improving deep neural network with Multiple Parametric Exponential Linear Units." In: *Neurocomputing* 301 (2018), pp. 11–24.

[Li+23] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. A. F. Florêncio, C. Zhang, Z. Li, and F. Wei. "TrOCR: Transformer-Based Optical Character Recognition with Pre-trained Models." In: *Proc. of the Conf. on Artificial Intelligence*. Washington, DC, USA, 2023, pp. 13094–13102.

[Lla+12] J. Lladós, M. Rusiñol, A. Fornés, D. F. Mota, and A. Dutta. "On the Influence of Word Representations for Handwritten Word Spotting in Historical Documents." In: *Int. Journal on Pattern Recognition and Artificial Intelligence* 26.5 (2012).

[LM81] D. G. Long and A. T. Milne. *The manuscripts of Jeremy Bentham : a chronological index to the collection in the Library of University College London*. Library and Bentham Committee, University College London, 1981.

[LNH09] C. H. Lampert, H. Nickisch, and S. Harmeling. "Learning to detect unseen object classes by between-class attribute transfer." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Miami, Florida, USA, 2009, pp. 951–958.

[Low04] D. G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." In: *Int. Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[LRM04] V. Lavrenko, T. M. Rath, and R. Manmatha. "Holistic word recognition for handwritten historical documents." In: *Workshop on Document Image Analysis for Libraries*. 2004, pp. 278–287.

[LSP06] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. New York, NY, USA: IEEE Computer Society, 2006, pp. 2169–2178.

[Lu+21] N. Lu, W. Yu, X. Qi, Y. Chen, P. Gong, R. Xiao, and X. Bai. "MASTER: Multi-aspect non-local network for scene text recognition." In: *Pattern Recognit.* 117 (2021), p. 107980.

[LZT07] L. Likforman-Sulem, A. Zahour, and B. Taconet. "Text line segmentation of historical documents: a survey." In: *Int. Journal on Document Analysis and Recognition* 9.2-4 (2007), pp. 123–138.

[Mat+21a] M. Mathew, L. Gómez, D. Karatzas, and C. V. Jawahar. "Asking questions on handwritten document collections." In: *Int. Journal on Document Analysis and Recognition* 24.3 (2021), pp. 235–249.

[Mat+21b] A. Mattick, M. Mayr, M. Seuret, A. Maier, and V. Christlein. "SmartPatch: Improving Handwritten Word Imitation with Patch Discriminators." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 12821. Lausanne, Switzerland, 2021, pp. 268–283.

[MB01] U. Marti and H. Bunke. "Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System." In: *Int. J. Pattern Recognit. Artif. Intell.* 15.1 (2001), pp. 65–90.

[MB02] U. Marti and H. Bunke. "The IAM-database: an English sentence database for offline handwriting recognition." In: *Int. Journal on Document Analysis and Recognition* 5.1 (2002), pp. 39–46.

[MGE11] T. Malisiewicz, A. Gupta, and A. A. Efros. "Ensemble of exemplar-SVMs for object detection and beyond." In: *Proc. of the Int. Conf. on Computer Vision*. Barcelona, Spain, 2011, pp. 89–96.

[MHN+13] A. L. Maas, A. Y. Hannun, A. Y. Ng, *et al.* "Rectifier nonlinearities improve neural network acoustic models." In: *Proc. of the Int. Conf. on Machine Learning*. Vol. 30. 1. Atlanta, GA, 2013, p. 3.

[MHR96] R. Manmatha, C. Han, and E. M. Riseman. "Word Spotting: A New Approach to Indexing Handwriting." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. San Francisco, CA, USA, 1996, pp. 631–637.

[Mic+19] J. Michael, R. Labahn, T. Grüning, and J. Zöllner. "Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 1286–1293.

[MM19] B. Moysset and R. O. Messina. "Are 2D-LSTM really dead for offline text recognition?" In: *Int. Journal on Document Analysis and Recognition* 22.3 (2019), pp. 193–208.

[Mon+13] T. Mondal, N. Ragot, J. Ramel, and U. Pal. "A Fast Word Retrieval Technique Based on Kernelized Locality Sensitive Hashing." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA, 2013, pp. 1195–1199.

[Moz89] M. C. Mozer. "A Focused Backpropagation Algorithm for Temporal Pattern Recognition." In: *Complex Systems* 3.4 (1989).

[MP43] W. S. McCulloch and W. Pitts. "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5 (1943), pp. 115–133.

[MP69] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969.

[Mur22] K. P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. URL: probml.ai.

[Nat+01] P. Natarajan, Z. Lu, R. M. Schwartz, I. Bazzi, and J. Makhoul. "Multilingual Machine Printed OCR." In: *Int. Journal on Pattern Recognition and Artificial Intelligence* 15.1 (2001), pp. 43–63.

[Nie15] M. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. URL: http://neuralnetworksanddeeplearning.com/.

[Nik+23] K. Nikolaidou, G. Retsinas, V. Christlein, M. Seuret, G. Sfikas, E. B. Smith, H. Mokayed, and M. Liwicki. "WordStylist: Styled Verbatim Handwritten Text Generation with Latent Diffusion Models." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 14188. San José, CA, USA, 2023, pp. 384–401.

[Ope23] OpenAI. "GPT-4 Technical Report." In: *CoRR* abs/2303.08774 (2023). arXiv: 2303.08774.

[OPH94] T. Ojala, M. Pietikäinen, and D. Harwood. "Performance evaluation of texture measures with classification based on Kullback discrimination of distributions." In: *Proc. of the Int. Conf. on Pattern Recognition*. Jerusalem, Israel, 1994, pp. 582–585.

[OSK18] S. A. Oliveira, B. Seguin, and F. Kaplan. "dhSegment: A Generic Deep-Learning Approach for Document Segmentation." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Niagara Falls, NY, USA, 2018, pp. 7–12.

[PF09] T. Plötz and G. A. Fink. "Markov models for offline handwriting recognition: a survey." In: *Int. Journal on Document Analysis and Recognition* 12.4 (2009), pp. 269–298.

[PG23] G. Paaß and S. Giesselbach. *Foundation Models for Natural Language Processing - Pre-trained Language Models Integrating Media*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, 2023.

[Pha+14] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. "Dropout Improves Recurrent Neural Networks for Handwriting Recognition." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Crete, Greece, 2014, pp. 285–290.

[PP23] D. Parres and R. Paredes. "Fine-Tuning Vision Encoder-Decoder Transformers for Handwriting Text Recognition on Historical Documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 14190. San José, CA, USA, 2023, pp. 253–268.

[PR09] F. Perronnin and J. A. Rodriguez-Serrano. "Fisher Kernels for Handwritten Word-spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Barcelona, Spain, 2009, pp. 106–110.

[Pra+14] I. Pratikakis, K. Zagoris, B. Gatos, G. Louloudis, and N. Stamatopoulos. "ICFHR 2014 competition on handwritten keyword spotting (H-KWS 2014)." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Crete, Greece, 2014, pp. 814–819.

[PRV14] J. Puigcerver, A. H. T. Rossi, and E. Vidal. "Word-Graph and Character-Lattice Combination for KWS in Handwritten Documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Crete, Greece, 2014, pp. 181–186.

[PTV15a] J. Puigcerver, A. Toselli, and E. Vidal. "ICDAR2015 competition on keyword spotting for handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nancy, France, 2015, pp. 1176–1180.

[PTV15b] J. Puigcerver, A. H. Toselli, and E. Vidal. "Probabilistic interpretation and improvements to the HMM-filler for handwritten keyword spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nancy, France, 2015, pp. 731–735.

[Pui17] J. Puigcerver. "Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition?" In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 67–72.

[PW16] A. Poznanski and L. Wolf. "CNN-N-Gram for HandwritingWord Recognition." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA, 2016, pp. 2305–2314.

[Rab89] L. R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition." In: *Proc. IEEE* 77.2 (1989), pp. 257–286.

[Ren+18] G. Renton, Y. Soullard, C. Chatelain, S. Adam, C. Kermorvant, and T. Paquet. "Fully convolutional network with dilated convolutions for handwritten text line segmentation." In: *Int. Journal on Document Analysis and Recognition* 21.3 (2018), pp. 177–186.

[Ren+22] P. Ren, Y. Xiao, X. Chang, P. Huang, Z. Li, X. Chen, and X. Wang. "A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions." In: *ACM Computing Surveys* 54.4 (2022), 76:1–76:34.

[Ret+16] G. Retsinas, G. Louloudis, N. Stamatopoulos, and B. Gatos. "Keyword Spotting in Handwritten Documents Using Projections of Oriented Gradients." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Santorini, Greece, 2016, pp. 411–416.

[Ret+19] G. Retsinas, G. Louloudis, N. Stamatopoulos, and B. Gatos. "Efficient Learning-Free Keyword Spotting." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 41.7 (2019), pp. 1587–1600.

[Ret+21] G. Retsinas, G. Sfikas, C. Nikou, and P. Maragos. "From Seq2Seq Recognition to Handwritten Word Embeddings." In: *Proc. of the British Machine Vision Conf.* Virtual, 2021, p. 98.

[RF87] A. J. Robinson and F. Fallside. *The utility driven dynamic error propagation network*. Vol. 1. University of Cambridge Department of Engineering Cambridge, 1987.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning representations by back-propagating errors." In: *Nature* 323.6088 (1986), pp. 533–536.

[RLF15] P. Riba, J. Lladós, and A. Fornés. "Handwritten word spotting by inexact matching of grapheme graphs." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nancy, France, 2015, pp. 781–785.

[RM07] T. Rath and R. Manmatha. "Word spotting for historical documents." In: *Int. Journal on Document Analysis and Recognition* 9.2–4 (2007).

[Roh+89] J. R. Rohlicek, W. Russell, S. Roukos, and H. Gish. "Continuous hidden Markov modeling for speaker-independent word spotting." In: *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*. Glasgow, UK: IEEE, 1989, pp. 627–630.

[Roj96] R. Rojas. *Neural Networks: A Systematic Inroduction*. Springer Berlin Heidelberg, July 12, 1996. 524 pp.

[Ros+16] A. H. T. Rossi, E. Vidal, V. Romero, and V. Frinken. "HMM word graph based keyword spotting in handwritten document images." In: *Information Sciences* 370-371 (2016), pp. 497–518.

[Ros58] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408.

[Rot+17] L. Rothacker, S. Sudholt, E. Rusakov, M. Kasperidus, and G. A. Fink. "Word Hypotheses for Segmentation-free Word Spotting in Historic Document Images." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017.

[Rot19] L. Rothacker. *Segmentation-free word spotting with bag-of-features hidden Markov models*. Dissertation. Dortmund, Germany, 2019.

[RP12] J. A. Rodriguez-Serrano and F. Perronnin. "A Model-Based Sequence Similarity with Application to Handwritten Word Spotting." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2108–2120.

[RPV15] A. H. T. Rossi, J. Puigcerver, and E. Vidal. "Context-aware lattice based filler approach for key word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nancy, France, 2015, pp. 736–740.

[RPV16] A. H. T. Rossi, J. Puigcerver, and E. Vidal. "Two Methods to Improve Confidence Scores for Lexicon-Free Word Spotting in Handwritten Text." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Shenzhen, China, 2016, pp. 349–354.

[RSN21] G. Retsinas, G. Sfikas, and C. Nikou. "Iterative Weighted Transductive Learning for Handwriting Recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 12824. Lausanne, Switzerland, 2021, pp. 587–601.

[RSN23] G. Retsinas, G. Sfikas, and C. Nikou. "Keyword Spotting Simplified: A Segmentation-Free Approach Using Character Counting and CTC Rescoring." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 14187. San José, CA, USA, 2023, pp. 446–464.

[Rus+11] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. "Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Beijing, China, 2011, pp. 63–67.

[Rus+15a] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. "Efficient segmentation-free keyword spotting in historical document collections." In: *Pattern Recognition* 48.2 (2015), pp. 545–555.

[Rus+15b] O. Russakovsky *et al.* "ImageNet Large Scale Visual Recognition Challenge." In: *Int. Journal on Computer Vision* 115.3 (2015), pp. 211–252.

[Rus+18] E. Rusakov, L. Rothacker, H. Mo, and G. A. Fink. "A Probabilistic Retrieval Model for Word Spotting Based on Direct Attribute Prediction." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Niagara Falls, NY, USA, 2018, pp. 38–43.

[Rus+20] E. Rusakov, T. Somel, G. A. Fink, and G. G. W. Mueller. "Towards Query-by-eXpression Retrieval of Cuneiform Signs." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Dortmund, NRW, Germany, 2020.

[RV13] A. H. T. Rossi and E. Vidal. "Fast HMM-Filler Approach for Key Word Spotting in Handwritten Documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA: IEEE Computer Society, 2013, pp. 501–505.

[RWF21] L. Rothacker, F. Wolf, and G. A. Fink. "Annotation-free word spotting with bag-of-features HMMs." In: *International Journal of Pattern Recognition and Artificial Intelligence* 35.04 (2021), p. 2153001.

[SAC07] M. D. Smucker, J. Allan, and B. Carterette. "A Comparison of Statistical Significance Tests for Information Retrieval Evaluation." In: *Conference on Information and Knowledge Management*. Lisbon, Portugal, 2007, pp. 623–632.

[Sán+16] J. Sánchez, V. Romero, A. H. Toselli, and E. Vidal. "ICFHR2016 Competition on Handwritten Text Recognition on the READ Dataset." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Shenzhen, China, 2016, pp. 630–635.

[Say73] K. M. Sayre. "Machine recognition of handwritten words: A project report." In: *Pattern Recognition* 5.3 (1973), pp. 213–228.

[SB12] R. Saabni and A. M. Bronstein. "Fast Keyword Searching Using 'BoostMap' Based Embedding." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Bari, Italy, 2012, pp. 734–739.

[SB19] V. Storchan and J. Beauschene. "Data Augmentation via Adversarial Networks for Optical Character Recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, NSW, Australia, 2019, pp. 184–189.

[SC78] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition." In: *IEEE transactions on acoustics, speech, and signal processing* 26.1 (1978), pp. 43–49.

[Sch+96] R. M. Schwartz, C. LaPre, J. Makhoul, C. Raphael, and Y. Zhao. "Language-independent OCR using a continuous speech recognition system." In: *Proc. of the Int. Conf. on Pattern Recognition*. Vienna, Austria: IEEE Computer Society, 1996, pp. 99–103.

[SCP17] B. Stuner, C. Chatelain, and T. Paquet. "Self-Training of BLSTM with Lexicon Verification for Handwriting Recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 633–638.

[SCX19] F. Sheng, Z. Chen, and B. Xu. "NRTR: A No-Recurrence Sequence-to-Sequence Model for Scene Text Recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 781–786.

[SF15] S. Sudholt and G. A. Fink. "A Modified Isomap Approach to Manifold Learning in Word Spotting." In: *Proc. of the German Conf. on Pattern Recognition*. Muenster, Germany, 2015, pp. 529–539.

[SF16] S. Sudholt and G. A. Fink. "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Shenzhen, China, 2016, pp. 277–282.

[SF17] S. Sudholt and G. A. Fink. "Evaluating Word String Embeddings and Loss Functions for CNN-based Word Spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 493–498.

[SF18] S. Sudholt and G. A. Fink. "Attribute CNNs for word spotting in handwritten documents." In: *Int. Journal on Document Analysis and Recognition* 21.3 (2018), pp. 199–218.

[SFR16] M. Stauffer, A. Fischer, and K. Riesen. "A Novel Graph Database for Handwritten Word Images." In: *S+SSPR*. Mérida, Mexico, 2016, pp. 553–563.

[SFR18a] M. Stauffer, A. Fischer, and K. Riesen. "Graph-Based Keyword Spotting in Historical Documents Using Context-Aware ausdorff Edit Distance." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vienna, Austria, 2018, pp. 49–54.

[SFR18b] M. Stauffer, A. Fischer, and K. Riesen. "Keyword spotting in historical handwritten documents based on graph matching." In: *Pattern Recognition* 81 (2018), pp. 240–253.

[SGS15] R. K. Srivastava, K. Greff, and J. Schmidhuber. "Training Very Deep Networks." In: *Proc. of the Neural Information Processing Systems Conf.* Montreal, Quebec, Canada, 2015, pp. 2377–2385.

[SH02] S. Sivadas and H. Hermansky. "Hierarchical tandem feature extraction." In: *Proc. of the Int. Conf. on Acoustics, Speech and Signal Processing*. Vol. 1. IEEE. 2002, pp. I–809.

[Shi+16] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai. "Robust Scene Text Recognition with Automatic Rectification." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Las Vegas, NV, USA: IEEE Computer Society, 2016, pp. 4168–4176.

[Soh+20] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. Raffel, E. D. Cubuk, A. Kurakin, and C. Li. "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence." In: *Proc. of the Neural Information Processing Systems Conf.* 2020.

[SP97] M. Schuster and K. K. Paliwal. "Bidirectional recurrent neural networks." In: *Transactions Signal Processing* 45.11 (1997), pp. 2673–2681.

[SR06] J. Schenk and G. Rigoll. "Novel hybrid NN/HMM modelling techniques for on-line handwriting recognition." In: *Proc. of the Int. Workshop on Frontiers in Handwriting Recognition*. La Baule, France, 2006.

[SRF17] S. Sudholt, L. Rothacker, and G. A. Fink. "Query-by-Online Word Spotting Revisited: Using CNNs for Cross-Domain Retrieval." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017.

[SRG16] G. Sfikas, G. Retsinas, and B. Gatos. "Zoning aggregated hypercolumns for keyword spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Niagara Falls, NY, USA, 2016, pp. 283–288.

[SRN17] E. Sabir, S. Rawls, and P. Natarajan. "Implicit Language Model in LSTM for OCR." In: *Workshop on Multilingual, Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 27–31.

[Stu+19] L. Studer, M. Alberti, V. Pondenkandath, P. Goktepe, T. Kolonko, A. Fischer, M. Liwicki, and R. Ingold. "A Comprehensive Study of ImageNet Pre-Training for Historical Document Image Analysis." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, NSW, Australia, 2019, pp. 720–725.

[Sue+18] J. Sueiras, V. Ruiz, Á. Sánchez, and J. F. Vélez. "Offline continuous handwriting recognition using sequence to sequence neural networks." In: *Neurocomputing* 289 (2018), pp. 119–128.

[SVL14] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks." In: *Proc. of the Neural Information Processing Systems Conf.* Montreal, Quebec, Canada, 2014, pp. 3104–3112.

[SZ15] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *Proc. of the Int. Conf. on Learning Representations*. San Diego, CA, USA, 2015.

[Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications*. 5th. Springer, 2010.

[Ten+18] C. Tensmeyer, C. Wigington, B. L. Davis, S. Stewart, T. R. Martinez, and W. Barrett. "Language Model Supervision for Handwriting Recognition Model Adaptation." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Niagara Falls, NY, USA, 2018, pp. 133–138.

[TH12] T. Tieleman and G. Hinton. "Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning." In: *COURSERA Neural Networks Mach. Learn* 17 (2012).

[Tho+15] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, and Y. Kessentini. "A deep HMM model for multiple keywords spotting in handwritten documents." In: *Pattern Analysis and Applications* 18.4 (2015), pp. 1003–1015.

[Tol+17] J. I. Toledo, S. Dey, A. Fornés, and J. Lladós. "Handwriting Recognition by Attribute Embedding and Recurrent Neural Networks." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 1038–1043.

[Ton+14] A. Tong, M. A. Przybocki, V. Märgner, and H. E. Abed. "NIST 2013 Open Handwriting Recognition and Translation (Open HaRT'13) Evaluation." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Tours, France, 2014, pp. 81–85.

[TSM17] C. Tensmeyer, D. Saunders, and T. Martinez. "Convolutional Neural Networks for Font Classification." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Kyoto, Japan, 2017, pp. 985–990.

[TW19] C. Tensmeyer and C. Wigington. "Training Full-Page Handwritten Text Recognition Models without Annotated Line Breaks." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 1–8.

[Vas+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is All you Need." In: *Proc. of the Neural Information Processing Systems Conf.* Long Beach, CA, USA, 2017, pp. 5998–6008.

[VHF19] E. Vats, A. Hast, and A. Fornés. "Training-Free and Segmentation-Free Word Spotting using Feature Matching and Query Expansion." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 1294–1299.

[Vil+16] M. Villegas, J. Puigcerver, A. H. Toselli, J. Sánchez, and E. Vidal. "Overview of the ImageCLEF 2016 Handwritten Scanned Document Retrieval Task." In: *Conference and Labs of the Evaluation forum (CLEF)*. Vol. 1609. Evora, Portugal, 2016, pp. 233–253.

[Vit67] A. J. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." In: *IEEE Trans. Inf. Theory* 13.2 (1967), pp. 260–269.

[Was54] G. Washington. "George Washington papers." In: *Letterbook 1, Aug. 11, 1754 - Dec. 25, 1755*. Series 2, Letterbooks 1754 to 1799. Washington, DC: Library of Congress, 1754. URL: https://www.loc.gov/item/mgw2.001/.

[WB16] T. Wilkinson and A. Brun. "Semantic and verbatim word spotting using deep neural networks." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Shenzhen, China, 2016, pp. 307–312.

[WBF20] F. Wolf, K. Brandenbusch, and G. A. Fink. "Improving Handwritten Word Synthesis for Annotation-free Word Spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Dortmund, Germany (virtual), 2020, pp. 61–66.

[WF20] F. Wolf and G. A. Fink. "Annotation-free Learning of Deep Representations for Word Spotting using Synthetic Data and Self Labeling." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vol. 12116. Winner of the Nakano Best Paper Award. Wuhan, China (virtual): Springer, 2020, pp. 293–308.

[WF22a] F. Wolf and G. A. Fink. "Combining Self-Training and Minimal Annotations for Handwritten Word Recognition." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Vol. 13639. Hyderabad, India, 2022, pp. 300–315.

[WF22b] F. Wolf and G. A. Fink. "Self-Training of Handwritten Word Recognition for Synthetic-to-Real Adaptation." In: *Proc. of the Int. Conf. on Pattern Recognition*. Montreal, Canada: IEEE, 2022, pp. 3885–3892.

[WF24] F. Wolf and G. A. Fink. "Self-Training for Handwritten Word Recognition and Retrieval." In: *Int. Journal on Document Analysis and Recognition* 27.3 (2024), pp. 225–244.

[WFS03] M. Wienecke, G. A. Fink, and G. Sagerer. "Towards Automatic Video-based Whiteboard Reading." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Edinburgh, Scotland: IEEE Computer Society, 2003, pp. 87–91.

[Wig+17] C. Wigington, S. Stewart, B. L. Davis, B. Barrett, B. L. Price, and S. Cohen. "Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. 2017, pp. 639–645.

[Wig+18] C. Wigington, C. Tensmeyer, B. L. Davis, W. A. Barrett, B. L. Price, and S. Cohen. "Start, Follow, Read: End-to-End Full-Page Handwriting Recognition." In: *Proc. of the European Conf. on Computer Vision*. Vol. 11210. Lecture Notes in Computer Science. Munich, Germany, 2018, pp. 372–388.

[WLB17] T. Wilkinson, J. Lindström, and A. Brun. "Neural Ctrl-F: Segmentation-Free Query-by-String Word Spotting in Handwritten Manuscript Collections." In: *Proc. of the Int. Conf. on Computer Vision*. Venice, Italy, 2017, pp. 4443–4452.

[WOF19] F. Wolf, P. Oberdiek, and G. A. Fink. "Exploring Confidence Measures for Word Spotting in Heterogeneous Datasets." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sydney, Australia, 2019, pp. 583–588.

[Wol96] D. H. Wolpert. "The Lack of A Priori Distinctions Between Learning Algorithms." In: *Neural Comput.* 8.7 (1996), pp. 1341–1390.

[WRF16] C. Wieprecht, L. Rothacker, and G. A. Fink. "Word Spotting in Historical Document Collections with Online-Handwritten Queries." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Santorini, Greece, 2016.

[WZG22] C. Wick, J. Zöllner, and T. Grüning. "Rescoring Sequence-to-Sequence Models for Text Line Recognition with CTC-Prefixes." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Vol. 13237. La Rochelle, France, 2022, pp. 260–274.

[Xu+15] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention." In: *Proc. of the Int. Conf. on Machine Learning*. Vol. 37. Lille, France, 2015, pp. 2048–2057.

[Yan+22] X. Yang, Z. Song, I. King, and Z. Xu. "A survey on deep semi-supervised learning." In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[You96] S. J. Young. "A review of large-vocabulary continuous-speech recognition." In: *IEEE Signal Process. Mag.* 13.5 (1996), p. 45.

[ZC88] Y. Zhou and R. Chellappa. "Computation of optical flow using a neural network." In: *Int. Conf. on Neural Networks*. San Diego, CA, USA, 1988, pp. 71–78.

[ZF14] M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks." In: *Proc. of the European Conf. on Computer Vision*. Vol. 8689. Zurich, Switzerland, 2014, pp. 818–833.

[Zha+23] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola. *Dive into Deep Learning*. https://D2L.ai. Cambridge University Press, 2023.

[Zhu+23] Y. Zhu, Z. Li, T. Wang, M. He, and C. Yao. "Conditional Text Image Generation with Diffusion Models." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Vancouver, BC, Canada, 2023, pp. 14235–14244.

[ZPG17] K. Zagoris, I. Pratikakis, and B. Gatos. "Unsupervised word spotting in historical handwritten document images using document-oriented local features." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 26.8 (2017), pp. 4032–4041.