FROM HANDWRITTEN WORDS TO CUNEIFORM SIGNS RETRIEVAL USING SEMANTIC ATTRIBUTES

Dissertation
zur Erlangung des Grades eines

DOKTORS DER INGENIEURWISSENSCHAFTEN

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

EUGEN RUSAKOV

Dortmund

2024

Tag der mündlichen Prüfung:	10.12.2024
Dekan:	Prof. Dr. Jens Teubner
Gutachter:	Prof. Dr.-Ing. Gernot A. Fink Prof. Dr. Xiaoyi Jiang

ACKNOWLEDGMENTS

When I reflect on the beginning of my journey as a PhD student, I never could have anticipated the experiences that awaited me throughout my doctorate. At that time, I applied for a position and assumed I would be turned down due to my lack of background in pattern recognition. Nevertheless, Gernot offered me a one-year contract for a "doctoral preparation year". Initially, I was interested in working with embedded systems. However, while working in Gernot's research group, I excited my curiosity for document image analysis, which ultimately became the focus of my dissertation. Completing this dissertation would not have been possible without the immense support of incredible people, and I would like to express my gratitude to them.

First, I would like to express my gratitude to Prof. Dr.-Ing. Gernot A. Fink for his supervision. Gernot, thank you for consistently taking the time to share your knowledge and for your invaluable support throughout the years. I am also grateful for the opportunity to pursue this path and for the freedom I had to follow my research direction. I would like to extend my thanks to my second supervisor, Prof. Dr. Xiaoyi Jiang, for promptly agreeing to review my dissertation and for patiently waiting for its submission. I also appreciate the participation of Prof. Dr. Mario Botsch and Prof. Dr. Stefan Harmeling as members of the examination committee.

During my first year of my PhD, I met fantastic colleagues, and I would like to thank Dr. René Grzeszick, Dr. Sebastian Sudholt, Dr. Axel Plinge, Julian Kürby, and especially Dr. Leonard Rothacker. Your support during my initial years helped me navigate the world of pattern recognition. From my second year onward, I had the pleasure of sharing my office with Dr. Fernando Moya. Thank you, Fernando, for being such a wonderful colleague and probably the best office neighbor in the world. Your enthusiasm and energy have always inspired and motivated me. I am also grateful to have worked alongside great colleagues, including Dr. Fabian Wolf, Dr. Philipp Oberdiek, Kai Brandenbusch, Dominik Koßmann, and Dr. Oliver Tüselmann. Thanks to all of you, I have fond memories of our time together at the chair, our experiences in the office, and the conferences and summer schools we attended.

I would like to dedicate this final paragraph to my family. You have always encouraged me to forge my own path and pursue my academic interests. I owe the completion of my studies and the achievement of my doctoral degree to your unwavering support. Finally, my deepest thanks to Leevke Wilkens. I cannot express enough my appreciation for your advice, assistance, and support, especially during challenging times. I am truly grateful for your willingness to sacrifice our time together and for your immense patience that allowed me to complete my work.

ABSTRACT

Archives all over the world store a vast number of manuscripts that contain invaluable information on cultural heritage. Extracting the content from these manuscripts requires considerable human effort. Being able to sift through extensive manuscript collections quickly is particularly interesting for historical documents. For this purpose, pattern recognition methods have been used to transform handwritten texts from document images into machine-readable formats fully automatically. Well-known approaches are based on recognizing individual letters or words to transcribe a text. While these approaches achieve outstanding performance for machine-printed documents, their results for handwritten texts often fail to meet expectations, especially for collections of historical documents. Handwritten texts from old manuscripts typically exhibit sundry degradations and large variability in writing styles. Faced with these challenges, document analysis approaches have increasingly adopted retrieval-based methods since they do not rely on individual letter or word recognition but rather evaluate similarities between a query and segmented parts of document images. The obtained retrieval list contains document parts sorted in descending order according to their similarity to the query. This feature lets the user compare the list's contents and determine relevant items. Words in handwritten documents are often the focus of interest, leading to the task of retrieving words being called word spotting in the document image analysis community. This thesis presents a methodology for segmentation-based word spotting in handwritten documents. Word images and textual strings must first be transformed into numerical representations to obtain a sorted retrieval list. Subsequently, a similarity between two representations is evaluated by a particular measure. This thesis utilizes the highly successful word embedding technique pyramidal histogram of characters (PHOC). This embedding is inspired by semantic attributes where textual strings are decomposed in a pyramidal scheme, and the occurrences of individual characters are indicated using binary values. A similarity between two numerical word representations is assessed through a novel measurement named probabilistic retrieval model (PRM). This model evaluates the probability between two binary-valued semantic attribute representations based on the assumption that PHOC attributes are Bernoulli distributed. The similarities obtained from the PRM highly depend on the quality of predicted attributes. While representations from textual strings are obtained through the PHOC algorithm, handwritten word images must be transformed first. This thesis uses convolutional neural networks (CNNs) to predict PHOC representations for corresponding word images. During the last decade, these networks have consistently achieved state-of-the-art results in various computer vision tasks. As a result, CNNs are nowadays the de-facto standard models for image classification. The presented method applies a statistical framework named generalized linear models (GLMs) to derive the binary cross-entropy loss (BCEL) as the suitable loss function for the PRM similarity measure. The BCEL is

subsequently used to train CNN models to estimate binary attribute probabilities accurately. A significant advantage is a direct connection between the binary cross-entropy loss and the probabilistic retrieval model. Minimizing the BCEL function is equivalent to maximizing the PRM similarity between two equal PHOC representations. This word-spotting methodology is adapted to an ancient writing system called cuneiform. Cuneiform signs are formed by characteristic wedge-shaped impressions. Norbert Gottstein proposed a representation based on alphanumeric expressions, describing a cuneiform sign according to their wedge impressions. These alphanumeric expressions are used to define a set of semantic attributes named Gottstein representation. This thesis further develops the holistic Gottstein representation, including spatial information. The wedge positions are encoded by applying a pyramidal segmentation, which yields binary attribute representations indicating wedge semantics and their approximate position within the cuneiform sign. This thesis presents two approaches to decomposing cuneiform signs according to predefined pyramidal schemes. The first approach divides cuneiform signs horizontally and vertically, following a grid-like pattern called spatial pyramid Gottstein (SPG) representation. The second approach is based on annotations describing wedge constellations according to their sequential order. These sign encodings are assigned to pyramidal splits by applying the PHOC algorithm, and the resulting representation is referred to as temporal pyramid Gottstein (TPG). By using these representations, signs can now be expressed by their wedge constellations and types, which enables the method to perform cuneiform sign spotting in a novel retrieval scenario named query-by-expression (QbX). This thesis proposes three core contributions: the design of the probabilistic retrieval model as a novel similarity measure, deriving the binary cross-entropy as the suitable loss function for the PRM, and two different cuneiform sign representations based on wedge impressions encoded as binary semantic attributes. These contributions are evaluated on six benchmarks in total. Four include handwritten text documents, and two benchmarks contain images of cuneiform tablets. The experiments show that combining the PRM and BCEL achieves state-of-the-art results and even exceeds the performance using other combinations of similarity measures and loss functions. Representing cuneiform signs by their wedge impressions enables the user to query a database without the necessity of visual examples. Moreover, the use of pyramidal decomposition provides a more detailed description of cuneiform signs, leading to increased retrieval performance.

PUBLICATIONS

[Rot+17] Leonard Rothacker, Sebastian Sudholt, Eugen Rusakov, Matthias Kasperidus, and Gernot A. Fink. "Word Hypotheses for Segmentation-Free Word Spotting in Historic Document Images." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 1174–1179.

[Rus+18] Eugen Rusakov, Leonard Rothacker, Hyunho Mo, and Gernot A. Fink. "A Probabilistic Retrieval Model for Word Spotting Based on Direct Attribute Prediction." In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2018, pp. 38–43.

[Rus+19] Eugen Rusakov, Kai Brandenbusch, Denis Fisseler, Turna Somel, Gernot A. Fink, Frank Weichert, and Gerfrid G. W. Müller. "Generating Cuneiform Signs with Cycle-Consistent Adversarial Networks." In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. 2019, pp. 19–24.

[Rus+20] Eugen Rusakov, Turna Somel, Gernot A. Fink, and Gerfrid G.W. Müller. "Towards Query-by-eXpression Retrieval of Cuneiform Signs." In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, pp. 43–48.

[Rus+21] Eugen Rusakov, Turna Somel, Gerfrid G. W. Müller, and Gernot A. Fink. "Embedded Attributes for Cuneiform Sign Spotting." In: *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II*. Springer-Verlag, 2021, pp. 291–305.

[BRF21] Kai Brandenbusch, Eugen Rusakov, and Gernot A. Fink. "Context Aware Generation of Cuneiform Signs." In: *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I*. Springer-Verlag, 2021, pp. 65–79.

[Alt+22] Erik Altermann, Fernando Moya Rueda, Eugen Rusakov, and Gernot A. Fink. "Retrieval-based Annotation of Multi-channel Time-Series Data for HAR." In: *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. 2022, pp. 212–217.

CONTENTS

ACRONYMS

Adam	adaptive moment estimation
AP	average precision
AwA	animals with attributes
BCE	binary cross-entropy
BCEL	binary cross-entropy loss
BLSTM	bidirectional long short-term memory
BoC	bag of characters
BoF	bag of features
BoF-HMM	bag of features HMM
BoW	bag of words
CBIR	content-based image retrieval
CE	cross-entropy
CNN	convolutional neural network
CTC	connectionist temporal classification
DAP	direct attribute prediction
DCT	discrete Cosine transform
DCToW	discrete Cosine transform of words
DFT	discrete Fourier transform
DoG	difference of Gaussians
DTW	dynamic time warping
ELU	exponential linear unit
EM	expectation maximization
FC	fully connected
GLM	generalized linear model
GMM	Gaussian mixture model
GNN	graph neural network
GR	Gottstein representation
GW	George Washington

HMMs	hidden Markov models
HOG	histogram of oriented gradients
i.i.d.	independent and identically distributed
IAM-DB	IAM handwriting database
IAP	indirect attribute prediction
ICP	iterated closed points
ILSVRC	ImageNet large scale visual recognition challenge
KCCA	kernelized canonical correlation analysis
kNN	k-nearest-neighbor
LBP	local binary pattern
LReLU	leaky rectified linear unit
LSA	latent semantic analysis
LSDE	levenshtein space deep embedding
LSI	latent semantic indexing
LSTM	long short-term memory
MAE	mean absolute error
mAP	mean average precision
MLE	maximum likelihood estimation
MLP	Multi-Layer Perceptron
MSE	mean squared error
MSII	multi-scale integral invariants
NLL	negative log-likelihood
OCR	optical character recognition
OOV	out-of-vocabulary
ORB	oriented FAST and rotated BRIEF
PDF	probability density function
PHOC	pyramidal histogram of characters
PMF	probability mass function
PReLU	parametric rectified linear unit
PRM	probabilistic retrieval model

QbE	query-by-example
QbO	query-by-online-trajectory
QbS	query-by-string
QbSP	query-by-speech
QbW	query-by-word
QbX	query-by-expression
ReLU	rectified linear unit
RNN	recurrent neural network
RoI	regions of interest
RPN	region proposal network
SGD	stochastic gradient descent
SIFT	scale invariant feature transform
SLP	Single-Layer Perceptron
SPG	spatial pyramid Gottstein
SPM	spatial pyramid matching
SPOC	spatial pyramid of characters
SPP	spatial pyramid pooling
SVG	scalable vector graphics
SVM	support vector machine
TPG	temporal pyramid Gottstein
TPP	temporal pyramid pooling

$a, b, c, \ldots$	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
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots$	multivariate 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
$\text{diag}(\mathbf{A})$	the main diagonal of $\mathbf{A}$
$\mathbf{A}^{(i)}$	the i-th element in set $\mathbf{A}$
$\lvert\mathbf{A}\rvert$	the cardinality of set $\mathbf{A}$
$\bar{\mathbf{A}}$	average value of the set $\mathbf{A}$, i.e., $\frac{1}{\lvert\mathbf{A}\rvert}\sum_{i=1}^{\lvert\mathbf{A}\rvert}\mathbf{A}^{(i)}$
$\mathbf{a}^T$	the transpose of vector $\mathbf{a}$
$\hat{a}, \hat{\mathbf{a}}$	estimate or prediction of a scalar or vector
a_i	the i-th element of vector $\mathbf{a}$
$a_{i,j}$	the element of the matrix $\mathbf{A}$ at row i and column j
$a_i^{(j)}$	the i-th element of the j-th vector in set $\mathbf{S} = \left\{\mathbf{a}^{(j)}\right\}_{j=1}^{n}$
$f(x), f(\mathbf{x})$	scalar function with scalar or vector argument
$\mathbf{f}(\mathbf{x})$	vector function with vector argument
$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_i}$	the partial derivative of $\mathbf{f}$ with respect to x_i
$\mathbb{E}[X]$	the expected value of the univariate random variable X
$\langle \mathbf{a}, \mathbf{b} \rangle$	inner product between the vectors $\mathbf{a}$ and $\mathbf{b}$

1 INTRODUCTION

Searching document collections is essential for analyzing and interpreting their content. Typically, the text is examined for particular words such as names, places, or dates. This function is commonly used for digital documents, as the machine-readable text can be automatically searched. Unfortunately, this functionality is not directly available for documents that were not created digitally. This type of document needs to be converted into a machine-readable form first. The common approaches use the findings and methods of pattern recognition and computer vision for this purpose. In this context, the most widely known approaches are referred to as OCR. Methods for optical character recognition have a long tradition in the field of computer science and date back to the early 1920's [Sch82]. These approaches achieve outstanding recognition performance for standardized machine-printed documents [Smi07]. The remarkable performance is mainly due to the limited variability in the visual appearance of modern fonts or high quality document images. However, the results are corrupted as the visual variability increases. This is especially true for handwritten documents [Lla+12]. Furthermore, OCR systems usually make hard classification decisions, which can be another source of error. Introducing a post-processing step that utilizes language models to correct misclassifications mitigates this disadvantage. However, these models often require large training data and a reasonable assumption regarding the vocabulary being used. Both criteria are particularly difficult to fulfill for historical document collections. In these scenarios, retrieval-based approaches often achieve more robust results than approaches based on recognition [Gio+17]. The core idea of retrieval-based approaches is to compare a given *query* to all *candidates* from a database and return a list of the *most similar* entries. Comparing similarities makes the class decision obsolete and prevents the possibility of incorrect classifications. These similarities are usually determined based on *visual appearance* or through an *embedding*. A *similarity* can be a binary value indicating whether a candidate is relevant for the given query. However, most retrieval-based methods use measures assessing real-valued numbers, indicating the degree of similarity between two representations. Although these approaches support a broad range of queries such as individual characters, text lines, or entire sentences, the most commonly used type of query is words. In the field of document image analysis, the task of retrieving specific words is known as *word spotting*. Word spotting techniques are particularly valuable for historical document collections. Archives around the world contain numerous manuscripts holding untapped knowledge about our cultural heritage. Reviewing these collections is labor-intensive and demands significant human effort. By enabling these manuscripts to be searchable, historians can access more precise and previously unknown information about past societies.

This thesis presents a methodology for word spotting in handwritten documents and adopts this retrieval-based approach to an ancient writing system named *cuneiform*. The retrieval-based method utilizes a popular word embedding named *PHOC* initially introduced in [Alm+13]. This embedding decomposes textual strings in a *pyramidal fashion*, where binary-valued *semantic attributes* represent individual or combinations of characters. By using a pyramidal decomposition, the embedding is able to encode the approximate position of the predefined semantics. PHOC has proven to be a powerful representation, achieving state-of-the-art performance for word spotting in handwritten documents [Gio+17]. The semantic attributes are based on the assumption that different classes *share similar characteristics*. A simple approach for word spotting is to consider words as holistic classes and the individual characters as their characteristic semantics. The assessment of similarity between numerical representations is crucial for retrieval-based approaches. This thesis introduces a new *similarity measure* called *PRM*, inspired by the *DAP* approach initially presented in [LNH09]. The authors use semantic attributes for zero-shot classification and introduce the AwA database [LNH09]. This dataset contains images of various animals with corresponding binary attribute annotations describing the animals' characteristics. The animal classes are mutually exclusive during training and testing. However, unseen animals are recognized using a combination of their semantic characteristics (binary attributes) learned during the training phase. The DAP evaluates these attribute predictions and determines the animal class using the nearest-neighbor approach. In analogy, the PRM similarity measure evaluates the probabilities of PHOC representations being similar. These probabilities indicate the similarities and are utilized to rank the retrieved lists. Accurate assessment of these similarities requires robust attribute estimation. Therefore, the method applies the *GLMs* framework to create a statistical model for attribute probability estimation and then connects it to *neural networks*. The statistical model is transformed into the well-known *BCE loss* and its respective output scaling. Training a neural network with the BCE loss has a significant advantage. Minimizing the loss error is equivalent to maximizing the probability between two similar PHOC vectors evaluated by the PRM. *CNNs* are applied to obtain high-quality estimations of attribute probabilities. These networks transform images of handwritten words or cuneiform signs into their respective attribute representations. CNNs started to dominate the field of computer vision almost a decade ago, and they remain the standard models for image recognition tasks today. Their ability to jointly optimize feature extraction and classification, known as *end-to-end* training, gives them a significant advantage over conventional approaches.

The word spotting approach is adapted for the ancient cuneiform writing system, which was utilized for writing multiple languages in the Near East region. It consists of signs that are formed by *characteristic wedge-shaped impressions*. In Latin, these impressions are called *cuneus*, which is where the writing system gets its name. Norbert Gottstein utilized these wedge impressions and proposed a cuneiform sign representation based on alphanumeric expressions [Got13]. A key contribution of the proposed method is the transformation of these alphanumeric expressions, representing cuneiform signs, to binary-valued

attribute vectors referred to as the *GR*. Furthermore, this wedge-based representation is further developed to include *spatial information*, depicting a wider variety of cuneiform signs in greater detail. This thesis proposes two annotation approaches using different partitioning schemes. The first approach divides cuneiform signs in horizontal and vertical direction according to a *grid-like* pattern that resembles the *spatial pyramid* approach as proposed in [LSP06]. Consequently, the resulting representation is referred to as *SPG*. The second approach annotates cuneiform signs according to their reading direction following a *sequence-like* pattern inspired by the sequential order of characters in Latin words. The resulting sign encodings are assigned to pyramidal splits applying the PHOC algorithm. Referring to the *TPP* [SF17], where features of convolutional neural networks are split along the horizontal direction, the cuneiform sign representation is named *TPG*. Describing signs by their wedge characteristics allows for querying cuneiform databases using expressions instead of visual examples. This capability introduces a new cuneiform sign-spotting scenario called *QbX*.

1.1 CONTRIBUTIONS

The methodology presented in this thesis has partially been published at scientific conferences in the field of document image analysis. The research has been carried out in the *Pattern Recognition Group* at TU Dortmund University under the supervision of Prof. Dr.-Ing. Gernot A. Fink. The author of this thesis implemented the presented methodology using the programming language Python[1] in combination with the deep learning framework PyTorch[2] and performed all the experiments unless noted otherwise. The method supports segmentation-based QbE and QbS word spotting, as well as QbX retrieval for cuneiform signs. The following paragraphs outline the methodological contributions proposed in this thesis and reference the published components of the method.

Similarity Measure based on Attribute Probabilities

The similarity measure is a crucial component in the retrieval process. A core contribution is a mathematically substantiated measurement assessing the similarity between two numerical representations according to their binary attribute probabilities. This similarity measure is referred to as *PRM* and is inspired by the *DAP* initially proposed in [LNH09]. The model interprets the binary values of the PHOC representation as *Bernoulli-distributed random variables*. Consequently, the similarity between two PHOC representations is determined by assessing the individual probabilities of corresponding attributes. The PRM similarity measure was initially published and partially evaluated by the author of this thesis in [Rus+18], where it received the best student paper award. The author was responsible for the rigorous mathematical formulation of this model and for conducting the experimental evaluation.

1 https://www.python.org/
2 https://pytorch.org/

Loss Function for estimating Attribute Probabilities

The transformation of images into numerical representations is another crucial process for retrieval systems. The quality of attribute representations greatly impacts retrieval performance. In the proposed method, *convolutional neural networks CNNs* are used to obtain accurate estimates for attribute probabilities. These network models have the advantage of optimizing feature extraction and classification simultaneously. To train the neural networks, a suitable loss function is required. A significant aspect of the method is the use of *generelized linear models (GLMs)* to derive a mathematically supported loss function called *binary cross-entropy (BCE)* loss. Training a convolutional neural network with the BCE loss implicitly enhances the quality of the *probabilistic retrieval model (PRM)*, assessing the similarity of two similar representations.

Attribute Representation for Cuneiform Sign Spotting

The concept of semantic attributes has proven to provide powerful representations. Consistently state-of-the-art results are achieved by word spotting approaches utilizing the PHOC representation [Gio+17]. This concept is adapted by the proposed method to represent cuneiform signs that are formed by wedge-shaped impressions. In [Got13], Gottstein introduce an alphanumeric encoding based on four wedge types and their respective counts. The alphanumeric encoding is transformed into binary-valued attribute vectors referred to as *GR*. This representation enable a retrieval system to perform *query-by-expression (QbX)* cuneiform sign spotting. First results, applying the Gottstein representation have been published by the author of this thesis in [Rus+20]. In addition, the GR is enriched by spatial information to represent cuneiform signs in greater detail. The first approach assign the wedge encodings to a predefined grid-like pyramidal pattern named *SPG* representation. An initial evaluation, have been published by the author of this thesis in [Rus+21]. The second annotation approach follows a sequence-like pattern and assign the wedge encodings of a sign according to their order in horizontal direction. The resulting representation is referred to as *TPG*.

Experimental Evaluation of the Probabilistic Retrieval Model, Architecture Components, and Cuneiform Signs Representations

The proposed method is extensively evaluated through experiments on four word spotting and two cuneiform sign spotting benchmarks. This thesis compare the PRM to three other similarity measures: *Cosine similarity*, *Euclidean distance*, and *Cityblock distance*. The comparison includes CNN models trained with different loss functions such as *Cosine loss*, *MSE*, and *MAE*. Additionally, the experiments evaluate the essential components of the CNN architecture regarding retrieval performance on two word spotting benchmarks. The last part of the experiments focuses on the evaluation of the proposed cuneiform sign representation. Experiments for both SPG and TPG are carried out using different pyramid levels. Finally, the results obtained from these representations are compared with each other.

1.2 OUTLINE

The current chapter gives an introduction and motivates the approach presented in this thesis, the remaining chapters are organized as follows:

CHAPTER 2 - FUNDAMENTALS

The methodology is part of the field of pattern recognition, specifically in computer vision. This chapter provides the essential fundamentals of these two areas of research to establish a strong understanding of the presented approach. It encompasses important basic concepts and the pattern recognition processing pipeline. Additionally, it discusses the functionality and architecture of neural networks.

CHAPTER 3 - KNOWLEDGE TRANSFER BETWEEN CATEGORIES

The presented retrieval-based approach is founded on semantic attributes. This chapter delves into the concept of semantic attributes and their preferred applications. It also discusses the advantages and disadvantages, along with influential publications in this field.

CHAPTER 4 - WORD SPOTTING ON HANDWRITTEN DOCUMENTS

In this thesis, word spotting is the application of interest. Chapter 4 gives an overview of word spotting and explains the established terminology used in the document analysis community. Furthermore, crucial aspects such as word image representations and the assessment of similarity are discussed in more detail.

CHAPTER 5 - RELATED WORK

Chapter 5 reviews related work that presents approaches for embedding word strings, convolutional neural network architectures for processing document images, and the direct attribute prediction method for zero-shot classification, which inspired the probabilistic retrieval model proposed in this thesis.

CHAPTER 6 - WORD SPOTTING WITH ATTRIBUTE PROBABILITIES

This chapter introduces the retrieval-based methodology for segmentation-based word-spotting proposed in this thesis. It explains the mathematical fundamentals of the probabilistic retrieval model, which serves as the similarity measure. Additionally, it introduces the generalized linear models and uses them to derive a loss function specifically adapted to the similarity measure.

CHAPTER 7 - CUNEIFORM SIGN SPOTTING

Adapting the word spotting methodology from the previous chapter to the cuneiform writing system is the second application of interest in this thesis. This chapter introduces the cuneiform writing system and reviews important related research in this area. The following part of the chapter explains how the word embedding approach based on binary attributes is adapted to cuneiform signs in order to enable a novel sign spotting scenario based on the description of wedge constellations.

CHAPTER 8 - EVALUATION

This chapter presents the results for segmentation-based word spotting and cuneiform sign spotting. Therefore, the six benchmarks containing handwritten texts and cuneiform scripts are introduced. Subsequently, the evaluation protocols and experiment configurations are explained. After that, the performance values obtained are listed and compared to selected results from the literature. Qualitative retrieval results and attribute statistics are shown in addition to quantitative numbers.

CHAPTER 9 - CONCLUSION

The final chapter summarizes the content of this thesis and presents the findings and insights drawn from the experimental evaluation. This chapter concludes with the outlook and possible future work.

In addition to citing articles from the literature, this thesis cites internet URLs and references to other sources, like the software libraries used to implement the presented method and experiments. All URL references in this thesis were last checked for availability and content on January 12, 2025.

2

FUNDAMENTALS

The methodology presented in this thesis requires images to be transformed into numerical representations. Therefore, approaches from the field of pattern recognition are applied. In general, pattern recognition methods strive to imitate the human ability of perception [DHS01, p.1]. However, these methods are based on mathematical-technical considerations compared to biological systems. As the data to be processed consists of images containing handwritten words or cuneiform signs, the retrieval-based approach presented in this thesis can be considered as a visual problem. The branch of pattern recognition that addresses visual problems is known as computer vision [Sze22, p.3]. Computer vision tasks include classifying images, segmenting or locating salient objects, and retrieving relevant instances from databases.

This chapter explains the fundamentals of pattern recognition and computer vision in general and neural networks in particular. The outline of this chapter is structured as follows. The ensuing section (Section 2.1) briefly introduces pattern recognition and computer vision. Based on this, the crucial concepts of *feature extraction* (Section 2.1.1) and *classification* (Section 2.1.2) are explained. Section 2.3.3 introduces the functionality of *neural networks* with the architectural structure of the *Perceptron* (Section 2.2.1) as well as the *Multi-Layer Perceptron* (Section 2.2.2), followed by the *training* procedure (Section 2.2.3) and established techniques for *generalization* and *regularization* (Section 2.2.4). The convolutional neural networks are explained in Section 2.3, starting with the two essential building blocks, namely the convolutional layer (Section 2.3.1) and the pooling layer (Section 2.3.2). Furthermore, this section addresses the idea of stacking multiple convolutional layers in order to obtain a *deep neural network* (Section 2.3.3) and the associated *vanishing gradient problem* (Section 2.3.4).

2.1 PATTERN RECOGNITION AND COMPUTER VISION

Pattern recognition is the task of mapping a given input to its corresponding label. The decision towards a label requires specific processing steps. As a result, the large variety of pattern recognition approaches are guided by a standard pipeline [DHS01, p.10]. The regular processing steps through this pipeline are visualized in Figure 1. Every pattern recognition system requires an annotated set of samples with corresponding labels for a classification task. *Data acquisition* is, thus, the first step in the processing pipeline. Although the input data can be arbitrary, ranging from audio samples over three-dimensional spaces to even handwriting trajectories, this thesis is interested in two-dimensional images. Therefore, in the following, the words data or samples refer to a set containing images and

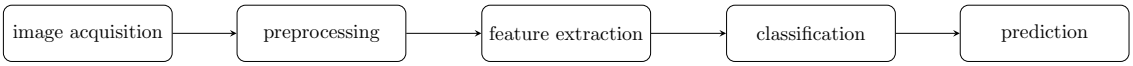


Figure 1: Visualization of a typical pattern recognition pipeline showing the five essential information processing steps.

their corresponding labels unless stated otherwise. Once the data is collected, the content needs to be preprocessed in order to reduce the amount of undesirable variance. According to [Sze22, p.87], *preprocessing* is applied to convert the image into a form suitable for further analysis. Preprocessing operations include color balancing, reducing noise, increasing sharpness, or straightening the image. After the preprocessing, the ensuing step is to extract features representing the image. An appropriate *feature extraction* is crucial for robust classification performance. These features are subsequently used by the *classification* step to assign the label. As mentioned in [DHS01, p.11], the distinction between feature extraction and classification is somewhat subjective. Ideal feature extraction should result in a representation that significantly simplifies the classification process. On the other hand, a highly capable classifier may not require advanced feature extraction. The classification step requires a *classifier model* that decides for a class using a particular decision rule. There exist various classifiers, such as statistical models, linear discriminant functions, or multi-layer neural networks [DHS01, p.4]. However, the model structure is not determined automatically; instead, it must be defined by a human expert. It has been shown that there exists no superior classifier without prior assumptions about the nature of the classification task [DHS01, p.454]. The considerations on this issue are known as the "no free lunch theorem" originally proposed in [Wol96]. The vast majority of classifiers are parametrized models, and once a decision on the classifier has been made, the *optimal* set of model *parameters* needs to be obtained. This process is generally known as *training*, where the classifier *learns* to predict the correct label for a given image input. The purpose of a training task is to *minimize* the *costs*, which are usually defined as a mathematical function that measures a discrepancy between the actual output of the classifier and the desired output defined by the label. Training a model is an iterative process in which the parameters are gradually adjusted to increase the classifier's accuracy. From the perspective of a classification model, "being guided" to predict desired outputs according to the annotations is known as *supervised learning*.

2.1.1 *Feature Extraction and Image Representation*

Feature extraction is applied to obtain a numerical representation of the (preprocessed) input that is subsequently evaluated by the classifier. The key is to design feature representations that expose *between-class variations*, which is the tendency for classes to look different from one another, and suppress *within-class variations*, the tendency for

instances within a class to look different [Dav12, p.482]. Consequently, numerical representations from the same class should remain close to each other in the *feature space*, while representations from two different classes should be far apart. In the context of pattern recognition, *closness* of numerical representations always refers to a mathematical distance measure like the Euclidean distance. This insight leads to the engineering approach of choosing *dicriminative features* that are *invariant* to irrelevant transformations of the input [DHS01, p.11]. Features extracted from images can represent different aspects such as color, light/shading, geometry, or texture. Features are inherently not superior to one another, and their selection heavily depends on the visual domain and the task at hand. For example, engineered features that only consider contrast differences between the dark foreground (ink) and the bright background (paper) could be discriminative enough to distinguish whether a given image contains handwritten text. However, if the semantics of a handwritten text, such as characters or words, have to be distinguished, contrast-based features might be insufficient. Sophisticated features taking the shape of pen strokes into account are the better choice. Since a pattern recognition system's best possible performance strongly depends on the interaction between feature extraction and classification, selecting an optimal feature extraction could be determined in a closed-loop optimization. However, this loop is open for classic approaches where feature representations and classifiers are optimized independently. This aggravation leads to hand-crafted feature representations designed by human experts based on heuristics and expertise.

Nevertheless, specific approaches have prevailed and established themselves as state-of-the-art techniques. Especially methods based on gradient histograms, such as *SIFT* [Low04] or *HOG* [DT05] are, besides neural networks (cf. Section 2.3.3) arguably, the most widely used classic feature representations for computer vision applications. SIFT is a method for extracting local image features consisting of an *keypoint* detector and feature *descriptors*. A descriptor is a numerical vector that *describes* the local image neighborhood around a keypoint. Therefore, keypoints are detected using a DoG scale space approach [Low04]. A Gaussian scale space is obtained by filtering the images using Gaussian kernels of increasing variance. This filtering simulates changing image details for an object captured at different distances [Rot19]. The adjacent scale spaces are subtracted to obtain the DoG scale space. These differences approximate a second-order derivative filter that extracts contour information at different frequency bands. Keypoints are extracted at local extrema while candidates directly located on edges are suppressed, as they are unlikely to be detected across image transformations. Additionally, gradient orientations and magnitudes are computed with respect to the scales of candidates. The gradient orientations are quantized in a histogram of 36 bins while the magnitudes are accumulated in their corresponding bin. Keypoints are extracted at all candidate locations for all orientations if their histogram value is within 80% of the maximum histogram value. Once the keypoints are detected, a descriptor consisting of 16 histograms is computed. discriptors are computed consisting of 16 histograms each of 8 orientation bins. These histograms contain 8 orientation bins and are extracted from a grid layout of 4×4 cells. Finally, the 16 histograms with

8 orientation bins are concatenated into a 128 dimensional vector. Computing gradients according to the scales of keypoints increases the property of scale invariance. The gradient orientations are rotated with respect to the keypoint orientations, leading to the rotation invariant property. The descriptors are first normalized, clipped at 0.2, and subsequently normalized again. These steps increase the invariance against non-linear contrast changes.

The SIFT method, as proposed in [Low04], heavily relies on a strong and robust detection of keypoints. This approach of detecting and describing points of interest has shown remarkable performance, especially for content-based image retrieval [ZYT18; OGL19]. However, the work from Dalal *et al.* [DT05] argues that the keypoint-driven approaches cause distinctively higher false-positive rates compared to the dense grid approach presented. The authors attribute this to the insufficiency of keypoint detectors in reliably detecting human body structures. HOG descriptors are similar to the SIFT descriptors because they result from a local neighborhood of gradient values. These gradient values are extracted from 16×16 pixel blocks of four 8×8 pixel cells. The significant difference to SIFT is the spatial positions at which descriptors are extracted. In contrast to keypoint detection, the positions of HOG descriptors are distributed uniformly in a regular grid with overlapping blocks. According to the results reported in [DT05], including overlapping regions significantly improves the performance. The gradient orientations within each cell are quantized and accumulated into orientation histograms containing 9 bins evenly spaced over 0° to 180°. Finally, the 4×4 histograms obtained from the 8×8 pixel cells are normalized by applying the L_1- or L_2-norm[1].

Another powerful image representation counts the occurrences of quantized feature descriptors. The approach is inspired by the *BoW* method for representing textual documents of variable length in the same vector space. Because of the image features used in the visual domain, this approach is named *Bag-of-Features* (BoF) [OD11]. First applied to image retrieval [SZ03], the BoF representation has shown remarkable results for image classification as well [OD11]. Principally, all types of image descriptors can be used to obtain a BoF histogram. However, improved results are achieved using SIFT descriptors extracted from a regular grid [KZ11; OD11]. These image descriptors are subsequently quantized. A standard method for quantizing numerical representations is Lloyd's algorithm [Llo82]. Clustering these descriptors is an essential step for obtaining typical representations. The number of these representations is an adjustable parameter, and the resulting vectors are called *visual words*. The set of visual words builds the *visual vocabulary* (or *codebook*), which can be seen as a bag of (quantized) features. A BoF representation is obtained by first computing the local feature descriptors and applying the clustering algorithm to assign each feature descriptor to its nearest visual word. The final histogram is computed by counting the occurrences of visual words. The resulting representations are orderless collections of quantized descriptors lacking any structure of spatial information [OD11]. Retaining this spatial information can benefit the representational power and improve the classification performance [LSP06]. In this regard, a prominent approach is dividing

1 The authors present results experimenting with the four normalizations L_1-norm, L_1-sqrt, L_2-norm and the L_2-hys as presented in the SIFT method [Low04].

a given image into multiple splits according to a pyramidal scheme referred to as *spatial pyramid* [LSP06]. Another approach is to split the image along the horizontal direction and extract BoF histograms by counting along the vertical direction. This splitting approach results in sequences of BoF histograms, which have been successfully used in combination with sequence models such as *hidden Markov models (HMMs)* [RVF12].

2.1.2 *Classification*

Given the numerical image representation, the classification task is to assign one or multiple classes from a predefined set of classes. In the pattern recognition field, a vast diversity of classifier types exist. The main difference between them is the mechanism of the *classification* or *decision rule* [DHS01, p.4]. These rules can be based on statistical models yielding *class probabilities*, *discriminating boundaries* such as hyper-planes in high dimensional vector spaces or the closest class representative determined through a *distance measure*. In this regard, very prominent classifier models are the *GMMs* [Mur12, p.339] for statistical classification, *SVMs* [Bis09, p.326] using hyper-planes to make class decisions and the kNN [DHS01, p.174] classifying samples according to their distance to prototype vectors representing a particular class. A GMM is a mixture model of k different normal distributions, and its probability density function (PDF) is defined as a weighted sum of these mixture components [Bis09, p.430]. The parameters of GMMs are usually estimated using the *EM* algorithm [DLR77], although gradient-based approaches are possible as well [GP21]. The EM algorithm is an iterative approach for *MLE*, altering between the two steps *expectation* and *maximization* until the difference of change between the parameter updates becomes very small. The *expectation* step computes the likelihoods of generating the feature vectors given the *mean* and *covariance* matrices of the mixture components. In the subsequent *maximization* step, these parameters (i.e., means and covariances) are updated to increase the likelihood of generating the feature vectors. In the cases of *binary* or *multi-class* classification, the class yielding the maximum posterior probability is assigned to the feature vector of the test sample. If multiple labels have to be assigned, known as *multi-label* classification, a threshold can be applied to estimate whether a label is present or not. The SVM classifier attempts to find a decision boundary in the feature space to separate the feature vectors of two distinct classes, making it a binary classifier in the first place [CV95]. Usually, multiple decision boundaries exist that separate both classes ideally. However, finding the boundary that leads to the lowest generalization error is preferred. A support vector machine approaches this problem through the concept of a *margin*, which is defined as the perpendicular distance between the decision boundary and the closest feature vectors [Bis09, p.326]. Maximizing this margin leads to a particular decision boundary, whose location is determined by a subset of feature vectors referred to as *support vectors* [Bis09, p.327]. As the support vector machine is fundamentally a binary classifier, multiple classes can be distinguished using the concepts of *one-versus-all* and *one-versus-one*. In the first concept, an SVM is trained to separate *one* target class

from *all* other classes, leading to K support vector machines where K is the number of classes. For the other concept, SVMs are trained to distinguish between *pairs* of classes. This approach requires $(K-1)^2$ support vector machines, considerably increasing the training effort. The k-nearest-neighbor classifier is based on a simple yet powerful concept. A feature vector from the test set is classified according to k vectors from the training set with the lowest distance. As the classes of each training sample are known, classification decisions are made following a majority voting [DHS01, p.174]. In order to avoid ties, an odd number of neighbors is recommended. The most prominent measure for determining the neighbors are the Euclidean distance and Cosine distance, although any other measure quantifying vector distances can be applied [DHS01, p.187].

2.2 FEEDFORWARD NEURAL NETWORKS

Feedforward neural networks are an essential class of neural networks. The prefix *feedforward* results from the computational information flowing from the model input through an intermediate representation to the output, without any *feedback* connections [GBC16, p.167]. The term *neural networks* has its origins in attempts to find mathematical representations of information processing in biological systems and has been used very broadly to cover a wide range of different models [Bis09, p.226]. However, the functioning of neural networks has nothing in common with biological systems. Instead, feedforward neural networks are regression models stacked in a hierarchical order to build a directed and acyclic graph. These models have remarkably high representational power, which is based on nonlinear functions applied after each regression step and a highly efficient gradient-based training procedure. This section explains the structure and function of neural networks as they are used in pattern recognition.

2.2.1 *Perceptron*

The *Perceptron* model was initially proposed by Rosenblatt [Ros58] and corresponds to a *binary classifier*. This model is based on a linear transformation where the multidimensional input $\mathbf{x}$ is mapped to either of two classes $y \in \{-1, 1\}$. Initially, this model was designed to operate as a machine that processed binary values and was later introduced as a model operating on real-valued numbers [MP69]. The structural overview of a simple Perceptron is visualized in Figure 2. The left side shows the distribution of two classes (blue and orange), and the Perceptron is displayed in the center of the figure. The separation of the two classes according to the decision boundary of the Perceptron is depicted on the right side. In order to obtain a class decision, the input $\mathbf{x} \in \mathbb{R}^d$ is *weighted* by the vector $\mathbf{w} \in \mathbb{R}^d$ and *biased* by the scalar $b \in \mathbb{R}^1$. Then, the resulting value is mapped to

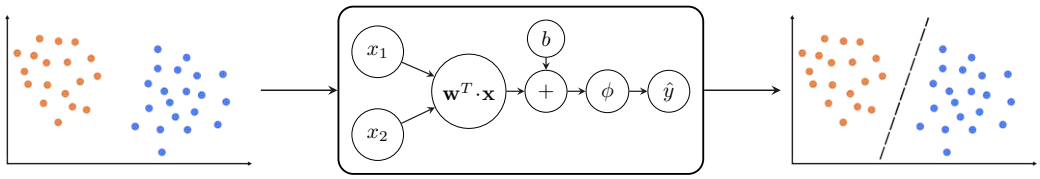


Figure 2: Visualization of the Perceptron model with a 2 dimensional input $\mathbf{x}$, bias value b and the non-linear function ϕ.

a class by applying a non-linear function $\phi(\cdot)$. Mathematically, the Perceptron model can be formulated as a non-linear model in the following way:

$$\hat{y}\left(\mathbf{x}|\mathbf{w}, b\right) = \phi\left(\mathbf{w}^T\,\mathbf{x} + b\right) \tag{1}$$

It is possible to simplify this model by specifying the bias value b as an additional input variable x_b with a constant value of 1. Consequently, the contribution of b can now be modeled by multiplying the new constant value x_b with an additional new weight value w_b. As a result, the formulation from Equation 1 simplifies to

$$\hat{y}\left(\mathbf{x}|\mathbf{w}\right) = \phi\left(\mathbf{w}^T\,\mathbf{x}\right) \tag{2}$$

where the dimensionality of $\mathbf{x} \in \mathbb{R}^{d+1}$ and $\mathbf{w} \in \mathbb{R}^{d+1}$, both increase by one. In the original formulation from [Ros58], the non-linearity $\phi(\cdot)$ is modeled through a step function defined as

$$\phi\left(\mathbf{x}|\mathbf{w}\right) = \begin{cases} 1 & \text{if } \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & otherwise \end{cases} \tag{3}$$

The Perceptron model *learns* to separate a given input $\mathbf{x}$ by adjusting the values for *weights* vector $\mathbf{w}$ according to a previously defined *criterion*. To *train* the model, Rosenblatt proposes an iterative optimization algorithm based on a per-sample classification error, called *Perceptron criterion* [Bis09, p. 193]. In each iteration, a training sample is processed individually by the Perceptron. A sample is correctly classified if $\hat{y} \cdot y = 1$, whereas $\hat{y} \cdot y = -1$ indicates a misclassification. The obtained class $\hat{y}$ corresponds to the desired label y if the criterion

$$\hat{y}\left(\mathbf{x}|\mathbf{w}\right) \cdot y > 0 \tag{4}$$

is satisfied. As a result, the values of $\mathbf{w}$ are not changed. Consequently, a missclassification leads to an update of $\mathbf{w}^n$ at iteration n according the rule:

$$\mathbf{w}^n \leftarrow \mathbf{w}^{n-1} + y \cdot \mathbf{x}\ . \tag{5}$$

The Perceptron criterion can be considered an early precursor of the *Gradient Descent* algorithm presented in Section 2.2.3.2. For linearly separable training samples, this algorithm is guaranteed to converge towards an optimal solution [MP69]. Although a simple Perceptron can only perform binary classification, the model can be extended to perform

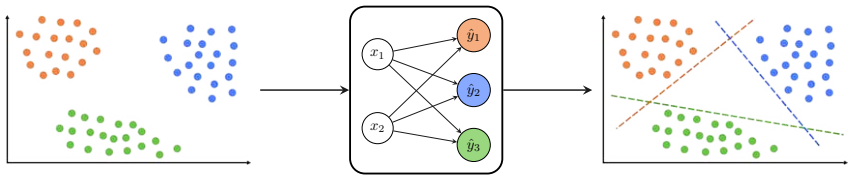


Figure 3: Visualization of the Single-Layer Perceptron, mapping the 2 dimensional input $\mathbf{x}$ to one of the 3 possible outputs $\hat{y}_1$, $\hat{y}_2$ or $\hat{y}_3$.

multi-class classification. This model is enhanced by combining multiple Perceptrons to obtain a more extensive model where multi-class classification is performed in a *one-vs-one* or *one-vs-all* fashion. The example in Figure 3 visualizes a three-class problem. Three Perceptrons displayed in the center classify the three classes (blue, green, and orange) on the left side. The corresponding class decision boundaries are shown on the right side. As the combination of multiple Perceptrons only contains a single *layer* of trainable parameters (i.e., $\mathbf{w}$), in this thesis, this model is referred to as *SLP*. A Perceptron with a single layer can only distinguish between linearly separable class distributions. Marvin Minsky and Seymour Papert criticized this limitation in [MP69]. The authors demonstrated the incapability of a Perceptron model to represent the *XOR-function* and presented the solution to this limitation in the same publication [MP69]. Minsky and Papert extended the model by concatenating multiple (Single-Layer) Perceptrons into a model known as *Multi-Layer Perceptron (MLP)* discussed in the following section.

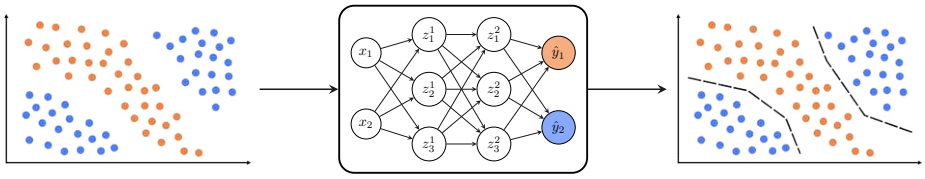


Figure 4: Visualization of the Multi-Layer Perceptron classifying the 2 dimensional input $\mathbf{x}$ into one of the two classes y_1 or y_2.

2.2.2 *Multi-Layer Perceptron*

The MLP consists of multiple sequentially concatenated *FC* Perceptron layers $\mathsf{L} = \{1, ..., \mathsf{L}\}$. Each previous layer $f^{(l-1)}$ computes an intermediate representation which, in turn, serves as an input to a subsequent layer $f^{(l)}$ [Bis09, p. 228]. The model consists of an *input* layer that forwards the input $\mathbf{x}$, multiple *hidden* layers each computing an intermediate representation $\mathbf{z}^{(1)}$, and the *output* layer mapping the last internal representation $\mathbf{z}^{(\mathsf{L})}$ to the desired output $\hat{\mathbf{y}}$. The simplified structure of the Multi-Layer Perceptron is visualized in the center of Figure 4. The toy example in Figure 4 demonstrates how two class distributions disposed in two different areas, shown on the left side, are separated by the Multi-Layer Perceptron using two decision boundaries as displayed on the right side. Transforming the input $\mathbf{x}$ to an intermediate representation $\mathbf{z}$ and subsequently to the output $\hat{\mathbf{y}}$ increases

the representational power of the Multi-Layer Perceptron model [Bis09, p. 230]. It is important to note that a trained model can lead to different boundaries than visualized in the toy example of Figure 4.

Modern Multi-Layer Perceptron models typically consist of two hidden layers, each containing a large number of neurons (i.e., up to 4096), which lead to high dimensional latent representations [GBC16, p. 196]. Mathematically, this model can be expressed analogous to the Perceptron model from Equation 2. A single layer of the MLP model is formally defined as

$$f^{(l)}\left(\mathbf{W}^{(l)}, \mathbf{z}^{(l-1)}\right) = \phi\left(\mathbf{W}^{(l)}\,\mathbf{z}^{(l-1)}\right) \ . \tag{6}$$

The Perceptron $f^{(l)}(\cdot)$ in layer l computes the dot product between the layer parameters $\mathbf{W}^{(l)}$ and the intermediate representation $\mathbf{z}^{(l-1)}$, obtained from the previous layer, followed by the non-linear transformation $\phi(\cdot)$. Consequently, the Multi-Layer Perceptron model can be formally described in the following way:

$$\hat{\mathbf{y}}_{\mathbf{W}}\left(\mathbf{x}\right) \ = \ f^{(\mathsf{L})}\left(\mathbf{W}^{(\mathsf{L})}, f^{(\mathsf{L}-1)}\left(\mathbf{W}^{(\mathsf{L}-1)}, \ldots f^{(2)}\left(\mathbf{W}^{(2)}, f^{(1)}\left(\mathbf{W}^{(1)}, \mathbf{x}\right)\right)\right)\right) \ . \tag{7}$$

The model parameters are represented by the set $\mathbf{W} = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(\mathsf{L})}\}$ where a set item $\mathbf{W}^{(l)}$ denotes a weight matrix in layer l. An essential component of a Multi-Layer Perceptron is the non-linearity $\phi(\cdot)$ referred to as *activation function* in the context of neural networks. The activation function must be non-linear. Otherwise, a single layer can represent the multiple layers, and the MLP boils down to a linear classifier [DHS01, p.286]. In this regard, an interesting theoretical property of the MLP named *universal approximation* has been proven by Hornik, Stinchcombe, and White [HSW89]. A model consisting of an input, output, and one hidden layer is able to approximate any function with arbitrary accuracy given a sufficient number of neurons. However, to unleash the enormous expressive power of the MLP a set of optimal model parameters $\hat{\mathbf{W}}$ is required. The process of finding optimal parameters is called *training* and is discussed in the next section.

2.2.3 *Training*

Neural networks are trained by adjusting the parameters (i.e., *weights* and *biases*) such that the overall discrepancy between the actual estimates of the network and the desired predictions defined by the labels is minimized. The differences between the network outputs and labels are quantified by so-called *loss functions* (Section 2.2.3.1). Model parameters are adjusted (or updated) following a *gradient-based* approach (Section 2.2.3.2). To compute the gradients, *differentiable* loss functions are required. The gradient intensities are influenced by the *error* that is *propagated backward* through the network layers (Section 2.2.3.3). This section explains the training procedure of neural networks, using the important concept of gradient-based error propagation to obtain the optimal model parameters.

2.2.3.1 *Loss Function*

The loss function quantizes the error between a neural network's prediction $\hat{\mathbf{y}}$ and the desired output $\mathbf{y}$. Typically, this quantization evaluates to a scalar value such that $\mathcal{L}\left(\hat{\mathbf{y}}_{\hat{\mathbf{W}}}, \mathbf{y}\right) \in \mathbb{R}^1$. In the context of neural networks, the resulting scalar is known as the *loss* of a neural network. Given a set of training samples $\mathbf{S} = \{(\mathbf{x}_1, \mathbf{y}_1), ...(\mathbf{x}_\mathsf{S}, \mathbf{y}_\mathsf{S})\}$ containing S pairs of input $\mathbf{x}$ vectors and their corresponding labels $\mathbf{y}$, the *total loss* of a model $\hat{\mathbf{y}}_{\hat{\mathbf{W}}}(\cdot)$ with parameters $\hat{\mathbf{W}}$ is assessed by calculating the average loss over all training samples:

$$\mathcal{L}_{total}\left(\hat{\mathbf{W}}, \mathbf{S}\right) \;=\; \frac{1}{|\mathbf{S}|}\sum_{s=1}^{\mathsf{S}} \mathcal{L}\left(\hat{\mathbf{y}}_{\hat{\mathbf{W}}}(\mathbf{x}_s),\, \mathbf{y}_s\right) \;. \tag{8}$$

Typically, in real-world applications, the model output $\hat{y}$ and the label y are represented by multi-dimensional vectors. Therefore, using a distance metric to evaluate the difference between the actual and the desired output is evident. Consequently, an often used loss function is the *Euclidean* loss defined as:

$$\mathcal{L}_{euc}\left(\hat{\mathbf{y}}, \mathbf{y}\right) = \sum_{d=1}^{D} \frac{1}{2}\left(\hat{y}_d - y_d\right)^2 \;. \tag{9}$$

This loss function computes the sum of squared differences between $\hat{\mathbf{y}}$ and $\mathbf{y}$, scaled by the factor $\frac{1}{2}$. The Euclidean loss fits well for *regression* tasks where labels are given as real-valued numbers. However, for *classification* tasks, class information is often represented following the *one-hot* encoding scheme (cf. Section 2.1.2). Although it is possible to predict one-hot encoded classes using the Euclidean loss, a more sophisticated loss function exists for this type of task. If labels are represented as K dimensional vectors where only 1 out of K elements is set to one while the others remain zero, the well-established *categorical cross-entropy loss* or simply *CE* is used to calculate the loss. The loss function is based on the entropy in information theory introduced by Claude Elwood Shannon [Sha48] and is defined as:

$$\mathcal{L}_{ce}\left(\hat{\mathbf{y}}, \mathbf{y}\right) \;=\; -\sum_{k=1}^{\mathsf{K}} y_k \cdot \log(p(\hat{y}_k)) \;. \tag{10}$$

The cross-entropy loss computes the negative logarithm of the class likelihood $\hat{p}(y_k)$, thus, sometimes referred to as *NLL* loss. The essential component for obtaining class likelihoods is the activation or *output* function used in the network's last layer. This function computes a *soft maximum* by normalizing the output $\hat{\mathbf{y}}$ in the following way:

$$p\left(\hat{y}_k\right) = \frac{e^{y_k}}{\sum_{i=1}^{\mathsf{K}} e^{y_i}} \quad \forall\; k \in \{1, ..., \mathsf{K}\} \tag{11}$$

and is, therefore, called the *softmax* function. The normalization applied in the softmax function restricts each $p(\hat{y}_k)$ to the interval $(0, 1)$ where class likelihoods close to zero have a high cross-entropy value and likelihoods close to one have a low cross-entropy value.

Loss functions and their respective non-linear outputs do generally not appear out of the blue. Instead, they are based on mathematical foundations. An elegant approach to derive the appropriate loss function for a given label distribution is provided by the statistical framework named GLM [NW72]. This framework is discussed in detail in the methodology of this thesis (Chap. 6). In short words, considering the Multi-Layer Perceptron as a statistical model, it is possible to derive the Euclidean loss by assuming that the labels of the training samples follow a Normal distribution. The same applies to the cross-entropy loss combined with the softmax function. Here, the assumption is that the labels follow a Multinomial distribution, sometimes referred to as *Categorical* distribution. For the interested reader further literature [Bis09; DHS01; Mur12] and the tutorial for *Minimizing the Negative Log-Likelihood* of Will Wolf[2] is referred.

2.2.3.2 *Gradient Descent*

Once the loss function has been defined, the total loss, as shown in Equation 8 of the neural network can be assessed. The average minimum loss value for all training samples is obtained by finding optimal parameter values $\hat{\mathbf{W}}_{opt}$ such that

$$\hat{\mathbf{W}}_{opt} \leftarrow \underset{\hat{\mathbf{W}}}{\operatorname{argmin}} \, \mathcal{L}_{total}\left(\hat{\mathbf{W}}, \mathbf{S}\right) . \tag{12}$$

However, finding an analytic solution to this problem is impossible, especially for neural networks consisting of multiple layers. For this reason, the training of multi-layer neural networks follows an iterative approach of weight adjustment referred to as *gradient descent* optimization algorithm. This algorithm iteratively applies small changes to the parameters $\hat{\mathbf{W}}$ in the direction of the steepest descent. The value of weight $w_{i,j}^{(l)}$ in layer l is changed for every iteration n by adding a fraction of the negative gradient of the loss $\mathcal{L}$ with respect to the weight itself. Formally, the *gradient descent* is defined as

$$w_{i,j}^{(l)}(n) \leftarrow w_{i,j}^{(l)}(n) - \eta \frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}(n)} . \tag{13}$$

Applying the steepest descent algorithm is guaranteed to converge to a local minimum and for convex loss functions even to the global minimum [Cur44]. The meta-parameter η is called *learning rate* and is usually set to $0 < \eta < 1$ in a practical training scenario. If the learning rate is too large, the optimization process might diverge, while a small learning rate might lead to a slowdown of the training. The learning progress is stabilized by introducing an inertia term to Equation 13 based on the momentum method [Pol64]. This method is a technique for accelerating gradient descent by accumulating a *velocity* v

2 https://willwolf.io/2017/05/18/minimizing_the_negative_log_likelihood_in_english/

in the direction of the gradient scaled by the *momentum* α [Sut+13]. The velocity $v_{i,j}^{(l)}$ for weight $w_{i,j}^{(l)}$ in iteration n is calculated as follows:

$$v_{i,j}^{(l)}(n) \;=\; \eta \frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}(n)} \;+\; \alpha \cdot v_{i,j}^{(l)}(n-1) \;. \tag{14}$$

Typically, the momentum is set to $0 < \alpha < 1$ and "throttles" the velocity. The accumulation results from the gradient in iteration n and the scaled fraction of the velocity from the previous iteration $n-1$. The improved version of gradient descent is defined as

$$w_{i,j}^{(l)}(n) \;\leftarrow\; w_{i,j}^{(l)}(n) - v_{i,j}^{(l)}(n) \;. \tag{15}$$

A neural network is trained iteratively by adjusting the parameters $\hat{\mathbf{W}}$ according to the update rule in Equation 15. This optimization algorithm was initially proposed to compute the gradients for each sample individually and, subsequently, update the network parameters using the average of all individual gradients [RHW86].

While this approach is feasible for a few training samples, it becomes computationally costly, especially for the sizes of modern datasets [BB07]. In order to reduce the computational costs, only a subset of the training samples is considered to compute the gradients within one update iteration [Bot91; Bot98]. This approach is widely known as *SGD*. In its extreme form, gradients are computed using only a single training sample referred to as *online learning*. However, nowadays, modern methods calculate the gradient using multiple training samples known as *mini batch* [KSH12; SZ15]. In recent years, several modified versions of the stochastic gradient descent were introduced, such as AdaGrad [DHS11] or AdaDelta [Zei12]. A prevalent approach called *Adam* [KB15] adaptively estimates the lower-order momentum and combines the benefits of RMSprop [TH+12] and SGD. The modifications share the common feature of adjusting the learning rate, momentum, or velocity based on the values from previous iterations, in contrast to the constant values used in the basic gradient descent optimization.

2.2.3.3 *Error Backpropagation*

The previous section's formulations are sufficient to train a model with only a single layer. However, calculating gradients for the weights of hidden layers is not as straightforward as presented in Equation 13. As the network computes the output through nested functions (cf. Equation 7), hidden layer gradients can be calculated by applying the *chain rule*. Following the chain rule, the gradient for weight $w_{1,j}^{(\mathsf{L})}$ in the last layer L (cf. Equation 6) can be written as

$$\frac{\partial \mathcal{L}_{euc}}{\partial w_{1,j}^{(\mathsf{L})}} \;=\; \frac{\partial \mathcal{L}_1}{\partial w_{1,j}^{(\mathsf{L})}} + \frac{\partial \mathcal{L}_2}{\partial w_{1,j}^{(\mathsf{L})}} \;=\; \frac{\partial \mathcal{L}_1}{\partial \phi_1^{(\mathsf{L})}} \cdot \frac{\partial \phi_1^{(\mathsf{L})}}{\partial z_1^{(\mathsf{L})}} \cdot \frac{\partial z_1^{(\mathsf{L})}}{\partial w_{1,j}^{(\mathsf{L})}} + 0 \tag{16}$$

where $\phi_1^{(\mathsf{L})}$ is the output value of the 1-th non-linearity and $z_1^{(\mathsf{L})}$ the result of the 1-th linear transformation $w_{1,j}^{(\mathsf{L})} \cdot x_j$. For the example[3] in Equation 16, suppose that $\mathcal{L}_{euc}$ is a two dimensional Euclidean loss (cf. Equation 9). Than, the weight $w_{1,j}^{(\mathsf{L})}$ only contribute to the first loss term $\mathcal{L}_1$ while the derivative for the second term $\mathcal{L}_2$ is zero. The gradient for a weight in the penultimate layer $w_{1,j}^{(\mathsf{L}-1)}$ is calculated similarly to Equation 16, with the major difference that this weight now contributes to both loss terms as shown in Equation 17.

$$\frac{\partial \mathcal{L}_{euc}}{\partial w_{1,j}^{(\mathsf{L}-1)}} = \left[\frac{\partial \mathcal{L}_1}{\partial \hat{y}_1^{(\mathsf{L})}} \cdot \frac{\partial \hat{y}_1^{(\mathsf{L})}}{\partial z_1^{(\mathsf{L})}} \cdot \frac{\partial z_1^{(\mathsf{L})}}{\partial w_{1,1}^{(\mathsf{L})}} + \frac{\partial \mathcal{L}_2}{\partial \hat{y}_2^{(\mathsf{L})}} \cdot \frac{\partial \hat{y}_2^{(\mathsf{L})}}{\partial z_2^{(\mathsf{L})}} \cdot \frac{\partial z_2^{(\mathsf{L})}}{\partial w_{2,1}^{(\mathsf{L})}} \right] \cdot \frac{\partial \phi_1^{(\mathsf{L}-1)}}{\partial z_1^{(\mathsf{L}-1)}} \cdot \frac{\partial z_1^{(\mathsf{L}-1)}}{\partial w_{1,j}^{(\mathsf{L}-1)}} \ . \tag{17}$$

In general, using the chain rule to compute the gradient of weight $w_{i,j}^{(l)}$ in layer l can be written as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w_{i,j}^{(l)}} = & \underbrace{\left[\sum_{n=1}^{D^{(L)}} \frac{\partial \mathcal{L}_n}{\partial \hat{y}_n^{(\mathsf{L})}} \cdot \frac{\partial \hat{y}_n^{(\mathsf{L})}}{\partial z_n^{(\mathsf{L})}} \cdot \sum_{m=1}^{D^{(\mathsf{L}-1)}} \frac{\partial z_n^{(\mathsf{L})}}{\partial \phi_m^{(\mathsf{L})}} \right]}_{\text{Layer } \mathsf{L}} \cdot \prod_{k=l+2}^{\mathsf{L}-1} \underbrace{\left[\sum_{n=1}^{D^{(k)}} \frac{\partial \phi_n^{(k)}}{\partial z_n^{(k)}} \cdot \sum_{m=1}^{D^{(k-1)}} \frac{\partial z_n^{(k)}}{\partial \phi_m^{(k-1)}} \right]}_{\text{Layer } \mathsf{L}-1 \text{ to } l+2} \\ & \cdot \underbrace{\left[\sum_{n=1}^{D^{(l+1)}} \frac{\partial \phi_n^{(l+1)}}{\partial z_n^{(l+1)}} \cdot \frac{\partial z_n^{(l+1)}}{\partial \phi_i^{(l)}} \right]}_{\text{Layer } l+1} \cdot \underbrace{\left[\frac{\partial \phi_i^{(l)}}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial w_{i,j}^{(l)}} \right]}_{\text{Layer } l} \ . \end{aligned} \tag{18}$$

The square brackets emphasize the gradient calculation for the corresponding layers. The dimensionality for layer l is denoted by $D^{(l)}$ while the indices n and m represent the current index of a neuron in the output and input of a layer. An important insight for applying the chain rule is that the weight $w_{i,j}^{(l)}$ in layer l only contributes to one output neuron, here $\phi_i^{(l)}$. Consequently, this neuron influences all output neurons in the ensuing layer $l+1$, computed as a sum of the gradients over $D^{(l+1)}$ elements. For all subsequent layers $\{l+2, ..., L\}$, all inputs using FC layers influence all output values. Instead of calculating individual gradients, a computational more efficient approach is to compute all gradients in layer l following a vectorized approach written as:

$$\frac{\partial \mathsf{L}}{\partial \mathbf{w}^{(l)}} = \frac{\partial \mathsf{L}}{\partial \mathbf{f}^{(\mathsf{L}-1)}} \cdot \prod_{k=l}^{\mathsf{L}-1} \frac{\partial \mathbf{f}^{(k)}}{\partial \mathbf{f}^{(k-1)}} \cdot \frac{\partial \mathbf{f}^{(l)}}{\partial \mathbf{w}^{(l)}} \ . \tag{19}$$

For the example in Equation 18, the gradient for a weight $w_{i,j}^{(\mathsf{l})}$ depends on the derivative of the loss function concerning the actual output of the network $\frac{\partial \mathcal{L}_i}{\partial \hat{y}_i^{(\mathsf{L})}}$. For example, if the network is trained using the Euclidean loss, then the derivative is defined as:

$$\mathcal{L}_{euc}(\hat{y}_i, y_i) = \frac{1}{2} \left(\hat{y}_i - y_i\right)^2 \ \Rightarrow \ \frac{\partial \mathcal{L}_{euc}}{\partial \hat{y}_i} = \hat{y}_i - y_i \ . \tag{20}$$

3 This example is adapted from Matt Mazur's tutorial *A Step by Step Backpropagation Example*: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

The difference between the actual network output $\hat{y}$ and the desired output y is directly influenced by the magnitude of the gradient. Wrong network predictions are penalized by larger gradients, leading to larger changes in the respective weights. Using the chain rule to calculate how the weights should change *propagates back* the *error* caused by the current parameters of a network. For this reason, the presented approach is commonly known as *error backpropagation* [RHW86] and is nowadays the standard method for gradient calculation. Another interesting insight from Equation 18 is that the input to the network should have values that are not zero. The reason is that the gradient of $z_i^{(l)}$ with respect to the weight $w_{i,j}^{(l)}$ in layer l is simply:

$$z_i^{(l)}(w_{i,j}^{(l)}, x_j) = w_{i,j}^{(l)} \cdot x_j \;\Rightarrow\; \frac{\partial z_i^{(l)}}{\partial w_{i,j}^{(l)}} = x_j \;. \tag{21}$$

Consequently, if x_j is zero, the gradient of $w_{i,j}^{(l)}$ will be zero as well, independent from the other factors in Equation 18. Furthermore, Equation 18 essentially demonstrates why the *step* function used for the initial Perceptron model [Ros58] is not applicable for the training of a multi-layered neural network. The derivative of the step function is zero for all input values except for the input value zero, for which the derivative is undefined. Choosing the step function as activation $\phi(\cdot)$ for the training of a neural network and applying the gradient descent method will result in zero-valued gradients, leading to stagnation in the weight updates. For this reason, a non-linearity known as the *sigmoid*[4] [Bis09, p.205] [GBC16, p.194] function is traditionally chosen as the activation in multi-layered networks. The Sigmoid function can be described as a *soft* step function. Formally, this activation and its derivative are defined as:

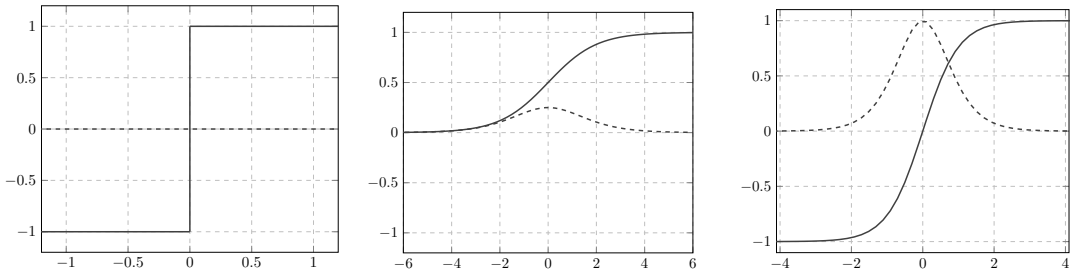
$$\sigma(x) = \frac{1}{1+e^{-x}} \;\Rightarrow\; \frac{\partial \sigma(x)}{\partial x} \;=\; \sigma(x) \cdot (1 - \sigma(x)) \;. \tag{22}$$

Another activation function used for multi-layered networks is the hyperbolic tangent [GBC16, p.194] that shares a similar characteristic as the sigmoid function. The hyperbolic tangent and its derivative is defined as:

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} \;\Rightarrow\; \frac{\partial tanh(x)}{\partial x} \;=\; 1 - tanh^2(x) \;. \tag{23}$$

Both functions transform the input value into a specific interval and act as a soft-step transition. However, the intervals of the output values differ as the sigmoid limits its output to values between 0 and 1, while the hyperbolic tangent transforms its output to a range from -1 to 1. A comparison between the three activation functions step, sigmoid, and hyperbolic tangent is visualized in Figure 5. Considering the derivatives, the sigmoid (b) and hyperbolic tangent (c) always have a gradient larger than zero in contrast to the step function (a), showing a constant gradient of zero.

4 The sigmoid function is also known as *logistic* function (cf. Section 6.2.2) [Agr15]



(a) step (b) sigmoid (c) hyperbolic tangent

Figure 5: Visualization of the step (a), sigmoid (b) and hyperbolic tangent (c) functions.

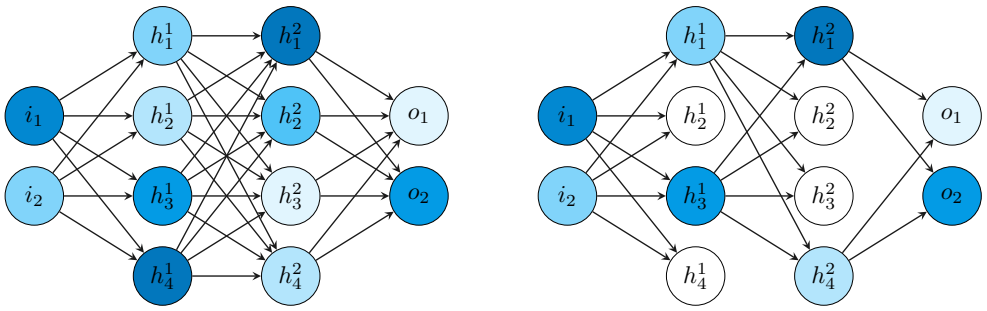
2.2.4 *Generalization and Regularization*

The central challenge of pattern recognition is to design a model that not only predicts well on the previously seen training data but instead shows satisfying performance on *previously unseen* test data. The ability to perform well on unobserved input data is referred to as *generalization* [GBC16, p.110]. Therefore, machine learning models are optimized using training data to reduce the *training error*. Subsequently, the generalization capabilities are evaluated on some previously unseen test data by computing the *test error*, also called *generalization error*. The test data is the target domain of the present task to be solved (i.e., learned). A machine learning model typically generalizes well if the training and test data are *identically distributed* and the model *capacity* is large enough to represent both data sets [GBC16, p.111]. For model training, the distribution of the training data is assumed to be representative of the distribution of the test data. While the impact on the data set is vanishingly small, except for the option to collect more data, the model capacity can be significantly influenced. The capacity of a neural network is defined by the number of parameters, which can be increased by adding more layers or increasing the number of neurons per layer. However, simply increasing the capacity of a neural network leads to an effect known as *overfitting*, where the training error continuously declines while the test error begins to increase. A model tends to overfit if the capacity is high enough to memorize the training samples. This undesirable effect can be alleviated by choosing a model with lower capacity. However, a model with low capacity may not obtain sufficiently low error values on the training set [GBC16, p.111]. This effect is referred to as *underfitting*. In practice, it is difficult to find the optimal capacity by adjusting the model size and, therefore, neural networks are typically designed to be *over-parametrized* [DL18; Ney+18; Ney+19]. In order to avoid overfitting, different *regularization* techniques are applied to control high model capacities. The prevalent approaches for regularization are based on penalizing larger weight values referred to as L_2 *regularization* [Nie15, p.79], randomly suppressing neuron outputs known as *Dropout* [Hin+12] and artificially expending the training set through *data* or *image augmentation*.

The idea of L_2 regularization is to add an L_2 normalizing term to the loss function that penalizes large weights and instead favors lower ones. Mathematically, weight decay can be formalized as follows

$$\mathcal{L}_\lambda(\mathbf{y}, \hat{\mathbf{y}}) = \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) \; + \; \lambda \|\mathbf{W}\|_2^2 \; , \tag{24}$$

where $\mathcal{L}_\lambda$ denotes the regularized loss function and $\mathbf{W}$ all network parameters. The factor λ is a hyperparameter that needs to be set in advance and defines how strong the weight decay contributes to the loss. A higher λ value results in a model with smaller weight values, while a small λ value softens the restrictions. In the context of neural networks, L_2 regularization is also known as *weight decay* and in other communities, often referred to as *ridge regression* or *Tikhonov* regularization [GBC16, p.231]. An alternative to weight decay is the L_1 regularization, which computes a sum of absolute values. The major difference between the L_1 and L_2 regularization is the effect on the model parameters. The gradient of the L_2 regularization with respect to a single weight is $2\lambda w$, which means that high weight values are penalize stronger than lower ones. On the other hand, the gradient of the L_1 term is the constant λ penalizing all weight values by the same intensity, leading to sparse representations [GBC16, p. 235].



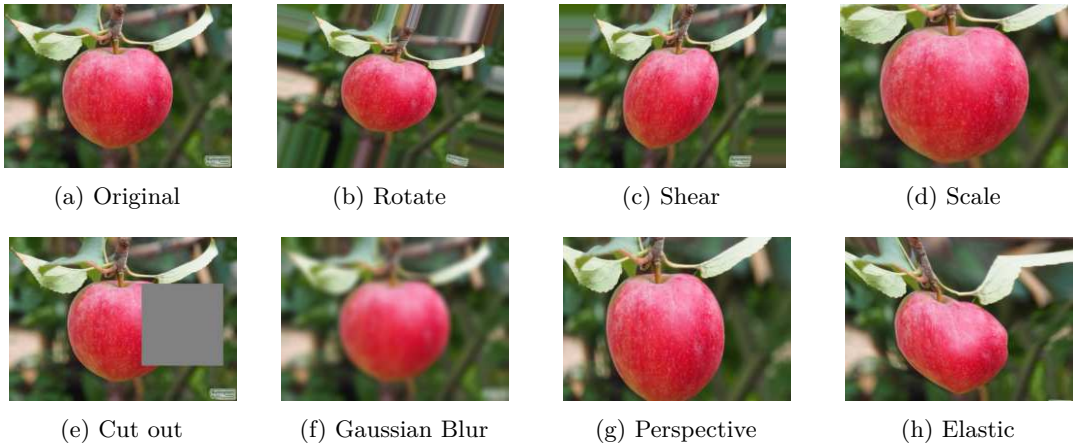
(a) Without dropout (b) With dropout

Figure 6: Visualization of a MLP if no Dropout is applied (a) and the effect of Dropout (b).

The regularization technique Dropout [Hin+12] is inspired by the findings that classifier ensembles as good as always improve the performance of machine learning models [Sri+14]. Approaches such as *bagging* [Bre96] proved that training an ensemble of individual models leads to higher performance and significantly decreases the generalization error [War+14]. However, training multiple large neural networks using subsets of data is computationally very expensive. Dropout addresses this issue and provides an efficient way of combining exponentially many different neural networks while at the time preventing overfitting [Sri+14]. This technique is applied to one or multiple hidden layers where individual neurons are "dropped out" with a certain probability. A neuron is dropped out by multiplying its output with zero. In the case of fully connected layers, Dropout is realized by sampling a vector of binary values (i.e., 0 or 1) from a Bernoulli distributed random variable and multiplying the output of a hidden layer by the binary vector. Figure 6 shows a visual example of Dropout being applied to a MLP. The left side depicts the MLP with-

out Dropout, where all neurons are active, represented by the fully connected arrows and color intensities. On the other hand, the right side of Figure 6 visualize the same MLP where some of the hidden neurons are "dropped out" emphasized by blank circles with no connections to the ensuing layer. Each neuron is retained during training with a fixed probability of p, for example, 0.5, independent of other neurons. A single neural network is used without Dropout at test time to explicitly average the predictions from exponentially many models. Therefore, the output of a neuron is scaled by p because the model weights adapt to this fraction of active neurons while being trained [Sri+14].

Larger datasets and more data diversity are the best way to prevent overfitting and improve generalization [GBC16, p.240]. Driven by this insight, data augmentation aims to increase the amount of data by altering the original samples while preserving the semantic content. While not a traditional regularization technique, data augmentation addresses overfitting and generalization issues stemming from a lack of training data [SK19]. Although augmentation techniques are not limited to images, only augmentations on image data are discussed in the following. Images are typically augmented by applying simple affine transformations like rotation, shearing, or scaling and more complex projections like perspective or elastic transformations. In classic approaches, an expert typically selects image transformations for the task at hand. Not all augmentations are applicable to specific domains, as this transformation might change the semantic content of the image. A prime example is mirroring techniques like horizontal or vertical flipping applied to handwritten or machine-printed text. Horizontally flipping the character b might lead to a visual appearance close to the character d, while vertical flipping would result in the character p. Visual examples of the most frequently used augmentations are shown in Figure 7. The original image (a) is depicted on the far left side, and on the right side, seven results (b to h) of different transformations are visualized. In real-world training scenarios, image augmentations are typically sampled randomly from a set of transformations. The intensity of a particular augmentation is typically sampled from a predefined range as well. For example, the range of rotation angles can be set from 5 to 25 degree, and when the rotation transformation is applied, an angle is uniformly sampled from this range. Image augmentations can either be applied individually or as a group executed consecutively. However, applying more than three augmentations at once rarely leads to useful results [SK19]. There exist many more augmentation approaches and especially with the rise of deep neural networks, techniques like *Adversarial Training* [GSS15], *Generative Adversarial Networks* (GANs) [Goo+14] or *Neural Style Transfer* [GEB16] starts to dominate the area of image manipulation in recent years. Nonetheless, in the scope of this thesis, only the classical image augmentations, as shown in Figure 7, are used for regularization purposes. The survey on image data augmentation proposed by Shorten *et al.* [SK19] is referred to for the interested reader.



(a) Original (b) Rotate (c) Shear (d) Scale

(e) Cut out (f) Gaussian Blur (g) Perspective (h) Elastic

Figure 7: Examples of the resulting augmentations (b to h) obtained from the original image (a).

2.3 CONVOLUTIONAL NEURAL NETWORKS

The neural network models based on the Multi-Layer Perceptron are powerful classifiers. They are capable of mapping arbitrarily input vectors to their corresponding class encodings. However, despite their property of being a universal approximator, the fully connected nature poses a crucial handicap, especially for classification tasks containing grid-like data such as images [GBC16, p.330]. When processing high-dimensional data, the dense architecture of a Multi-Layer Perceptron is a disadvantage. Considering an image of 256×256 pixels and an MLP with one hidden layer of size 100, a large amount of $256^2 \times 100 = 6\,553\,600$ parameters is required to transform the input. Additionally, the *full connectivity* of the MLP cannot transfer knowledge between translated objects as every parameter only processes the same pixel position across all data samples. When dealing with grid-like data (i.e., audio or images), it is desired to obtain a model with three essential properties, namely *sparse interactions*, *parameter sharing* and *equivariant representations* [GBC16, p.335]. A model class specifically designed to cover all three properties is referred to as *convolutional neural networks CNNs*. Although a predecessor of these networks models called *Neocognitron* was initially proposed by Kunihiko Fukushima [Fuk80; FM82; FMI83], the convolutional neural networks gained popularity from the publication of Yann LeCun [LeC+89a; LeC+89b; Lec+98]. Interestingly, both authors demonstrated the model capabilities for handwriting digit recognition tasks. The essential components of CNNs are the *convolutional layers* that are eponymous for this type of neural network (Section 2.3.1). Typically, a convolutional neural network consists of multiple cascaded convolutional layers. Although these networks are not fully *scale invariant*, these models *tolarate* a certain degree of objects being scaled. This property is provided by another type of layer performing an operation widely known as *pooling* (Section 2.3.2). This operation computes a summary value from a local neighborhood of pixels [GBC16, p.330]. While the earlier models consisted of three convolutional layers followed by a Multi-Layer Perceptron [Lec+98], later versions strived towards higher numbers of cascaded convolutional layers [KSH12; SVZ14; He+16], which gave rise to the term *deep neural networks* (Section 2.3.3).

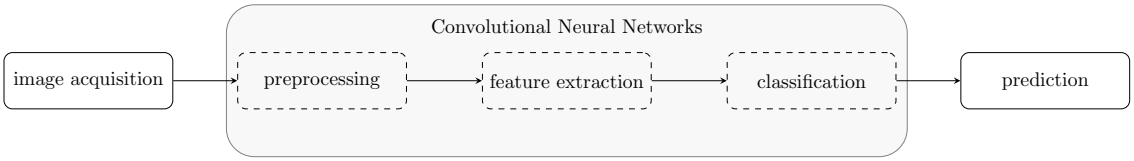


Figure 8: Visualization of a typical pattern recognition processing pipeline using a convolutional neural network (CNN).

As shown in Figure 8, the primary advantage of convolutional neural networks is the *joint optimization* of the *feature extraction* and *classification* stages, compared to classic pattern recognition pipelines (cf. Figure 1 in Section 2.1). In contrast to classic approaches, where both stages are optimized separately, CNNs are trained in an *end-to-end* fashion. This training procedure results in a model containing jointly tuned parameters for feature extraction and classification. Nowadays, the most performant approaches in classification tasks use convolutional neural networks.

2.3.1 *Convolutional Layers*

A convolutional layer is based on operations referred to as *convolutions* [Sze22, p.232]. This layer transforms a given input tensor $\mathbf{I}_{\mathsf{c},\mathsf{h},\mathsf{w}}$ with c number of channels, height h and width w by multiplying small patches of the input $\mathbf{I}_{\mathsf{c},\mathsf{y},\mathsf{x}}$ with corresponding *kernel weights* $\mathbf{K}_{\mathsf{c},\mathsf{y},\mathsf{x}}$. These weights are the *trainable parameters* of a convolutional neural network and represent the equivalent to the weights of a Multi-Layer Perceptron (cf. Section 2.2.2). The spatial size of a kernel defined by y and x corresponds to the spatial size of the input patch and is, typically, referred to as *receptive field* of a convolutional layer. This kernel size defines the number of locally neighboured positions being discretely convolved. In practical applications the kernel size is usually set to a receptive field of 3×3 or 5×5, and less common to 7×7 for individual layers [KSH12; SZ15; He+16]. Given the input $\mathbf{I}_{\mathsf{c},\mathsf{h},\mathsf{w}}$ and weight kernel $\mathbf{K}^{(k)}_{\mathsf{c},\mathsf{y},\mathsf{x}}$, the single convolutional operation at position h, w is defined as:

$$f^{(k)}(h,w) \;=\; \sum_{c=1}^{\mathsf{c}} \sum_{i=-\lfloor \frac{\mathsf{y}}{2} \rfloor}^{\lceil \frac{\mathsf{y}}{2} \rceil - 1} \sum_{j=-\lfloor \frac{\mathsf{x}}{2} \rfloor}^{\lceil \frac{\mathsf{x}}{2} \rceil - 1} \mathbf{K}^{(k)}_{c,i,j} \cdot \mathbf{I}_{c,h+i,w+j} \tag{25}$$

A full descrete convolution is performed by shifting the kernel $\mathbf{K}^{(k)}_{\mathsf{c},\mathsf{y},\mathsf{x}}$ along height h and width w of input Tensor $\mathbf{I}_{\mathsf{c},\mathsf{h},\mathsf{w}}$ and compute $f^{(k)}$ at every single position h, w. A kernel is shifted with a step size called *stride*, a hyperparameter that must be defined in advance. The result of a convolution is a two-dimensional matrix $F_{\mathsf{u}\times\mathsf{v}} \in \mathbb{R}^{\mathsf{u}\times\mathsf{v}}$ of height u and width v which in the context of convolutional neural networks is known as *feature map*. A single value in the feature map $F(u, v)$ at position u, v is obtained by computing $f(u, v)$ and applying a non-linearity ϕ to the result as shown in the following:

$$F(u,v) \;=\; \phi\left(f(u,v)\right) \tag{26}$$

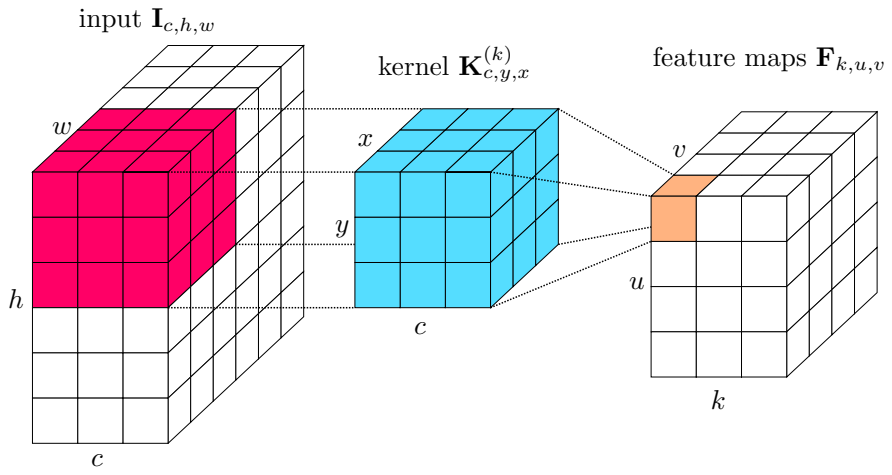
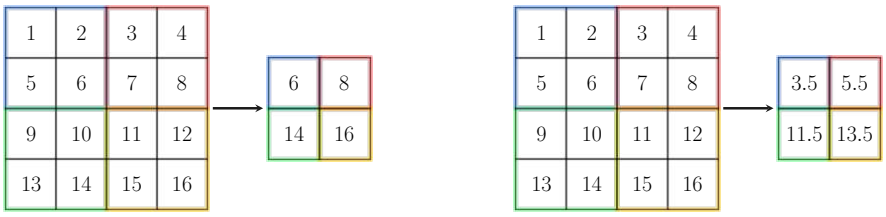


Figure 9: Conceptual visualization of a convolutional operation that takes the *input tensor* $\mathbf{I}$ and convolve it with the *filter kernel* $\mathbf{K}$. The resulting value is stored in the output tensor $\mathbf{F}$, also known as a convolutional layer's *feature maps*.

The result of $F(u, v)$ is equivalent to the output of one neuron of a Perceptron layer (cf. Section 2.2.2). A convolutional layer typically contains more than one kernel, resulting in multiple feature maps. These feature maps are concatenated to a tensor $\mathbf{F}_{\mathsf{k,u,v}} \in \mathbb{R}^{\mathsf{k}\times\mathsf{u}\times\mathsf{v}}$ which represents the output of a convolutional layer containing k feature maps, each of height u and width v. Figure 9 gives a visual impression of a convolution performed by a convolutional layer as described in Equation 25 and 26. The input $\mathbf{I}_{\mathsf{c,h,w}}$ is processed by convolving small input patches (i.e.$\mathbf{I}_{\mathsf{c,y,x}}$), using k kernels (i.e.$\mathbf{K}_{\mathsf{k,c,y,x}}$) which results in k feature maps represented by the tensor $\mathbf{F}_{\mathsf{k,u,v}}$.

2.3.2 *Pooling Layers*

Another important operation in modern convolutional neural networks is the so-called *pooling*, mainly serving two purposes. A pooling layer is integrated into a network's architecture to increase its robustness against translations and progressively reduce the dimensionality of feature maps in height and width across the layers. The pooling operation aggregates locally neighbored input values into a single scalar. Like the convolutional operations, pooling is performed by shifting a predefined kernel window according to the stride. However, pooling is applied channel-wise, meaning that the aggregation is performed for each channel individually without considering the values from other channels in contrast to the convolutions described in the previous section. A pooling layer may reduce the feature map's height and width while keeping the number of channels unchanged. The most commonly used operation is the *max pooling* [KSH12; SZ15; He+16] where the maximum value is computed at each window position. A less common aggregation strategy is the *average pooling* [Sze+16]. Here, the average value given by the window position is calculated. Figure 10 shows conceptually how a pooling layer transforms the 4×4 matrix input into the respective 2×2 sized output. Pooling is performed using a kernel size of 2×2 and a



(a) Maximum pooling (b) Average pooling

Figure 10: Example of the maximum (a) and average (b) pooling operations. The color boxes emphasis the input of a kernel window and its respective output.

stride of 2. The height and width of the resulting output matrix are reduced by the factor $\frac{1}{2}$ that is reciprocal to the stride value. The color boxes in Figure 10 indicate the position of a kernel window and its respective output value.

2.3.3 *Deep Neural Networks*

Deep neural networks are designed by cascading multiple convolutional layers and pooling layers. Usually, a Multi-Layer Perceptron is added to this architecture, serving as a classifier. Conventional architectures forward the input through the convolutional layers; afterward, the feature maps of the last convolutional layer are vectorized and mapped to a class by fully connected layers. The longstanding opinion is that the network's convolutional part extracts the features, while classification is performed by the MLP. However, the joint parameters optimization of the convolutional and fully connected layers blurs the seperation between the feature extraction and classification. A classifier can be obtained by directly mapping the convolutional features to their corresponding class encodings using a Single-Layer Perceptron [Zho+16b; He+16]. Furthermore, network architectures designed explicitly for *semantic segmentation* (i.e., pixel-wise classification) entirely omit FC layers and purely operate with convolutions [Min+22]. The pixel-wise classification is performed in the last convolutional layer where a class is represented by one corresponding channel of the feature map [RFB15; BKC16]. Similarly, class predictions are obtained using the output of the last convolutional layer in classification tasks as well [Spr+15]. The output of the last layer and the one-hot encoded class vector are bridged by computing average values per channel, which is referred to as *global average pooling* [LCY14].

An important feature of CNNs is the growth of the *effective receptive field* in the "deeper" layers. The effective receptive field refers to the window size in height and width covering all input values that can potentially influence a particular value in the feature map of a deeper layer. Figure 11 demonstrates the expansion of this field starting at the input on the left side and ending with one value in the feature map of layer 3 on the right side. This example is based on three convolutional layers, all having a kernel size of 3×3. The areas colored in green mark all potential influences on the (center) value in layer 3. In each layer, one convolutional operation using a 3×3 kernel is visualized with dashed gray lines, and a gray dot represents the resulting value. Starting in layer 3 and going backward, the

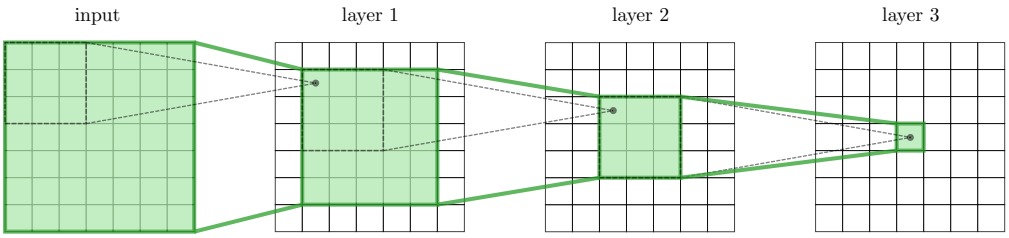


Figure 11: Conceptual visualization of the effective receptive field. One pixel in layer 3 is potentially influenced by 7×7 pixel in the input.

center value in layer 3 (green square) is influenced by a field of size 3×3 in layer 2 (green window) while this field, in turn, is influenced by a 5×5 sized field in layer 1, and last but not least, the area in layer 1 with 5×5 values are influenced by a window of size 7×7 in the input. The growth rate of the receptive field depends on the kernel size and the stride applied in each convolutional layer. However, explaining the different configurations for the growth rate of the receptive field is beyond this thesis, and for the interested reader, further literature is referred to [BKC16; YK16; Luo+16].

The kernels in the convolutional layers serve as feature detectors, producing higher activations for specific object patterns or structures. It has been shown that the layers close to the input tend to learn primitive patterns like blobs or edges [ZF14; SVZ14]. The subsequent layers build on the activations of the preceding layers and learn more complex patterns, such as parts of objects [Spr+15]. A convolutional neural network adapts the kernel parameters for deeper layers to learn the input data's hierarchical features. These hierarchical characteristics are generally provided by natural images where whole objects (ex., dog) are composed of object parts (ex., dog snout, tail, paws), which in turn can be composed of blobs and edges. The same principle applies to images of handwritten documents. Words can be broken down into individual characters, and those characters can be further broken down into pen strokes.

Forwarding the input through a convolutional neural network can be described by nested functions (cf. Equation 7), which has earned this type of model the prefix *deep* (instead of long or wide). However, the term deep does not specify the number of layers required for a neural network to be considered deep. While in [Ben09], deep architectures are defined as having many hidden layers, other works like [GBB11] consider networks as being deep if the architecture consists of more than two hidden layers. CNN architectures contain 5 to 9 [Lec+98; KSH12] or 13 to 19 [SZ15] layers with trainable parameters. Deeper neural networks contain between 50 and 150, or up to 1001, trainable layers [He+16]. According to [Ben09], long before the recent success of deep neural networks, architectures with more depth were assumed to learn more powerful representations, and increasing the number of convolutional layers was expected to gain performance for a broad range of pattern recognition tasks. Although the existing knowledge of deeper networks is crucial regarding recognition performance, training deep architectures proved to be difficult [GB10].

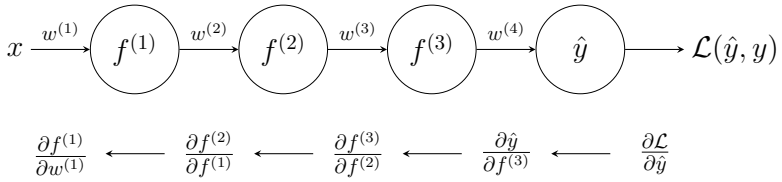


Figure 12: The structural value flow through the layers in the forward pass (top) and the back-propagated gradient during the backward pass (bottom).

2.3.4 *Vanishing Gradient Problem*

For a long time, the structure of the deep neural networks prevented training them [GB10; Ben09]. The gradient, being backpropagated through the network, shrinks after each layer, leading to a *vanishing gradient* with increasing depth of the architecture. A vanishing gradient was first described for *RNNs* [Hoc98] and occurred during the training of (forward) neural networks with a higher amount of layers as well. This structural problem was caused by choosing a less suitable activation function. This problem is demonstrated by the example visualized in Figure 12 adapted from [Nie15]. Suppose the architecture of a Multi-Layer Perceptron consisting of 4 hidden layers, each containing a single neuron connected to only one weight. Additionally, the sigmoid is used as the activation function at the layer's outputs. The forward pass is shown in the upper part of Figure 12, starting with the input x on the left and the resulting loss $\mathcal{L}(\hat{y}, y)$ on the far right. Underneath, the gradient is depicted, showing the partial derivatives of each layer starting on the right side and propagating the error backward to weight $w^{(1)}$ in the first layer. Every node except the last one is defined as $f^{(l)} = \sigma(z^{(l)})$ where $z^{(l)} = (w^{(l)} \cdot f^{(l-1)})$. To further simplify the example, the non-linearity is written as $\sigma(z^{(l)}) = \sigma^{(l)}$. For this example, the Euclidean loss is used that gives an output node $\hat{y}$ without any non-linearity so that $\hat{y} = w^{(4)} \cdot f^{(3)}$. The equation for the gradient of weight $w^{(1)}$ is than defined as:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \sigma^{(3)}} \cdot \frac{\partial \sigma^{(3)}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial \sigma^{(2)}} \cdot \frac{\partial \sigma^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial \sigma^{(1)}} \cdot \frac{\partial \sigma^{(1)}}{\partial z^{(1)}} \cdot \frac{\partial z^{(1)}}{\partial w^{(1)}} \,. \tag{27}$$

In the equation above, all terms are multiplied according to the chain rule and the partial derivative of the sigmoid function with respect to its input (i.e., $\frac{\partial \sigma^{(l)}}{\partial z^{(l)}}$) occurs three times. The first part of the structural problem is that the derivative of the sigmoid (cf. Equation 22) reaches its peak at 0.25 for an input of 0, as shown in Figure 5. This peak value implies that, in the best case, the gradient is scaled by $\frac{1}{4}$ for every node using the sigmoid activation function. For the example in Figure 12, the gradient of weight $w^{(1)}$ is scaled by $(\frac{1}{4})^3 = \frac{1}{64}$ in the best cast. While this scaling factor has little influence on the gradients of layers close to the output, the gradients of layers significantly further away from the output

almost vanish. A straight forward solution could be an upscaling by forcing $\frac{\partial z^{(l)}}{\partial \sigma^{(l-1)}} \geq 4$. This consideration is supported by the following derivative

$$z^{(l)} = (w^{(l)} \cdot \sigma(z^{(l-1)})) \quad \Rightarrow \quad \frac{\partial z^{(l)}}{\partial \sigma^{(l-1)}} = w^{(l)} \ . \tag{28}$$

So if the expression above has a weight value greater than or equal to 4, the following term should result in

$$\frac{\partial \sigma^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial \sigma^{(l-1)}} \geq 1 \ . \tag{29}$$

However, this consideration is more difficult to realize than it seems. The reason is that the derivative of the sigmoid itself depends on the weight $w^{(l)}$, which should have large values to meet the requirement of Equation 29. The following example will highlight this problem:

$$\frac{\partial \sigma^{(l)}}{\partial z^{(l)}} \cdot \frac{\partial z^{(l)}}{\partial \sigma^{(l-1)}} = \sigma(w^{(l)} \cdot \sigma^{(l-1)}) \cdot (1 - \sigma(w^{(l)} \cdot \sigma^{(l-1)})) \cdot w^{(l)} \ . \tag{30}$$

On the one hand, large values of the weight $w^{(l)}$ will upscale Equation 30, but saturate the sigmoid function (i.e.values close to 0 or 1) and result in tiny gradients (cf. Figure 5). On the other hand, setting $w^{(l)} = 0$ leads to the highest gradient of the sigmoid function (i.e.$\frac{1}{4}$), but the weight itself scales the expression in Equation 30 to zero. This contrast is the second part of the structural problem, causing a vanishing gradient.

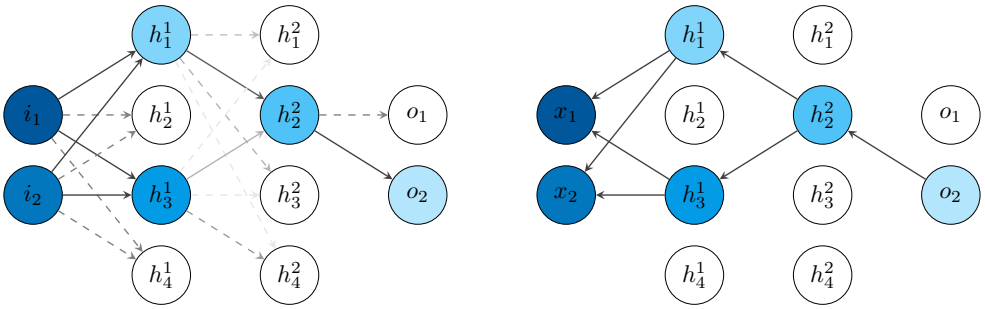
Initial approaches propose to incrementally pre-train each layer following the idea of greedy layer-wise unsupervised learning [Ben09]. These approaches are either based on the *Restricted Boltzmann Machine* (RBM) [HOT06] or some form of *auto-encoder* [Ben+06]. However, while using the sigmoid activation function, approaches for layer-wise pre-training are designed to bypass the vanishing gradient rather than mitigate this structural problem. A significant step in this regard was achieved by replacing the sigmoid activation function with the so-called *ReLU* [GBB11].

This activation function was originally proposed in the context of cortex-inspired digital circuits [Hah+00]. Early studies of neural networks introduced the rectifying activation function for recurrent networks [SA96; Hah98], while others integrated it in the context of cortex-inspired digital circuits [Hah+00]. Later, the ReLU was used to train a *deep belief network* (DBN) following greedy layer-wise unsupervised approach based on RBMs [NH10]. Subsequently, it has been shown that layer-wise pre-training can be omitted by replacing the deep belief networks with unsupervised trained auto-encoders using the ReLU as an activation function [GBB11]. Arguably, the convolutional neural network named *AlexNet* gained the most attention regarding the performance of deep neural networks [KSH12]. This network has been trained end-to-end in a supervised scenario, using the ReLU activation function, and outperformed the second-best error rate in the ILSVRC-2012 competition by 10.9 percent points from 26.2% to 15.3%. This gain in performance kicked off a series of deep network models, each reaching a new level of recognition performance in computer vision (cf. Section 2.4). The ReLU function cuts off the negative

values by setting them to zero while the positive values are linearly propagated. Formally, the ReLU and its derivative are defined as

$$relu(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & else \end{cases} \qquad relu'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & else \end{cases}, \tag{31}$$

where the derivative corresponds to the step function (cf. Equation 3). This characteristic gives the ReLU the desired property that positive gradients are not scaled, as the scaling factor is 1. According to [GBB11], using a rectifying activation allows a network to quickly obtain sparse representations, which yield several advantages like information disentangling or linear separability [Ben09]. Moreover, no exponential functions have to be computed in contrast to the sigmoid or hyperbolic tangent. However, a potential problem mentioned in [GBB11] is that the gradient might be *blocked* during the back-propagation due to the hard saturation at zero.



(a) forward (b) backward

Figure 13: Visualization of the forward (a) and backward (b) pass through an MLP using the ReLU activation function.

This circumstance is demonstrated by the example in Figure 13, where the forward pass is visualized on the left side, and the corresponding back-propagated gradients are shown on the right side. This example shows a Multi-Layer Perceptron consisting of two hidden layers ($\mathbf{h}^1$ and $\mathbf{h}^2$) as well as two-dimensional input and output layers ($\mathbf{i}$ and $\mathbf{o}$). Positive node values are highlighted by color intensities where darker colors represent higher outputs ($x > 0$), and colorless nodes imply values less than or equal to zero ($x \leq 0$). Arrows indicate the connections between the nodes, where positive valued weights are represented by continuous lines and negative ones are depicted by dashed lines. No lines are drawn between a neuron and the subsequent layer if the output is zero. Only gradients greater than zero are visualized on the right side by backward arrows. As one can see, every "zero" node in the Multi-Layer Perceptron blocks the gradient during the back-propagation. The authors in [GBB11] hypothesized that more network layers hurt the optimization. For this reason, a smooth alternative was proposed, referred to as *softplus* activation function [Dug+00]. The softplus function has an exponential slope at zero, and

its derivative, consequently, corresponds to the smoothed version of the step function, namely the sigmoid activation:

$$softplus(x) = \log(1 + e^x) \qquad softplus'(x) = \frac{1}{1 + e^{(-x)}} \ . \tag{32}$$

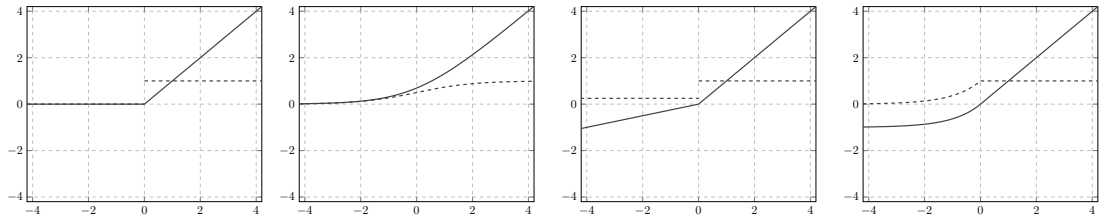
Multiple improved versions have been published since the initial introduction of the ReLU activation function. For example, a rectifying activation function with a non-zero slope for negative values is referred to as *LReLU* [Maa13]. Formally, this function and the resulting derivative are defined as

$$lrelu(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & else \end{cases} \qquad lrelu'(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0.01 & else \end{cases} \ . \tag{33}$$

The non-zero slope prevents the gradient of nodes with a negative output value from being scaled to zero and improves the training of acoustic neural network models. Shortly after, a generalized version of the ReLU was proposed where the non-zero slope is defined as a trainable parameter named *PReLU* [He+15a]. Similar to the softplus, the *ELU* activation function replaces the sharp rectifying property of the LReLU [CUH16]. Mathematically, the exponential linear unit and its derivative are defined as

$$elu(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & else \end{cases} \qquad elu'(x) = \begin{cases} 1 & \text{if } x > 0 \\ elu(x) + \alpha & else \end{cases} \ . \tag{34}$$

The main argument for this function is that the vanishing gradient is alleviated via the identity for positive values, while at the same time, the smooth slope for negative values pushes the mean unit activations closer to zero, which speeds up the training of the network parameters [CUH16]. Figure 14 shows the four activation functions described above. The continuous line visualizes the resulting activation level $f(x)$, while the dashed line depicts the respective derivation $f'(x)$.



(a) ReLU (b) softplus (c) Leaky ReLU (d) ELU

Figure 14: Visualization of the ReLU (a), softplus (b), LReLU (c), and ELU (d) functions.

2.4 MODERN ARCHITECTURES

Since the remarkable results achieved by the convolutional neural network named AlexNet [KSH12], a large diversity of network architectures has been introduced. Some of the initial architectures form the templates for today's modern neural networks, continuously achieving state-of-the-art performance in *computer vision*, *speech recognition* as well as *natural language processing* (NLP) tasks [WSH20; Alz+21; Khu+23; Pra+24]. Besides the LeNet [Lec+98] and AlexNet [KSH12], arguably the most influential network architectures can be summarized in the following as the VGGNet [SZ15], GoogLeNet [Sze+15], ResNet [He+16], DenseNet [Hua+17] and last but not least the Transformer [Vas+17]. This list of architectures is by no means complete. However, most neural networks build upon the ideas proposed by the publications above.

The LeNet architecture was one of the first convolutional neural networks that demonstrated exceeding performance in recognizing handwritten digits [Lec+98]. This network consists of only 60 000 model parameters and outperforms other standard classifiers such as the kNN, polynomial classifier, radial base function networks, MLPs, and SVMs. With the introduction of the rectified linear unit activation function and the computational power of graphical processing units (GPUs), the AlexNet demonstrated remarkable classification performance in the ILSVRC-2012[5] competition and pushed the error-rate by 10.9 percentage points from 26.2% to 15.3% compared to the second-best entry [KSH12]. The network architecture consists of 6 convolutional and 3 FC layers, totaling 62.3 million parameters. Two years later, Simonyan *et al.* explored a deeper architecture named VGGNet (Section 2.4.1) with an increased number of convolutional layers [SZ15]. The authors proposed multiple versions regarding the depth, ranging from 8 up to 16 convolutional layers while retaining a 3 layered MLP as the classifier. The version with 16 convolutional layers (19 weight layers) is the winner of the ILSVRC-2014[6] in classification consisting of 143.7 million trainable parameters. The following year, a novel architecture approach named GoogLeNet was presented in [Sze+15]. The network architecture is based on the so-called *inception* module, consisting of 3 parallel convolutional layers each using different kernel sizes (i.e.1×1, 3×3 and 5×5). Compared to the VGGNet, the authors built a network architecture of 21 inception modules (depth-wise) and one FC layer for classification. In contrast to the VGGNet, replacing a Multi-Layer Perceptron with a single fully connected layer drastically reduces model weights. Thus, the GoogLeNet only consists of 6.3 million trainable parameters. In the following year, a network architecture referred to as ResNet (Section 2.4.2) based on *residual skip connections* was presented [He+16]. The idea of skipping a residual block, consisting of multiple convolutional layers, improves the *gradient flow* and allows the training of much deeper architectures with up to 152 layers[7]. This immense

5 ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) [Rus+15c]: https://www.image-net.org/challenges/LSVRC/2012/

6 ImageNet Large Scale Visual Recognition Challenge 2014 (ILSVRC2014) [Rus+15c]: https://www.image-net.org/challenges/LSVRC/2014/

7 The authors presented experiments with architectures of 1201 convolutional layers. While the training procedure shows no optimization difficulties regarding a vanishing gradient, the testing results, on the

leap in network depth raised great attention across the computer vision community. Like GoogLeNet, only one fully connected layer is used to map between the network features and the class encoding. The smallest version of the ResNet, with 18 layers, contains 11.2 million parameters, and the largest version, with 152 weight layers, contains 58.2 million trainable parameters. Inspired by the success of the ResNet architecture, a convolutional neural network referred to as *DenseNet* pushes the idea of residual skip connections to the upper limit. Similar to the ResNet, the architecture is based on the idea of so-called *dense blocks* where layers are bypassed. However, in the DenseNet architecture, the output of a convolutional layer is connected to all subsequent layers within a dense block. This model achieves comparable results to the ResNet architecture on the CIFAR[8] dataset while at the same time reducing the number of parameters. The smallest model regarding trainable weights consists of 0.8 and the largest of 27.2 million parameters. Depth-wise, the authors presented architectures ranging from 40 to 250 trainable weight layers. Like the GoogLeNet and ResNet, one FC layer is used for class mapping. In natural language processing, a different type of network architecture has prevailed. For about two decades, language processing models have been dominated by RNNs such as the *LSTM* [Hoc98]. These *sequence-to-sequence* models were typically based on the so-called *encoder-decoder* architectures and suffered from the structural impediment of encoding the whole source sequence into a fixed-length vector representation from which a decoder generates the desired target sequence [SVL14; Cho+14]. This bottleneck was mitigated by introducing the so-called *attention mechanism*, which learns to align and translate jointly [BCB15].

Driven by the argument that neither recurrent nor convolutional layers are required to build a robust performing sequence-to-sequence model, an architecture based on attention modules was proposed as the *Transformer* [Vas+17]. The transformer architecture gives way to novel language model representations such as the *Bidirectional Encoder Representations from Transformers* (BERT) [Dev+19]. The novelty consists in the *pre-training* of deep bidirectional representations from unlabeled text, and subsequently, *fine-tune* the obtained model to a wide range of language processing tasks. While the BERT model is a pre-trained transformer (PT), a similar approach based on *generative pre-training* [HS06] led to a model referred to as *Generative Pretrained Transformer* (GPT) [Rad+18]. Since then, pre-trained (*task-agnostic*) transformer models have continuously demonstrated outstanding performance for different tasks across the field of natural language processing. However, a tremendous drawback of these large language models is the large number of trainable parameters, the required training data, and the incredibly long time it takes to train one model. For example, the third version, GPT-3, is a model with up to 175.0 billion parameters requiring 800GB storage and around 500 billion training tokens crawled from the web [Bro+20]. Besides the language processing domain, transformers are used for computer vision tasks as well and, thus, analogously referred to as *Vision Transformers* (ViT) [Dos+21].

other hand, are worse than the 151 layered version, although both versions perform equally on training data. The authors argue that the test-error differences are caused by overfitting [He+16].

8 Canadian Institute for Advanced Research (CIFAR): `https://www.cs.toronto.edu/ kriz/cifar.html`

Many different network architectures based on the ideas of GoogLeNet, ResNet, and Transformers have been since presented, steadily improving the model performance in the field of computer vision [Xie+17; Sze+17; HSS18; TL19]. However, the two essential architectures in the context of this thesis are the VGGNet (Section 2.4.1) and the ResNet (Section 2.4.2), as both are broadly used in the literature of word spotting (cf. Chap. 4) and especially in related approaches (cf. Chap. 5). For this reason, the architectural details of both models are examined in the following two sections.

2.4.1 *VGGNet*

The architecture of the VGGNet is one of the most frequently used convolutional neural networks regarding computer vision tasks [LSD15; BKC16; Ren+17]. The network follows a simple architectural design and mitigates some weaknesses of previously proposed architectures such as AlexNet [KSH12], ZFNet [ZF14] or OverFeat [Ser+14]. Instead of different-sized kernel filters along the convolutional layers, the VGGNet architecture follows a rigorous approach of a unified kernel filter size of 3×3 [SZ15]. The authors argue that a filter size of 7×7 as used in [ZF14] can be equivalently obtained by three consecutive convolutional layers, each consisting of 3×3 sized filters, effectively, leading to the same receptive field size (see Figure 11 in Section 2.3.3). A network architecture with larger filter sizes contains more trainable parameters and, consequently, is more prone to overfitting. Suppose a convolutional layer with 64 channels and a kernel size of 7×7. The amount of parameters would be 3 136. The equivalent of three convolutional layers, each containing 64 channels and a kernel size of 3×3, leads to 1 728 weights, which are only 55.1% of the initial amount of parameters. Furthermore, adding more convolutional layers results in more non-linear transformations, leading to more powerful representations [Ben09]. However, it is essential to note that the memory consumption for the example above grows by a factor of three due to the output of the convolutional layers. Multiple versions with different network depths, consisting of 11, 13, 16, and 19 trainable weight layers, have been proposed. The deepest versions with 16 and 19 layers are the most often used variants, thus referred to as VGG16 and VGG19, respectively. An overview of the VGG16 architecture is visualized in Figure 15. The convolutional layers are structured in five stages, and the number of channels is doubled for each stage. A maximum pooling (cf. Section 2.3.2) with a kernel size of 2×2 and a stride of 2×2 is applied before each (new) stage, reducing the feature dimensionality in height and width by a factor of 2. Figure 15 shows the reduction of the spatial dimensionality above each stage for the input size 224×244. After the last pooling operation, the resulting tensor of size $7 \times 7 \times 512$ is flattened into a 25 088 dimensional feature vector and, finally, mapped to the class encoding using three fully connected layers. The VGGNet architecture uses the ReLU activation function after every trainable weight layer except the last one, which maps the resulting representation to a one-hot encoded class vector by applying the Softmax function.

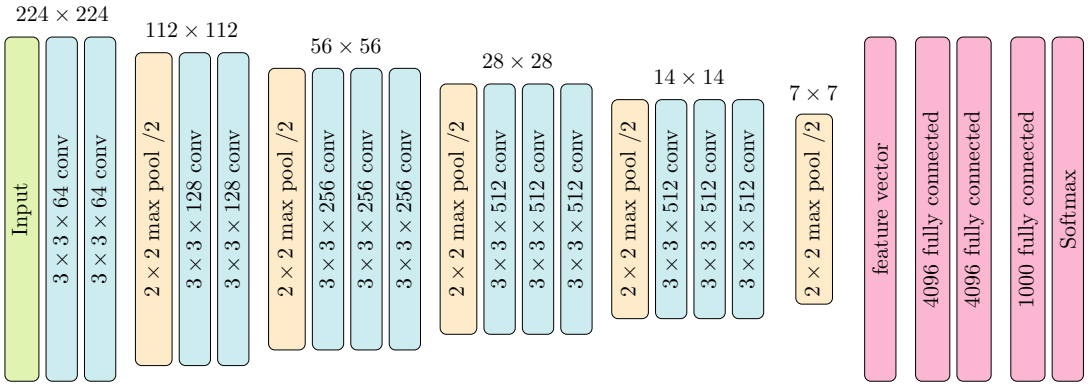


Figure 15: Visualization of the VGGNet architecture [SZ15] with 13 convolutional layers, 5 subsampling steps (pooling) and the MLP consisting of 3 FC layers. The numbers above the convolutional layers show how the original input size of 224×224 pixel evolves while being forwarded through the architecture.

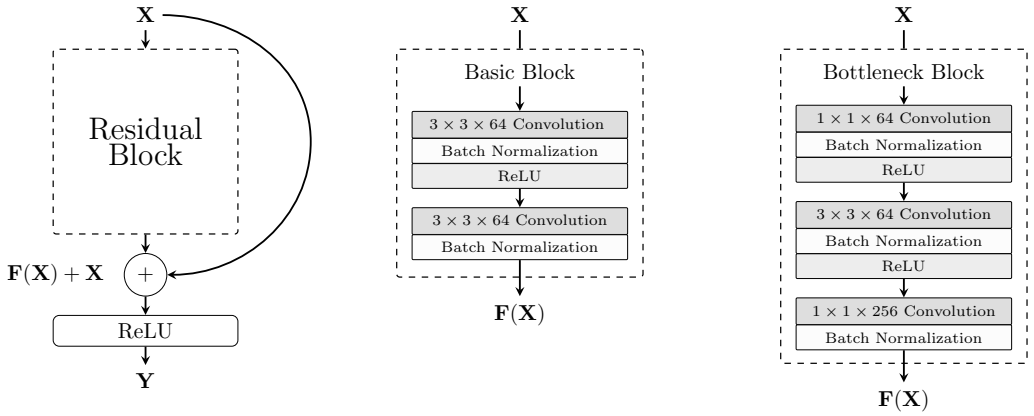
2.4.2 *ResNet*

Motivated by the evidential results that deeper architectures lead to higher recognition performance [SZ15; Sze+15] and network depth is crucial, the ResNet considerably elevates the depth of a convolutional neural network architecture. Although the notorious problem of vanishing gradients has mainly been addressed by techniques like normalized weight initialization [LeC+98; GB10; He+15a] and intermediate normalization layers [IS15], a degradation problem has been exposed for deeper networks [He+16]. The accuracy of a model saturates with a certain depth, which might be unsurprising. However, the surprising insight is that such degradation is not caused by overfitting, and adding more layers leads to a decline in accuracy as reported in [SGS15].

This problem is addressed by introducing the so-called *deep residual learning* framework [He+16]. The authors argue that the deeper counterpart of a shallower architecture can be constructed by adding *identity mapping* layers while the other layers are copied from the learned shallower model. Theoretically, this solution should produce no higher training error than its shallower counterpart by learning weights for the additional layers such that these layers perform identity mapping. However, experimental results demonstrate that adding more layers to the (same) architecture leads to higher training and test errors [He+16]. The residual learning is based on the following consideration: Instead of directly fitting a desired underlying mapping $\mathcal{H}(\mathbf{X})$, the stacked non-linear layers fit the following mapping $\mathcal{F}(\mathbf{X}) = \mathcal{H}(\mathbf{X}) - \mathbf{X}$ which can be rewritten as $\mathcal{H}(\mathbf{X}) = \mathcal{F}(\mathbf{X}) + \mathbf{X}$. The authors hypothesize that it is easier to optimize the function $\mathcal{F}(\mathbf{X})$ and the identity $\mathbf{X}$ than the unreferenced mapping $\mathcal{H}(\mathbf{X})$ [He+16].

A building block for residual learning or simply *residual block* is realized using *shortcut connections*, which skip one or more weight layers. The structure of a residual block is visualized on the left side of Figure 16. The input $\mathbf{X}$ is forwarded through the residual

block and added to the output of this block to realize the formulation $\mathcal{F}(\mathbf{X}) + \mathbf{X}$ [He+16]. Subsequently, the resulting sum is transformed through the ReLU non-linearity.



(a) Shortcut connection (b) Residual basic block (c) Residual Bottleneck block

Figure 16: Visualization of the shortcut connection (a) skipping a residual block. Depending on the chosen model version, the residual block's structure is either the basic block (b) or a bottleneck block (c).

Two different types of residual blocks exist, namely *basic block* and *bottleneck block*. The basic residual block is shown in the center (b) of Figure 16 and consists of two convolutional layers, each following the architectural approach of the VGGNet only using 3×3 kernel filters. The outputs of the convolutional layers are normalized by applying a batch normalization as proposed in [IS15]. A ReLU non-linearity is applied after the first (batch normalized) convolutional output, while the second ReLU function is applied outside the residual block, as mentioned before. A deeper bottle architecture with up to 152 trainable layers is introduced to increase the depth further. This architecture is based on the bottleneck blocks, which are depicted on the right side (c) of Figure 16. A bottleneck block consists of three convolutional layers with kernel sizes of 1×1, 3×3, and again 1×1. The bottleneck property is realized using the 1×1 filters, which reduce and then increase (restore) the dimensionality. Based on this, the 3×3 filters effectively operate on smaller dimensions. Analog to the basic block, each convolutional layer output is batch normalized and transformed through a ReLU non-linearity. The authors note that the bottleneck design is solely introduced to reduce the computational complexity in terms of floating-point operations [He+16]. However, it is important to note that integrating more convolutional layers increases memory consumption, similar to the VGGNet architecture. Although the bottleneck block reduces the number of floating-point operations by 22.2% compared to the basic block, the memory consumed increases by a factor of 3. Five versions, consisting of 18, 34, 50, 101, and 152 trainable layers, were initially presented. An overview of the smallest variant with 18 layers is visualized in Figure 17. The first convolutional layers directly reduce the dimensionality in height and width using a filter size of 7×7 and a stride of 2. Subsequently, maximum pooling

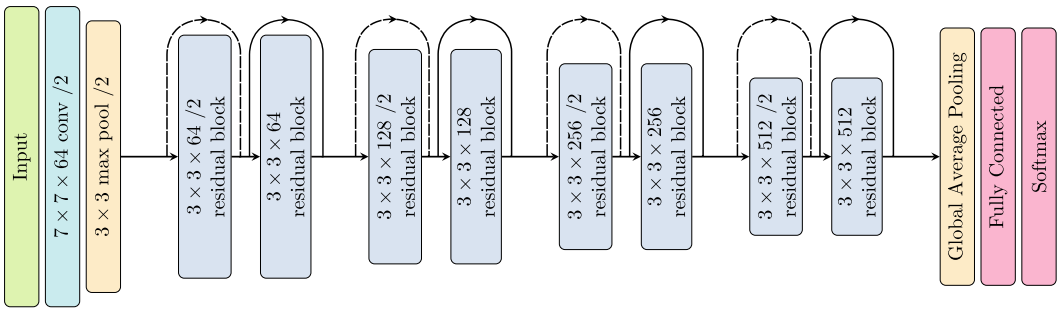


Figure 17: Visualization of the ResNet [He+16] architecture showing the model version ResNet_{34} with 8 residual blocks, 33 convolutional layers and a final fully connected layer.

is applied using a stride of 2 and, different from the VGGNet, a pooling kernel size of 3×3. Afterwards, the resulting feature maps are processed through 8 residual blocks, and the factor of 2 reduces the dimensionality after every second block (denoted by /2). In Figure 17, a dimensionality reduction in height and width is shown using a dashed shortcut connection, while solid lines visualize unchanged dimensionalities. Within a residual block, the dimensionality of a feature map is reduced by using a stride of 2 for the second convolutional layer in the basic and the bottleneck block. For shortcut connections, feature map dimensions along height and width are reduced using a 1×1 convolutional layer with stride 2. Furthermore, if the path through the residual block increases the number of channels in the resulting feature map, the ResNet architecture provides two options. The identity mapping is padded with zeros or transformed by a convolutional layer using a kernel size of 1×1 to fit the dimensionality. The resulting features are mapped to the predicted class vector in the final stage. However, in contrast to the VGGNet architecture, classification is realized by applying a global average pooling, resulting in a 512 or 2 048 dimensional feature vector, depending on the depth variant of the network. This feature vector is then transformed to a one-hot class encoding using a single fully connected layer followed by the softmax function. As mentioned in Section 2.4, omitting the fully connected layers drastically reduces the amount of model parameters. Supposing the global average pooling is applied to a feature map containing 512 respectively 2 048 filters, the fully-connected layer in the ResNet architecture only consists of 512 thousand or 2.05 million parameters for the ImageNet [Rus+15c] classification task with 1 000 categories. A Multi-Layer Perceptron, as used in the VGGNet, would have 22.97 and 29.26 million parameters, respectively, for the same number of feature maps and classes.

3

KNOWLEDGE TRANSFER BETWEEN CATEGORIES

In the last two decades, computer vision has significantly advanced image classification and object recognition. With the advent of deep convolutional neural networks [Lec+98; KSH12], visual recognition performance has been elevated to an astonishing level. The performance of computer vision models is based on the enormous amount of training data, for which datasets like ImageNet [Rus+15c] or MS COCO [Lin+14] have been proposed, containing tens of thousands and even millions of training samples. This *multi-class* classification task is mostly solved by a class mapping through a one-hot encoded vector where the model predicts one out of K categories for a given image (cf Section 2.1.2). This approach works well for tasks with a small number of categories supported by sufficient training data.

However, many real-world applications face the problem of insufficient training material, which may have several reasons. For some particular tasks, it is often expensive and time-consuming to obtain annotated training samples [Wan+19], such as in the semantic segmentation problem [Min+22] or image captioning [Ste+22]. Other tasks have to deal with tens of thousands of categories. Although obtaining a large number of training samples for common categories like vehicles or dogs is feasible, it is a considerable effort to obtain annotated training samples for infrequent categories, such as particular vehicle models or specific dog breeds [FLP17]. According to [Bie87], tens of thousands of superior and significantly more subordinate categories exist. Different approaches have been proposed in order to deal with this problem. For small training data with just a few samples for each category, methods for *one-shot learning* [FFP06] or *few-shot learning* [RL17] are introduced. If a task contains a large number of categories, techniques such as *cumulative learning* [FWL16] or *class-incremental learning* [Reb+17] are applied. For example, in text recognition, Jaderberg *et al.* [Jad+16] use a lexicon of 10 000 words represented through a one-hot encoded vector. An incremental learning approach is applied to handle the lexicon size. The training process is split into several steps, and the number of word categories progressively increases during training. For applications with a potential open set of target categories, methods like *open set recognition* [Sch+13] and *open world recognition* [BB15] are proposed. These methods can detect unknown target categories but cannot determine which specific category an instance belongs to.

3.1 SEMANTIC PROPERTIES FOR ZERO-SHOT LEARNING

The approaches that deal with the problem of classifying *unseen* target categories are referred to as zero-shot learning [LNH09; Far+09] or zero-data learning [LEB08]. Zero-shot learning is the task of recognizing categories for which no training samples are available

(i.e., zero data). However, the *unseen* category must be known beforehand. The general approach of zero-shot learning is to exploit the training data's knowledge to recognize unseen test categories. The problem formulation of knowledge transfer is studied over a wide area of methods referred to as *transfer learning* [PY10; Zhu+21]. In transfer learning, the knowledge contained in the source domain (e.g., training data) is transferred to the target domain (e.g., test data) to learn the target task. If the category spaces covered by the training and test data are disjoint, transfer learning is classified as *heterogeneous*; for the case of jointed category spaces, transfer learning is called *homogeneous*. Thus, zero-shot learning can be seen as a heterogeneous transfer learning problem [Wan+19]. In zero-shot learning, knowledge can be transferred by defining the categories through intermediate semantic properties shared across category boundaries. Unseen categories can be classified by first predicting the semantic properties for a given image and then determining the (*unseen*) category for which the combination of properties fits best. However, this approach requires each category to be represented by a unique set of properties in order to be distinguishable [LNH14]. Different approaches exist for obtaining semantic spaces that contain information about categories. According to [Wan+19], the spaces can be categorized into *learned* and *engineered* semantic spaces. Learned semantic spaces are prototypes of each category of interest obtained from the output of some learned models. As human experts did not design these prototypes, different output components do not necessarily have an explicit semantic meaning. Prominent approaches for learned semantics are word embedding techniques for natural language processing, such as Word2Vec [Mik+13] or GloVe [PSM14]. While other approaches learn the embedding model from Wikipedia descriptions [Soc+13]. Learned semantic spaces have the advantage of being less labor intensive due to their synthetic nature. Additionally, learned semantics can contain discriminative information not captured by human-designed representations. On the other hand, engineered semantic spaces are designed by human experts, and information about categories is incorporated explicitly into the semantic representations. However, the disadvantage of engineered semantics is the heavy reliance on human assessment and the time-consuming process. Engineered semantic spaces can be obtained from lexical databases like WordNet as demonstrated in [Aka+16]. Text descriptions of categories can also be exploited as semantic spaces, for example, by including object descriptions from websites like Wikipedia [Aka+15] or domain-specific encyclopedias [ESE13].

3.2 SHARING KNOWLEDGE WITH ATTRIBUTES

Besides the lexical databases and text descriptions, the most popular approach to implementing semantic properties is referred to as *attributes* [FLP17]. Attributes are human-specified high-level descriptions of arbitrary semantic properties like shape, color, or environmental information [LNH09]. According to Lampert *et al.* [LNH09], attributes are properties of objects if a human can decide whether the property is present or not for a particular object. The authors argue that attributes are related to human capabilities of identifying objects only based on their descriptions. When given a sentence like "large

gray animals with long trunks", humans might reliably identify elephants [LNH14]. Consequently, attributes for the class elephant could be *large*, *gray* and *long trunks*. Lampert *et al.* states that attributes are not required to be directly visible, like long trunks, but are nameable properties, such as an animal's natural habitat. According to Ferrari *et al.* [FZ07], attributes are purely visual properties, such as *striped*, *dotted*, *round*, *squared*, *red* or *blue*, which is why the authors explicitly name these properties *visual attributes*. In general, attributes are intermediate representations of semantic properties which are human-interpretable. However, the definition of attributes remains imprecise. For example, it is unclear whether attributes should be assigned to categories or image instances. If attributes are assigned for a given image, recognizing these attributes would be nothing else than *multi-label* classification [DHS01; Bis09]. An example is the classification of scene images, where the term attributes refers to a multi-label annotation for scene images without unique representations for scene categories [Pat+14]. Assigning attributes to categories, on the other hand, reduces the annotation time and allows sharing of knowledge across categories. Both Farhadi *et al.* [Far+09] and Lampert *et al.* [LNH09] use attributes under the second paradigm in order to recognize object categories efficiently. Despite the different definitions, technically, attributes are often applied as binary values $\mathbf{y} \in \{0,1\}^n$ referred to as *binary attributes*. Binary-valued attributes are often used due to their straightforward interpretation. The binary values indicate whether a property, represented by an attribute, is present (i.e., 1 or 0). Attributes are not limited to binary values and can be defined as real-valued numbers $\mathbf{y} \in \mathbb{R}^n$. These attributes are referred to as *continuous* [Wan+19]. Continuous attributes can indicate intensities, confidences, or counts, and their numerical range depends on the specific domain to which they are applied. Furthermore, Parikh *et al.* [PG11] successfully demonstrated that attributes can be defined as relations between images, indicating whether a particular attribute fits more for one of two given images, which are, consequently, referred to as *relative attributes*.

Semantic attributes are used in a broad range of different computer vision applications. They can be helpful for interactive recognition of fine-grained categories [Bra+10], or image search with humans in the loop [KPG15] as well as actively selecting annotations [KVG11]. As the semantics are human-interpretable, attributes are provided as rationales to machines for more substantial supervision [DG11], exploit as privileged information [SQL13], and serve as decision explanations to humans [Hen+16]. Attribute-based representations play a crucial role in person re-identification [SHX15], facial attributes recognition [Yan+15; Zha+16], estimating human poses [Zha+14] or recognizing pedestrians [Tan+19]. In e-commerce, they are applied to clothing retrieval [Hua+15] or recommending fashion styles [Liu+13]. Recently, class attributes have been used for human activity recognition [RF18; RF21]. Here, attributes represent fine-grained semantics, such as *lifting right hand* or *moving left leg*, to recognize complex human activities. Besides that, applications like scene analysis [Pat+14] or image retrieval [DRS11] can benefit from attribute representations.

3.3 SEMANTIC ATTRIBUTES IN THE CONTEXT OF HANDWRITTEN DOCUMENTS

In document image analysis, collections of documents containing handwritten texts face similar challenges, which are addressed by the approaches above using semantic attributes. If words are defined as individual categories, methods in this area must deal with an extensive vocabulary. Moreover, the available training data do not include many handwritten words that need to be transcribed or retrieved. This circumstance requires zero-shot learning techniques. In addition, extensive data collections of handwritten texts are not available, especially not for historical documents. These challenges of *many classes*, *unseen samples*, and *scarce data* are addressed by the concept of semantic attributes. Several approaches have been proposed for the word spotting task using binary [Alm+14b] or continues [Jos13; WB16] attributes to represent large lexicons. These approaches define single or multiple characters from an alphabet as individual semantics represented by attributes. Moreover, visual characteristics of individual characters can be defined as semantics as well [RKC21]. Approaches that address the word spotting task using semantic attributes are highly related to the methodology presented in this thesis and, thereof, discussed in more detail in Chap. 5.

4

WORD SPOTTING ON HANDWRITTEN DOCUMENTS

In document analysis, the approach known as word spotting has attracted considerable research interest over the last two decades. The task of word spotting is to retrieve all relevant candidates of word images from a given document collection with respect to a user-defined query. Ideally, the candidate list retrieved by the word spotting system is sorted according to a similarity measurement in a decreasing order, where the most similar candidates will appear in the first positions in the retrieval list. This essential requirement of presenting the results makes word spotting a particular case of content-based image retrieval (CBIR) [Alm+14b]. Due to the significant overlap in methodologies, word spotting often needs to be clarified with word recognition. However, text recognition aims to find the correct transcription for a given word image and return only a single recognition result. Different methods for text recognition have been successfully applied to machine-printed documents [Rus+15a]. While the transcriptions obtained for machine-printed documents generally yield satisfactory results, this quality of performance significantly drops when dealing with handwritten documents. This is especially true for historical document collections. Compared to machine-printed documents, handwritten texts contain a large variability in writing styles caused by different writers, where parts of the text are not recognizable due to sloppy handwriting. For historical handwritten documents, different aging effects, fading ink, or bleed-throughs increase the difficulty level. Written texts often consist of different vocabularies and spellings, which present a significant challenge for language models. Word spotting circumvents the problem of incorrect transcription by retrieving a list of relevant candidates instead of presenting a single result. The candidate list retrieved by the word spotting system offers a significant advantage: The user can choose from several possible candidates. A word spotting system mainly consists of three processing stages. The pages from a document collection are, first, *segmented* into word images and subsequently transformed into *word image representations*. Once the images are mapped to their respective representations, a *similarity measurement* is applied to obtain the similarity scores between the word images and a given query.

This chapter presents the task of word spotting on handwritten documents and is structured as follows. Section 4.1 introduces the established terminology used across the word spotting literature. Subsequently, the beginnings of word spotting are briefly introduced in Section 4.2. Following that, the evolution of word image representations is presented in Section 4.3. Finally, the approaches for evaluating similarity scores are discussed in Section 4.4. It is important to note that this thesis assumes that word images are already segmented. As a result, methods dealing with the problem of word region segmentation are not discussed. Instead, the word spotting works [WLB17; Gio+17; RWF21] are referred for further reading.

4.1 TERMINOLOGY

Word spotting is increasingly used to aid in document analysis by searching through collections. Therefore, the word regions must be located to extract the relevant document parts (i.e., word images). Locating the relevant regions is referred to as *segmentation*. Afterwards, the segmented word images are transformed into numerical representations, and similarities are computed with respect to the query. A query can be provided in different *query modalities* and depends on the supported modalities of the word spotting system. The word spotting methods can be characterized concerning the aforementioned three properties, which are the necessity of *annotated data*, the functionality of *segmenting* document images, and the support of different *query modalities*.

If a word spotting method requires annotated training samples to learn the mapping between word images and their corresponding annotations, it is characterized as *annotation-based* and *annotation-free* otherwise. In [Gio+17], word spotting methods are characterized into *learning-based* and *learning-free* approaches with a focus on the model parameters being adjusted. This characterization is extended in [Sud18, p. 40] with the argument that a considerable amount of methods characterized as *learning-free* make use of *unsupervised* training on the supplied annotations and thus should be distinguished into *supervised*, *unsupervised* and *training-free* methods. In recent years, classic *training-free* methods have become infrequent and give way to approaches using convolutional neural networks. Thus, in this thesis, word spotting methods are characterized as *annotation-based* or *annotation-free*, according to annotated training samples required to obtain the model parameters. In the context of this thesis, *annotation-free* methods do not require annotated data from the target document collection. It is important to note that different techniques like *transfer-learning*, *self-supervised learning* or *domain-adaptation* make use of annotated data sets from other document collections or synthetically generated handwriting for training the word spotting models [KJ16; KDJ16; WF20]. However, they do not require annotated data from the target domain.

A word spotting system's input typically consists of page images from a document collection and a user-defined query. Word regions must be segmented prior to retrieval to make the document searchable. If the segmentation is integrated into the word spotting method, it is referred to as *segmentation-free*. On the other hand, if the method requires segmented words or text lines, it is said to be *segmentation-based*. According to [Rot19], word spotting methods can be grouped into three categories, namely *word-level*, *line-level*, and *document-level*. Approaches operating on the word level require the word regions to be segmented in advance, while line-level approaches depend on segmenting text lines. In general, it is more challenging to extract word regions than text lines from document pages. In the context of this thesis, word spotting methods are characterized as *segmentation-free* and *segmentation-based*.

A word spotting system is applied to a document collection where relevant parts are extracted based on a user-defined query. The most prominent query modality is given

as a visual example referred to as *QbE*. For word spotting, visual examples are usually images containing a single word (i.e., word images). The QbE retrieval scenario has the advantage that no prior knowledge is required on the user's part about the content of the document collection. Furthermore, as the similarity of candidates in the retrieval list can be evaluated purely on the visual appearance, no recognition step is required. Hence, many annotation-free approaches [Ret+19; RWF21; ZPG17] supporting the QbE spotting scenario have been published. However, the user must search through the documents to find the desired visual example. This process can be very tedious, especially for infrequent words. This limitation is circumvented by another prevalent query modality referred to as *QbS*, where the query is presented as a textual string. The obvious advantage is that the user no longer has to search through the document collection; instead, the desired word string must be defined. However, a considerable disadvantage is the necessity of annotated training samples. A set of annotated samples is required to learn the mapping between word images and their corresponding textual representation. Due to the necessity of mapping from visual to textual representation, word spotting methods supporting QbS retrieval are characterized as annotation-based approaches. Besides QbE and QbS, several other query types are used in word spotting. For example, Rusiñol *et al.* [Rus+15b] present a QbSP retrieval scenario where spoken words are used as queries. In [WRF16], queries are defined by handwriting on a pen-based system referred to as QbO.

4.2 WORD SPOTTING ORIGINS

Word spotting originates from signal processing, where spotting has been applied to speech before [Fis+12]. For example, Myers *et al.* [MRR80] compare a fixed range and local minimum version of the DTW algorithm in order to perform word spotting and connected word recognition on a dataset of audio recordings over a dialed-up telephone. Later, Rohlicek *et al.* [Roh+89] use continues HMMs with Gaussian output modeling applied on audio feature representations in order to perform spoken word spotting.

In the early 1990s, word spotting was adapted by the document analysis community for machine-printed documents. Khoubyari *et al.* [KH93] propose a method for segmentation-based query-by-example word spotting in noisy printed document images. The approach is based on XOR images computed between the query and word images from the document collection. An XOR image is obtained by applying the XOR operation to corresponding pixels, where white pixels (i.e., 1) indicate a difference and black pixels (i.e., 0) indicate equality. Therefore, all images are rescaled to the same size and aspect ratio. The word images are rescaled by aligning them according to a horizontal band of fixed height covering the pen stroke's most dense area. In the next step, all white pixels are replaced by the Euclidean distance between the white pixel and the nearest black pixel. Finally, the similarity score is assessed by calculating the sum of squares from all pixel values in the XOR image and subsequently normalizing the resulting value L_1. This approach is referred to as Euclidean distance matching (EDM). Chen *et al.* [CWB93] present an ap-

proach for font-independent segmentation-free query-by-string word spotting in scanned machine printed documents. The word regions are extracted using morphological closing operations to merge characters within a word and compute the bounding box from the connected components. Word image representations are based on the external shape and internal structure of the image obtained from pixel values that are extracted column-wise. The keywords are modeled through context-dependent HMMs while sub-character HMMs are used for non-keywords. Inspired by the 2D nature of printed documents, Kuo *et al.* [KA94] present an approach based on pseudo 2D HMMs. Instead of fully connected networks, the usual 1D horizontal models are linked with vertical superstates, giving them the prefix "pseudo 2D". These models achieve superior performance in a segmentation-based QbW retrieval scenario compared to the standard 1D HMMs.

In the document analysis community, handwritten text is considered to be much more difficult to analyze than machine-printed text. Handwritten documents exhibit significantly greater visual variability in the writing style and structure. Multiple instances of the same word on the document page written by the same writer show differences. Inspired by the word spotting works on printed documents mentioned above, Manmatha *et al.* [MHR96] present the first approach for word spotting on handwritten texts in historical documents. The document pages are, first, binarized and segmented using a process of smoothing and thresholding as proposed in [Man+96]. Subsequently, the segmented word images are eliminated from the candidate's list based on a mismatch of their size and aspect ratio compared to the query image. The authors use two different algorithms for word image matching. The first is similar to [KH93] based on XOR images and Euclidean Distance Matching (EDM). The second algorithm models the transformation between words as an affine transform referred to as SLH [SL91]. Manmatha *et al.* argues that matching based on affine transformations outperforms the EDM based on XOR images. The reason is that a pixel-wise image comparison will show considerable differences for two equal words, although written by the same writer.

4.3 WORD IMAGE REPRESENTATIONS

Word images need to be transformed into numerical feature representations. Afterward, similarities are computed between a given query and the representations. Word image representations are strongly inspired by techniques from computer vision and handwriting recognition. Three different types of word image representations exist, mainly based on the position and order of the extracted features. The word image can be represented holistically as a single feature vector. A sequential word representation is obtained by extracting feature vector frame-wise along the writing direction. Another approach of representing word images, is to compute local image features at previously detected interest-points. Besides that, further image representations exist as hybrids of the three main types. In this regard, the prevailing representation is based on features extracted in a pyramidal

fashion and concatenated into a holistic feature vector. This section reviews the different approaches for representing word images.

4.3.1 *Geometric Features*

In the first work for word spotting on handwritten documents, Manmatha *et al.* use the XOR operation to represent word images [MHR96]. The pixel-wise comparison, however, is hardly applicable to handwritten words. Two equal words written by the same writer can differ considerably. Around the millennium, ensuing works [KGG97; Kol+00; RM03] moved towards geometric features as word image representations. Geometric features encode the structural properties of handwritten words. As handwriting is based on the pen-strokes, geometric features are sometimes referred to as *pen-stroke* features [Rot19]. Typically, geometric features are obtained by computing different word profiles, such as upper- and lower-profiles, character cavities, or pixel distributions, like the number of foreground-background transitions [Gio+17]. However, geometric features suffer from two considerable drawbacks. Geometric features are sensitive to slight variations in the writing style and often require a normalization of the slant, skew, or image size [FB14]. The structural properties of geometric features are based on extracted pen strokes and, therefore, require binarization and skeletonization steps in advance [Gat14]. Besides the word profiles and transition counts, images are transformed into a frequency domain by applying the DFT [Sze22, pp. 113–115]. Then, the lower-order frequency coefficients are used as feature vectors [KGG97; RLM03]. The representations of word images can be enhanced by combining locally neighbored features into histograms. Therefore, the word image is partitioned into a grid of cells. The discrete features are counted within every cell and, subsequently, concatenated in order to build a histogram. The dimensionality of the feature vector is defined by the number of cells and the histogram bins. Approaches like the shape context [LS07] or the blurred shape model [For+11] count the pen-stroke pixel within each cell. In [FLF11], characteristic Loci features are used as word image representations. Loci features [Glu69] encoded the number of foreground-background transitions in four directions (i.e., horizontal and vertical) starting from a specific keypoint. The counts extracted at each keypoint position are called Locu codes, and the histogram is obtained by concatenating these codes.

4.3.2 *Descriptors based on Gradient Histograms*

Geometric features bear the drawback of being sensitive to small visual variations in word images. In order to detect the pen stroke, the word image must be binarized and skeletonized beforehand. For some of the approaches, keypoint detection is required. Small changes in the visual pen-stroke appearance immediately affect the pen-stroke detection. While binarization and skeletonization produce satisfactory results for high-contrast or printed document images, they are less effective for low-contrast handwritten documents.

Binarization and skeletonization can become highly challenging, especially for historical documents, due to aging artifacts, fading ink, or bleed-through.

From the mid-2000s, the focus of image representations shifted towards local image features based on *gradient histograms*. Gradient-based representations encode the entire image region, not just the extracted pen stroke. These histograms are directly obtained from gray-scaled images and eliminate the necessity for binarization or skeletonization. This advantage makes these representations more robust against visual variations in the pen-stroke appearance. The gradient-based histograms are adapted versions of the very popular SIFT [Low04] and HoG [DT05] algorithms, mainly differing in cell arrangement and histogram normalization. A HOG descriptor is obtained by sampling the gradient histograms from a rigid grid layout. This approach is used in [TT09] to compute sequences of HOG descriptors along handwritten text lines. Kovalchuk *et al.* [KWD14] combine HOG and LBP descriptors [AHP06], both obtained from binarized word images. In [Alm+12], whole document pages are represented in dense grids of HOG descriptors. Segmentation-free word spotting is performed through a patch-based approach, where locally neighbored descriptors represent each patch. In contrast, the original SIFT algorithm extracts the descriptors from specific keypoints computed in advance. For handwritten text images, keypoints are usually detected using algorithms such as Harris corners [HS88]. After the SIFT descriptors are extracted, retrieval is performed by matching these keypoints.

4.3.3 *Descriptor Quantization*

Finding the best alignment between all the keypoints is computationally exhaustive. Consequently, image descriptors based on the SIFT are often used in the BoF framework [OD11] that has been applied in segmentation-based word spotting scenarios [AD07; YM12; SJ12]. Quantizing local image descriptors into a visual vocabulary yields a holistic and fixed-length word image representation while keeping the discriminative capabilities of local descriptors [Ald+15]. According to [NJT06], the performance of BoF representations correlates with the number of sampled regions. The performance gain is related to the number of regions rather than the kind of sampled regions. This insight led to a sampling strategy where local gradient histograms are sampled according to a dense grid layout, similar to the HOG approach. Rusinol *et al.* [Rus+11] pursue this sampling strategy in a segmentation-free word spotting scenario. SIFT descriptors are sampled over a regular grid where descriptors with a low gradient magnitude are discarded. Afterward, clustering is applied using the BoF approach. The resulting representations are orderless collections of quantized descriptors lacking any structure of spatial information [OD11]. In order to retain this information, a pyramidal partitioning of the image is applied [LSP06]. Besides a holistic representation of word images, sequences of BoF histograms are used as input for HMMs referred to as *BoF-HMM* [RVF12]. Therefore, SIFT descriptors are extracted at each interest point detected by the Harris corners algorithm [HS88]. A sequence of BoF histograms is obtained by moving a window over the word image in the writing di-

rection. Only interest points inside the window are considered at each window position. In [RMF13], this BoF-HMM approach is extended towards segmentation-free QbE word spotting. In contrast to [RVF12], the SIFT descriptors are sampled from a dense grid over the document page and clustered according to the BoF framework. Afterward, word spotting is performed in a patch-based approach similar to [Rus+11]. The sequences of BoF histograms are extracted column-wise at each patch position.

Alongside the BoF histograms, Fisher vectors are utilized as word image representations. Almazán *et al.* [Alm+14a] use Fisher vectors in a segmentation-free word spotting scenario. The authors reutilize the approach from [Alm+12] where document images are represented as dense grids of HOG descriptors. Afterward, potential candidates are extracted from patches moved over the descriptor grids. Fisher vector representations are used to re-rank the candidate's list to expand the training set in an Exemplar-SVM framework [MGE11]. Later Almazán *et al.* [Alm+14b] use the Fisher vector on top of densely sampled SIFT descriptors to perform segmentation-based QbE as well as QbS word spotting. The authors retain spatial information by subdividing the grid of SIFT descriptors into 12 bins (6×2) and estimate individual GMMs for each bin. Computing higher-order statistics like Fisher vectors yields more discriminative representations compared to quantized descriptor frequencies obtained from BoF histograms [Sán+13]. However, due to the high computational demand, Fisher vectors are mainly used for segmentation-based word spotting [Alm+14b] and re-ranking [Alm+14a].

4.3.4 *Learned Representations*

Word image representations based on geometric features, gradient histograms, or quantization techniques (i.e, BoF or Fisher vectors) are directly obtained from document images. Therefore, annotated material is not required. On the other hand, learned word image representations require annotated training data to predict class information. The annotation granularity depends on the problem domain and is obtained on the character or word level.

For QbS retrieval scenarios, annotations on character level are typically used as visual character templates to obtain sequence models such as HMMs [Edw+04; CZF06]. The approach in [Tho+15] uses HMMs to segment individual characters in handwritten text lines. Subsequently, these segmented characters are used to train character models. In both cases, visual character samples must be segmented, which is a tedious task, especially for handwritten documents. Transitions between characters are difficult to distinguish, and, in contrast to machine-printed texts, handwritten characters vary considerably in width.

As a result, word spotting benchmarks more often provide annotations on the word level. Word annotations are used to learn a linear embedding of local image descriptors to perform QbE word spotting [SRF15]. The approach is inspired by [Phi+10] and [HBW07] based on an embedding where distances between descriptors correspond to their semantic similarity. To learn the embedding, training samples of corresponding and non-

corresponding descriptor pairs are obtained from annotated word images. Another approach exploits a CNN to obtain word image representations for query-by-example word spotting [SK15]. The CNN is trained to assign word classes for a given input. Therefore, 1000 most frequent word classes are extracted from target or synthesized datasets. Retrieval can be performed by representing word images through the output of the penultimate FC layer. In [SRG16], a CNN is pre-trained to classify individual characters [JVZ14], and, subsequently, used as a feature extractor to perform QbE word spotting. In contrast to [SK15], image descriptors are obtained from the output of the last convolutional layer. Inspired by [BL15], convolutional features are split into 6 zones along the horizontal axis and aggregated to a fixed-length vector representing the word image descriptor. Furthermore, word image representations are obtained from CNN models trained using deep metric learning techniques, referred to as *Triplet-CNN* [WB16]. The name originates from the *triplet loss* function applied during training [HA15]. A triplet consists of three images with two corresponding examples (i.e., anchor and positive) and a third non-corresponding example (i.e., negative). The loss function is designed to minimize the distance between the anchor and the positive image sample and maximize the distance between the anchor and the negative sample. RNNs are used for QbS word spotting in text lines [Fri+12]. A *BLSTM* model is trained on annotated text lines using the *CTC* algorithm [Gra+06]. This approach results in a sequential model representing a handwritten text line as a sequence of one-hot encoded character class estimations.

4.4 SIMILARITY MEASUREMENT

Once the word image representations are obtained, a measure for similarity assessment is applied. A numerical value representing the similarity is evaluated between the query and all word image candidates from the document collection. In the literature, two different approaches for similarity measurement are most popular. Distance-based approaches assess similarities directly by applying distance metrics such as the Euclidean distance. These approaches do not require any learning parameters and are thus suitable for learning- or annotation-free word spotting methods. In contrast, *learning-based* approaches train a query model and evaluate similarities according to model parameters. The model parameters are usually obtained from annotated training samples or estimated from the query given as a visual example.

4.4.1 *Distance-based Similarity*

For machine printed documents, the similarity between two word images is evaluated by the Euclidean distance mapping (EDM) algorithm [Dan80]. Therefore, XOR images are computed from binarized and pairwise aligned word images. Then, the Euclidean distance is computed for every differing pixel (indicated by 1) to the nearest non-differing pixel (indicated by 0). A similarity score between two word images is obtained by accumulating

all distances [KH93]. This alignment approach has been adapted to handwritten historical documents in [MHR96]. However, this pixel-level approach leads to insufficient retrieval performance as multiple handwritten word images obtained from the same writer show considerable differences on the pixel level. Thus, the authors in [MHR96] increase the robustness by applying a more sophisticated alignment strategy that includes baseline estimates and alignment in the horizontal and vertical directions. Furthermore, the similarity between word images is modeled as an affine transformation using the SLH algorithm [SL91] that outperforms the EDM algorithm [MHR96].

Ensuing works apply a more complex matching scheme based on an optimal alignment of two sequences of feature vectors. Regarding the alignment, the prevalent approach is referred to as DTW initially proposed for speech recognition [SC78]. In early word spotting approaches, this alignment technique is applied to sequences of geometric features obtained from column pixels of word images [Kol+00; RLM03]. However, extracting column-wise feature vectors can quickly result in long vector sequences, making the DTW alignment computationally expensive. With the advent of gradient-based histograms (cf. Section 4.3.2), column-based feature extractions has given way to frame-based approaches. HOG descriptors are extracted from word image frames in a sliding window approach along the text line [TT09], where the number of feature vectors is further reduced by extracting overlapping zones. Each of the zones is represented by DFT coefficients that are referred to as *modified projections of oriented gradients (mPOG)* [Ret+19]. This annotation-free approach achieves state-of-the-art results for segmentation-based query-by-example word spotting.

A similarity measure based on the spatial consistency of matching features is presented in [RFR03]. Therefore, the authors extract local features at interest points using the Harris corners algorithm [HS88]. The similarity is obtained by computing the accumulated displacement of corresponding image locations using the sum of squared differences (SSD). A cohesive elastic matching is applied in [Ley+09], where matches are restricted to local neighborhoods defined by regions of interest. In [ZPG17], center keypoints of potential query words are detected in the document image. Then, locally neighbored keypoints are projected into size-normalized coordinate systems relative to the center points. Finally, only spatially consistent keypoint matches are considered for similarity measurement computation. Both [Ley+09] and [ZPG17] heavily rely on heuristics and parameter tuning, which raises the question of whether these methods can be adapted to a broad range of document types.

In contrast to the similarity measurements mentioned before, word images can be represented holistically (cf Section 4.3). Retrieval becomes a nearest neighbor search if word images are represented as real-valued vectors with fixed dimensionality. This approach is efficient and comes with low computational cost. In this regard, distances are mainly used to assess similarities. A similarity is directly obtained by computing the distance between two word image representations. In order to obtain sufficient retrieval performance, appropriate similarity measures (i.e., distance metrics) are crucial. In the word spotting

literature, various distance metrics have been proposed, ranging from correlation-based distance [ZSH03], KL divergence [AD07] or Bray-Curtis dissimilarity [SF16]. A common approach for similarity assessment in lower dimensional vector spaces is often based on the Euclidean distance [RLM03; RFR03; KWD14]. However, with increasing dimensionality of the image descriptors, the Euclidean distance becomes less discriminative as large-valued vector components dominate the resulting distance value [PSM10]. This problem is addressed by normalizing the descriptors and considering the components' distribution. For this reason, measuring the angle between two vectors using the cosine similarity [BR11] has become the prevalent approach in recent word spotting methods [RL14; Alm+14b; RF15; SF15; SF17; Rot+17], consistently achieving state-of-the-art results.

4.4.2 *Learning-based Similarity*

In addition to the distance-based approaches, models are trained to assign similarities between the query and all candidates from the retrieval list. Therefore, the model learns to estimate similarities from annotated training data or corresponding query examples. Approaches operating on word-level have been proposed using statistical models [RLM03], SVMs [PR09; Alm+14a], or CNNs [Zho+16a]. On the other hand, sequence-based models, such as HMMs [RP12; Fis+12] or RNNs [Fri+12], are usually applied on character-level.

A SVM distinguishes between relevant and irrelevant word image representations regarding the query. In [PR09], query-by-example word spotting is performed as a fully supervised classification task. Therefore, the authors use annotated word images to train the SVM. This classification-based approach limits the retrieval to a lexicon of previously learned word classes. In contrast, in [Alm+14a], the exemplar SVM [MGE11] is directly trained to distinguish between the given query and randomly sampled examples from the document collection. Word image candidates are extracted in a sliding-windows approach and classified by the exemplar SVM into relevant or irrelevant, where similarity scores are assigned according to the distances to the hyperplane. In the subsequent step, top-ranked examples are used to train a second SVM and apply the sliding-window approach again in order to obtain the final retrieval list. This unsupervised approach does not require annotated training data and is not bound to a fixed-sized vocabulary.

Inspired by Siamese networks [CHL05], Zhong *et al.* [Zho+16a] use a CNN to estimate similarity scores between two given word images. The CNN is trained to predict relevant or irrelevant image pairs in a binary classification task. Therefore, the authors train the model using the CE and BCE loss functions. The word images are converted into gray-scaled images and rescaled to a uniform size of 40×100 pixels. Subsequently, pairs of gray-scaled images are presented to the network model as one image consisting of two color channels. This way, the forward pass through the network can be realized as a single pass, and the model has to decide whether the given two-channeled image belongs to the class relevant or irrelevant. The network architecture consists of two outputs, and the similarity score is computed by first applying the softmax and sigmoid functions, respectively, and then computing the average value. Intuitively, it is not entirely clear why the authors use two

loss functions to train the model and obtain the similarities. However, the results reported show that higher retrieval performance is achieved using both loss functions.

The methods above are based on decision boundaries that discriminate the feature space according to relevance. In contrast, generative approaches aim to model the generation of feature vectors by estimating statistical distributions. The estimated model evaluates similarity by computing the probability of generating image features. In this regard, a common approach is Bayes' theorem [Joy21]. In [RLM03], word images are described by a vocabulary of discretized word features. A joint probability distribution of features and word labels is estimated over an annotated training set of word images. Retrieval is performed by calculating the conditional probabilities for all vocabulary entries of the training set, given the discretized query features.

Sequence-based models are used to estimate statistics of feature vector sequences. In this regard, hidden Markov models (HMMs)s are the prevalent approach. The models are used for segmentation-based [RP12] and segmentation-free [RMF13] QbE word spotting where no annotations are available. The HMMs query models are estimated from a single visual query sample. In [RP12], word images are represented by a *shared* GMM estimated in an unsupervised manner. Only the state-dependent mixture and transition probabilities are obtained from a given query example at query time. Similarities between vector sequences are obtained using the Bhattacharyya similarity [Bha46] as the local measure in a DTW approach. Rothacker *et al.* [RMF13] use the BoF framework to represent word images (cf. Section 4.3). A query model is estimated from a sequence of BoF histograms extracted from the visual query example. Similarities are obtained by computing the best path probabilities using the Viterbi algorithm [Vit67]. The BoF-HMM approach is extended towards segmentation-free QbS word spotting [RF15]. A query word given as a textual string is modeled by concatenating *character* HMMs according to the query. These character models are estimated from annotated training data, and retrieval is performed similarly to [RMF13]. In [Fis+12], character HMMs are applied to perform QbS word spotting on line-level. In contrast to [RMF13; RF15], where similarities are computed between the word query and segmented word images, the character HMMs in [Fis+12] computes similarities between word queries and segmented word lines. The character HMMs are estimated using annotated text lines as training data. Similar to [RMF13], query models are obtained from concatenated character HMMs according to textual query strings and the text line scores are computed applying the Viterbi algorithm [Vit67]. Alongside the state decoding of character HMMs using the Viterbi algorithm, a modified version of the CTC token passing algorithm [Gra+09] is applied to perform QbS word spotting on handwritten text lines [Fri+12]. The token passing algorithm was initially presented for HMMs based speech recognition [YRT89] and later integrated into the CTC [Gra+09]. The algorithm's input consists of the query word given as a textual string and the text line represented as a sequence of character probabilities provided by the BLSTM (cf. Section 4.3.4). For retrieval, the algorithm's output value is used as the matching score, indicating whether a given text line contains the requested query word.

5

RELATED WORK

The approaches presented in the previous chapter are capable of performing either query-by-example or query-by-string word spotting. For this purpose, word images are transformed into a holistic representation or a sequence of feature vectors (cf Section 4.3). Similarities are obtained as inverted distances or aligned sequences (cf. Section 4.4.1). If a query is given as a textual string, retrieval is performed using sequential models such as character HMMs or BLSTMs (cf. Section 4.4.2). However, sequential approaches are computationally expensive, while holistic representations cannot handle queries given as textual strings. Word images and textual strings are represented in a common vector space to handle both query modalities. In this regard, word embeddings based on attributes provide an elegant solution (Section 5.1). While textual strings are directly converted into their corresponding embedding, word images are transformed from the visual domain to the respective string embedding. This transformation is usually achieved by either predicting individual attributes using an ensemble of SVMs or estimating the holistic attribute representation using a convolutional neural network (Section 5.2). Attribute-based word spotting provides three significant advantages. Firstly, computing distances between two vectors involves only tiny computational costs. This approach is computationally efficient. Secondly, the attribute representations can encode tens of thousands of word classes. This feature is crucial for document collections containing an extensive vocabulary. Finally, it is possible to perform OOV word spotting using an embedding to represent arbitrary query strings. After the word images are mapped into the attribute space, word spotting is performed in both retrieval scenarios as a nearest neighbor search, applying distance-based similarity measures. This technique allows for efficient and accurate word spotting in large datasets. If the attribute representation is binary, probabilities between individual attributes can be evaluated to assess similarities (Section 5.3).

5.1 TEXTUAL STRING EMBEDDING

Obtaining similarities between two domains requires both to be represented in the same vector space. A straightforward approach is to encode a predefined lexicon and train a model to predict one of K word classes. However, this restricts the retrieval approach to the word classes that are part of the lexicon (i.e., vocabulary), while OOV retrieval remains impossible. Hence, an essential requirement for word image representations is the allowance for querying word strings unknown from training. Therefore, decomposing the word classes into semantic units, such as characters, could extend the lexicon-based approach. This way, individual characters within the word image would be predicted in-

stead of word classes from the predefined lexicon. For example, the model will predict the characters $\{e, i, l, n, s, t\}$ for an image containing the word *silent*. This approach would effectively result in a multi-label classification task (cf. Section 2.1.2). However, predicting the characters $\{e, i, l, n, s, t\}$ is orderless, and two different words, such as *silent* and *listen*, are not distinguishable, which is a significant drawback. For that reason, taking the order of characters into account is a crucial requirement. This requirement, for example, is fully met by sequence-based models such as HMMs [Fis+12] or BLSTMs [Fri+12] where characters are predicted according to their order.

Another approach is based on *n-gram* models where textual strings are represented as histograms of n-grams [Ald+13]. The word transcription is decomposed into uni-, bi-, and trigrams. Neighborhood information is integrated by using higher n-grams to encode consecutive characters. The set of considered n-grams is generated from the training set where the textual information is available. A word string descriptor is obtained by counting the occurrences of each n-gram and normalizing the resulting histogram using the $L2$-norm. The word images are represented by BoF descriptors with integrated spatial information by applying a spatial pyramid partitioning [LSP06]. Both textual and visual representations are embedded into a common subspace by applying the LSI algorithm [Dee+90]. Finally, query-by-string word spotting is performed in this subspace. A similar approach based on textual string embedding is proposed in [GRK17]. However, instead of using LSA, word images and textual strings are embedded into a latent space using CNNs. The approach is based on *deep metric learning* [KB19], and the goal is to obtain a word embedding, such that distances between embedded word images and strings reflect their *Levenshtein* [Lev66] distance.

Inspired by the BoC string kernels [Lod+00; LEN02], a very influential embedding approach based on histograms of character occurrences is proposed in [Alm+13; Alm+14b]. The string embedding is called PHOC and embeds textual strings into a n dimensional binary attribute space. To avoid ambiguity between words consisting of the same characters, such as *stress* and *rest*, strings are decomposed in a pyramidal scheme. The basic idea of the PHOC representation is to encode if a particular character occurs in a particular region of a textual string. Figure 18 visualize a decomposition according the PHOC principal of the word string *captain*. The PHOC algorithm splits the string according to predefined pyramid levels. The quantity of binary character histograms extracted corresponds to the level indices. That is, level 1 will result in 1 histogram, level 2 in 2 histograms, and so on. Whether a character falls into a particular split is decided according to the occupancy between the character and the respective split [Alm+13]. A character is associated with a split if the overlap is $\geq 50\%$. Within each histogram, the value of an attribute is set to 1 if the character associated with the attribute occurs at least once in the corresponding split. The dimensionality of a histogram is defined by the quantity of attributes included. For example, in [Alm+13], the authors use all 26 lower-case characters of the Latin alphabet and 10 digits as the attribute set, leading to 36 dimensional histograms. However, the choice of semantic units heavily depends on the language of document collections and the

application of interest. For example, the Latin alphabet is sufficient to represent English documents. However, this alphabet is insufficient to present words from French, Spanish, or German texts, and additional characters are required. Finally, the resulting PHOC representation is obtained by concatenating the histograms from all splits into a single vector of binary values. For the example in Figure 18, if the dimensionality of a histogram consists of 26 letters plus 10 digits and the string is split at the pyramid levels 1, 2, and 3, the resulting PHOC vector will have a dimensionality of $(26 + 10) \times (1 + 2 + 3) = 42$.

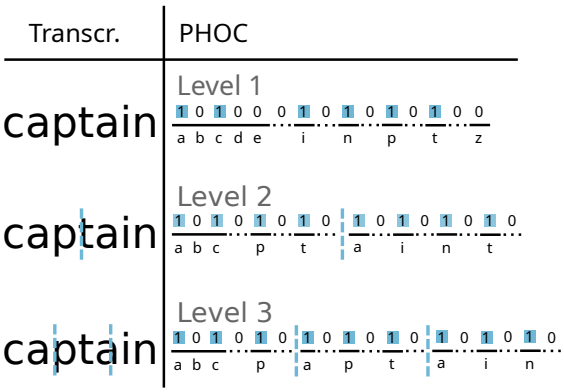


Figure 18: Visualization of the construction principal of the pyramidal histogram of characters (PHOC) [Alm+13]. The example shows the resulting PHOC vectors on level 1, 2 and 3 for the word string *captain*. For each split in the respective level, the binary vector indicates whether a given character is present or not. The final PHOC vector is obtained by concatenating all binary vectors from all splits.

In a similar approach, histograms of character counts are computed to build a string descriptor referred to as *spatial pyramid of characters* (SPOC) [RGP15]. In contrast to the PHOC representation, the actual quantity of a character in a particular split is considered. For example, if a character occurs twice in the left split of level 2, the corresponding attribute is set to the value 2. Computing the occupancy is not required, as characters that are located between two splits are counted according to their fraction. Considering the example from Figure 18, the attribute for character t on level 2 would have a value 0.5 in both splits. The final SPOC representation is obtained by concatenating all histograms of character counts into a single real-valued vector that, subsequently, can be normalized, as noted by the authors. In this regard, the PHOC can be seen as a binary, and the SPOC as a continuous attributes representation (cf Chap. 3).

A rather exotic word embedding based on the *discrete Cosine transform* (DCT) is presented in [WB16]. Given a word string, individual characters are encoded as one-hot vectors and stacked into an indicator matrix of shape $L \times N$, where L is the alphabet size and N is the length of a given word. The DCT is applied along each row, and the three largest values are concatenated into the final descriptor referred to as *discrete Cosine transform of words* (DCToW). The sequential order of the characters is encoded by the sequence of values obtained from the DCT. Similar to SPOC, the DCToW descriptor can be seen as a continuous attributes representation. In contrast to the PHOC and SPOC representations, the encoded string is not directly interpretable when represented by the DCToW.

5.2 WORD IMAGE EMBEDDING

Textual strings and word images are embedded into a common subspace, mainly following two approaches. One approach is based on a separate representation of the textual and visual domains. Both domains are embedded into the common subspace, applying a predefined transformation that minimizes the distances between word image and textual string descriptors [Ald+13; GRK17]. Other approaches, such as PHOC, SPOC, or DCToW, determine the representations for textual strings algorithmically in a deterministic way. The resulting attribute vectors represent the target descriptors, and additional subspace transformations are not required. Visual models are designed to estimate the string representation directly, and word spotting is performed in the target space using distance-based similarity measures [Alm+13; Alm+14b; SF16; WB16; SRF17; KDJ18].

Textual and visual descriptors are embedded into a latent subspace to perform query-by-string word spotting [Ald+13]. The embedding is obtained by applying the LSA algorithm [Dee+90]. As described in the previous section, text strings are represented by histograms of n-grams. Word image descriptors, on the other hand, are computed using the BoF framework [OD11] based on SIFT features [Low04] and a pyramidal partitioning of the word image [LSP06]. The space transformation matrix is obtained from corresponding visual and textual descriptors by applying the LSA method. Finally, distance-based word spotting is performed by applying the space transformation matrix to all word images from the document collection and the textual queries.

The approach from [Ald+13] depends on handcrafted descriptors and requires a subsequent embedding of both domains. A method based on CNNs that jointly optimizes the embedding of textual strings and word images is presented in [GRK17]. Inspired by deep metric learning [KB19], pairs of images and their corresponding transcriptions are forwarded through two separate networks. The basic idea is to obtain a latent embedding such that distances between strings and images correspond to their Levenshtein distance [Lev66]. Therefore, a textual string is encoded as a 27×24 matrix where the columns correspond to a predefined alphabet of 27 signs[1] and the number rows represent the maximum word length of 24. The authors use a shallow CNN to embed textual strings, consisting of a single convolutional layer with 256 channels and a receptive field size of 27×1. In contrast, word images are processed by a deep CNN consisting of 4 convolutional and 2 fully connected layers. The network architecture is adapted from the field of scene text recognition [Jad+14].

If string representations, such as PHOC or SPOC, are computed deterministically, word images are directly mapped to the respective target representation. An ensemble of SVMs is used to predict the binary PHOC vector, referred to as *AttributeSVM* [Alm+14b]. One SVM model is trained to predict one attribute in the respective representation. Subsequently, this AttributeSVM is used to predict the presence of attributes given a word image. Instead of using the predicted binary class information, the distance between a

1 26 characters from the Latin alphabet and 1 additional filler sign

given Fisher vector and the hyperplane of the respective SVM is used. The distance is projected into a range between 0 and 1 using a Platts scaling [Pla99] approach based on extreme value theory [Sch+12]. The word images are encoded as Fisher vectors enriched with spatial information on top of densely sampled SIFT descriptors. Similar to [Alm+14b], a structured SVM framework [NL11] is applied to estimate the continuous attributes of the SPOC representation in [RGP15]. Inspired by the ranking SVM from [Joa02], the learning objective is modified towards a smooth upper bound compared to the hinge loss, which is typically used. The proposed structured SVM approach directly predicts the continuous attributes of the SPOC representation. Compared to SIFT descriptors and Fisher vectors, more sophisticated feature representations are provided by CNN based approaches. Holistic word image features are extracted following a deep feature embedding approach [KDJ16]. The CNN model is trained to classify synthetically generated word images. Subsequently, the output of the penultimate FC layer is utilized as the word image descriptor. Finally, the authors apply the AttributeSVM approach from [Alm+14b] to map the word image descriptors and PHOC representations.

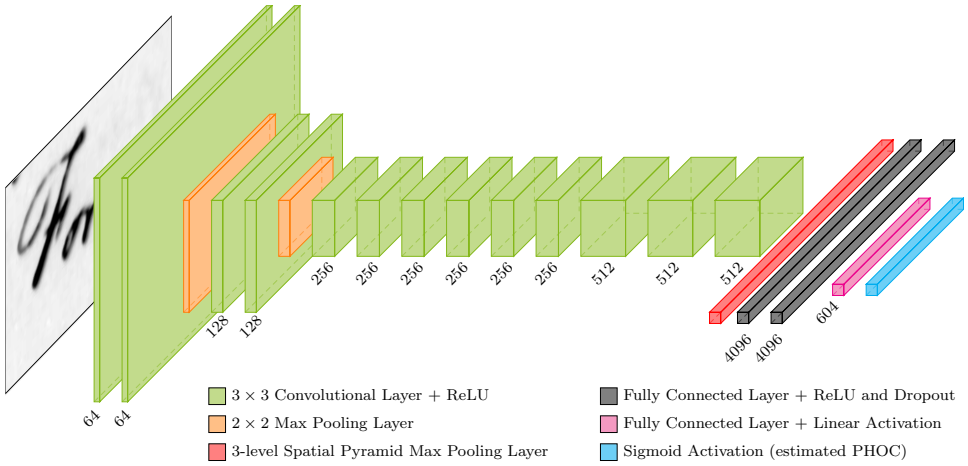


Figure 19: Visualization of the PHOCNet architecture as presented in [SF16]. The network is based on the VGGNet [SZ15] and consist of 13 convolutional layers (green) with an increasing number of filter kernels (i.e., channels), while the input image is downsampled twice using max-pooling (orange). A spatial pyramid pooling (red) is applied to the output of the last convolutional layer in order to process arbitrarily sized images. The output of the spatial pyramid pooling is processed by a multi layer perceptron with two hidden layers (black) and, finally, mapped to the PHOC representation (blue) using a sigmoid activation function.

The SVM-based approaches from [Alm+14b; RGP15; KDJ16] require a disjoint optimization between the feature extraction and attribute prediction. For that reason, Sudholt *et al.* propose a CNN architecture called PHOCNet [SF16]. The PHOCNet provides a solution to estimate PHOC representations given their respective word images directly. A significant advantage over the approaches from [Alm+14b; RGP15; KDJ16] is that the feature extraction and attribute prediction are optimized jointly and, thus, tuned to each other. As displayed in Figure 19, the architecture of the PHOCNet is based on the popular VGGNet [SZ15]. The network is adapted to process handwritten word images by

incorporating a *SPP* [He+15b] after the last convolutional layer. Using the SPP enables the model to process arbitrarily large image sizes. The sigmoid function is applied to the network's output to estimate the binary PHOC attributes, and the BCE loss is used to optimize the model parameters. Sudholt *et al.* [SF17] propose a modified version of the PHOCNet. The pyramid pooling is adjusted to perform a partitioning exclusively along the writing (i.e., horizontal) direction referred to as *TPP*. Furthermore, the authors discard the sigmoid function after the last fully connected layer and apply the Cosine loss to optimize the network parameters. Using a linear output enables the CNN model to estimate continuous attribute representations such as the SPOC [SF17]. Similarly, a CNN model that estimates the DCToW representations is proposed in [WB16]. The network is based on the widely used ResNet architecture [He+16] and trained using the Cosine embedding loss. The objective of the Cosine embedding loss is to minimize or maximize the cosine distance between the network estimate and a given string representation. The correspondence and non-correspondence are indicated by the label values 1 and -1, respectively. On the other hand, the authors in [SF17] train the network by only applying the corresponding case of the Cosine embedding loss, which results in the Cosine loss [Cho16]. However, the significant differences between the CNN models from [SF17] and [WB16] are the network architecture and the pooling strategy after the last convolutional layer. Moreover, the approaches from [SF17] and [WB16] use different training protocols and, thus, are difficult to compare. In this regard, Krishnan *et al.* [KJ19] evaluate the performance influence of the network architecture and pooling strategy using the same protocol. The results reported in [KDJ16] show a significant performance increase of 8 percentage points if the predecessor architecture is replaced by the ResNet [He+16]. Furthermore, compared to a global pooling as used in [WB16], an impressive improvement of 12.59 percentage points is achieved using the RoI pooling [KJ19] and even 16.05 percentage points if a temporal pyramid pooling [SF17] is applied. The evaluation in [KJ19] demonstrates the enormous impact of the architectural network design and the pyramidal pooling strategy. In [KDJ23], the authors present an improved approach. The "two-stream" architecture from [KJ19] is modified using one CNN model for both embedding streams. This modification significantly improves the retrieval performance while the ResNet architecture remains unchanged.

5.3 ZERO-SHOT CLASSIFICATION WITH DIRECT ATTRIBUTE PREDICTION

Class representations based on attributes are used to predict categories that are not part of the training set known as zero-shot learning [FZ07; LEB08; Far+09; PG11] (cf Chap. 3). In this regard, the influential work of Lampert *et al.* [LNH09] present the prediction of unseen classes based on probabilistic attribute similarities. This approach is highly relevant for the methodology presented in the ensuing Chap. 6, and, thus, is discussed in more detail in the following. The authors exploit the fact that meaningful high-level concepts, such as attributes, transcend class boundaries. Instead of creating a separate set of training

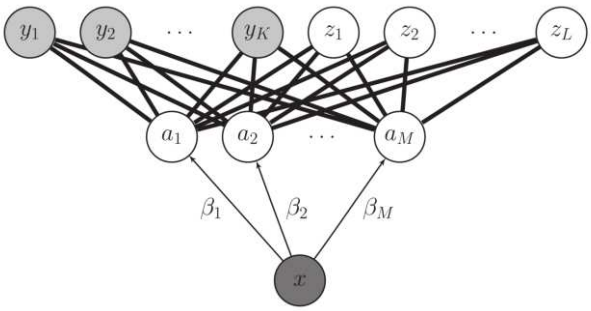


Figure 20: Visualization of the graphical representation for the DAP approach as presented in [LNH09]. The model parameters $\{\beta_1, \beta_2, ..., \beta_M\}$ are learned by providing a set of *seen* training classes $\{y_1, y_2, ..., y_K\}$, which are then used to estimate the attributes $\{a_1, a_2, ..., a_M\}$. These attributes are subsequently used for zero-shot recognition of an input image $\mathbf{x}$. Finally, the correct class is determined, based on the estimated attribute vector ${}^z\mathbf{a}$.

samples for each class from the test set, existing training data describes images by their properties shared between training and test samples. For example, the attribute *striped* is learned from images of zebras, bees, and tigers. The attribute *yellow* would not refer to the class zebras. However, images of bees and tigers would still provide helpful training samples [LNH14]. The authors argue that the possibility of obtaining knowledge about attributes from different object classes and, vice versa, using each attribute to detect many object classes makes the proposed learning method statistically more efficient. The approach is based on the concept of transferring knowledge from a training set $\mathbf{y} = \{y_1, y_2, ..., y_K\}$ towards a test set $\mathbf{z} = \{z_1, z_2, ..., z_L\}$, where K and L is the number of unique classes respectively. Therefore, an intermediate layer $\mathbf{a} = \{a_1, a_2, ..., a_M\}$ is introduced consisting of M binary attributes to share the information between the classes in $\mathbf{y}$ and $\mathbf{z}$. Each class from $\mathbf{y}$ and $\mathbf{z}$ is represented by a unique attribute vector (i.e., ${}^y\mathbf{a}$ or ${}^z\mathbf{a}$). The per-attribute parameters $B = \{\beta_1, \beta_2, ..., \beta_M\}$ are obtained applying a supervised training. Subsequently, zero-shot classification is performed estimating the attribute representation ${}^z\mathbf{a}$ of an unseen class z referred to as *DAP* [LNH09]. Figure 20 visualize the DAP approach as a graphical representations.

The goal of the DAP approach is to assign the best output class $\hat{z}$ from all test classes $\mathbf{z}$ given the input image $\mathbf{x}$ which is obtained by using the MAP prediction:

$$\hat{z} \;=\; \underset{z \in \mathbf{z}}{\operatorname{argmax}} \; p\left(z \mid \mathbf{x}\right) \;. \tag{35}$$

This output class is obtained by first estimating the attribute probabilities from a test image $p\left(\mathbf{a} \mid \mathbf{x}\right)$ and subsequently evaluating the probability of the test class given the estimated attributes $p\left(z \mid \mathbf{a}\right)$. Formally the relationship for calculating the posterior of a test class is defined as:

$$p\left(z \mid \mathbf{x}\right) = \sum_{a \in \{0,1\}^M} p\left(z \mid a\right) \; p\left(a \mid \mathbf{x}\right) \;. \tag{36}$$

The estimates of $p(\mathbf{a} \mid \mathbf{x})$ are provided by a trained classifier. Therefore, the authors train one nonlinear SVM model per attribute similar to the AttributeSVM used in [Alm+14b]. Attribute probabilities are obtained by fitting a sigmoid curve using 10% of the training data following the Platt scaling approach [Pla99]. The images are represented by BoF histograms on top of SIFT descriptors and a pyramidal version of the HOG descriptor referred to as pyramidal histogram of gradients (PHOG) [BZM07]. For every test class z it is assumed that an attribute ${}^{z}a$ is predicted in a deterministic way, such that $p(a \mid z) = 1$ only if $a = {}^{z}a$ and 0 otherwise. The probability of the test class z can now be recovered by applying the Bayes' rule $p(z \mid {}^{z}\mathbf{a}) = \frac{p(z)}{p({}^{z}\mathbf{a})} p({}^{z}\mathbf{a} \mid z)$ and rewriting Equation 36. Finally, Equation 36 is plugged into Equation 35 which defines the DAP method as

$$\hat{z} \;=\; \underset{z \in \mathcal{Z}}{\operatorname{argmax}} \; p(z \mid \mathbf{x}) \;=\; \frac{p(z)}{p({}^{z}\mathbf{a})} \prod_{m=1}^{M} p({}^{z}a_m \mid \mathbf{x}) \;. \tag{37}$$

The estimates $p({}^{z}a_m \mid \mathbf{x})$ are evaluated as $p(a_m \mid \mathbf{x})$ if the value of attribute ${}^{z}a_m$ is 1, and $1 - p(a_m \mid \mathbf{x})$ otherwise. The prediction of an unseen class $\hat{z}$ only depends on the estimates $p({}^{z}a_m \mid \mathbf{x})$ and a prior $\frac{p(z)}{p({}^{z}\mathbf{a})}$, while the authors note that in practice the prior can be ignored [LNH14].

6

WORD SPOTTING WITH ATTRIBUTE PROBABILITIES

In the field of word spotting, textual strings represented as attribute vectors consistently achieve state-of-the-art results [WB16; SF18; KDJ23]. The excellent performance is mainly due to the flexible PHOC representation, which can be adapted to any lexica [Alm+14b]. This chapter introduces a new similarity measure, called *PRM*, that utilizes the PHOC representation to evaluate similarities between a query and all candidates from a database. This similarity measure assesses individual probabilities between corresponding attributes given two binary-valued representations. The resulting similarity is then determined by computing the total probability. Section 6.1 presents the mathematical fundamentals of the probabilistic retrieval model. Utilizing the PHOC to represent word images and textual strings enables the proposed method to perform segmentation-based QbS and QbE word spotting. In QbS, the attribute representations are obtained from textual strings by applying the PHOC algorithm. In the case of QbE, the PHOC attributes are estimated using a convolutional neural network based on the popular PHOCNet [SF18] architecture. Optimal parameters for the neural network are obtained by applying a suitable loss function.

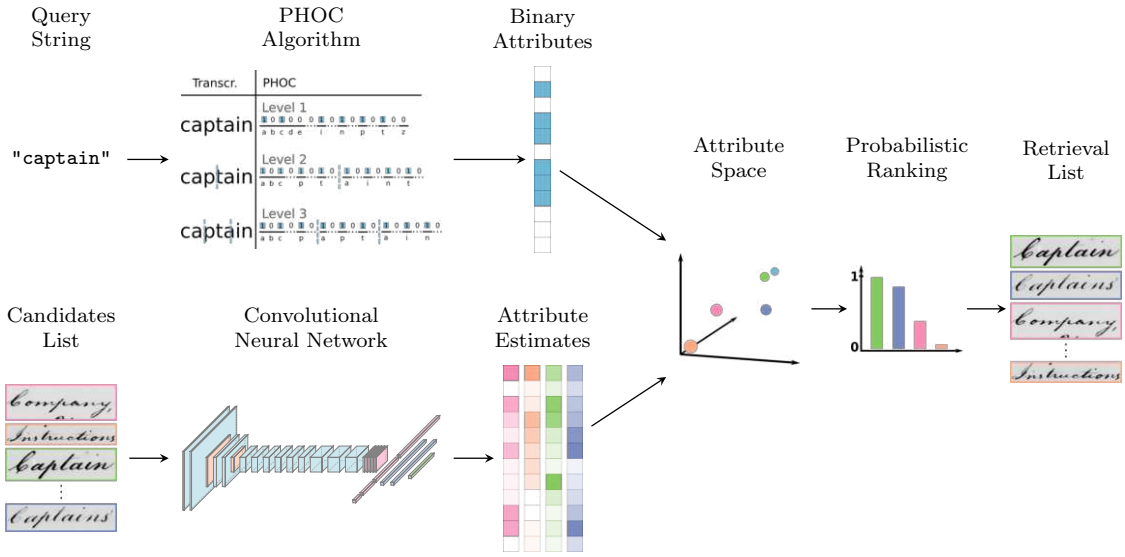


Figure 21: The figure shows all steps of the proposed method for a QbS retrieval scenario. The user defined query string is given in the upper left corner. The query is transformed into an binary attribute vector by applying the PHOC algorithm. The lower left corner shows the list of candidates (i.e., word images) from a given document collection. These images are transformed through the convolutional neural network in order to obtain the attribute estimates. Both, the query and the candidates are now represented by vectors in the attribute space. Finally, retrieval is performed in the PRM framework resulting in a retrieval list according their probabilities (i.e., probabilistic ranking).

Section 6.2 presents the derivation of the appropriate loss function specifically designed to train the network model for estimating binary attribute probabilities. An overview of the word spotting framework presented in this chapter is shown in Figure 21. The different processing steps for the query-by-string retrieval are visualized from left to right. The *query string* "captain", shown in the upper left corner, is transformed into its corresponding vector representation of *binary attributes* by applying the PHOC algorithm. In the lower-left corner, the *candidates list* of word images is forwarded through the *convolutional neural network* to obtain the *attribute estimates*. Both the query and all candidates are now represented in the *attribute space* by vectors consisting of individual attribute probabilities. Subsequently, the similarities between the query and all candidates are evaluated using the *probabilistic retrieval model*. Finally, the similarities are sorted in descending order based on their probability values to obtain the *retrieval list*.

6.1 PROBABILISTIC RETRIEVAL MODEL

To evaluate the similarity between word image $\mathbf{x}$ and string query q, the posterior $p(q|\mathbf{x})$ is required. The PRM adapts the DAP approach [LNH14] and evaluates the posterior of a query given a word image in the following way:

$$p(q|\mathbf{x}) = \sum_{i=1}^{D} p(q|a_i)\ p(a_i|\mathbf{x})\ , \tag{38}$$

where D is the dimensionality of attribute vector $\mathbf{a} \in \{0,1\}^D$ consisting of binary values. The PRM is based on the assumption that the individual attribute probabilities are conditionally independent of each other. Consequently, the posterior $p(q|\mathbf{x})$ results from a summation of the individual attribute probabilities. Furthermore, the PRM assumes that the attribute vector $\mathbf{a}$ for every string query q is given in a deterministic way according to the construction principle of the PHOC [Alm+14b] algorithm. Therefore, the individual probabilities of attribute vector ${}^q\mathbf{a}$ for query q are obtained from the following condition:

$${}^qa_i \ = \ p(A_i = 1 \mid q) \ = \ \begin{cases} 1 & if\ PHOC(q)_i = 1 \\ 0 & otherwise \end{cases} .$$

It is important to note that $p(\mathbf{a}|q)$ is nonzero only for a single attribute vector, namely the PHOC representation of query q. The following condition expresses this relation:

$$p(\mathbf{a} \mid q) \ = \ \begin{cases} 1 & if\ \mathbf{a} = {}^q\mathbf{a} \\ 0 & otherwise \end{cases} .$$

In the next step, the probability of attribute vector $\mathbf{a}$ is obtained by rewriting the posterior $p(q|\mathbf{a})$ using the Bayes' rule:

$$p(q|\mathbf{a}) = \frac{p(q)}{p(\mathbf{a})}\, p(\mathbf{a}|q)\ . \tag{39}$$

By applying the Bayes' rule, the relationship between $p(q|\mathbf{a})$ and $p(\mathbf{a}|q)$ can be obtained and subsequently plugged into Equation 38. Furthermore, considering the previously defined condition that $p(\mathbf{a}|q)$ is 1 only if $\mathbf{a}$ is the PHOC representation of query q (e. g. , ${}^{q}\mathbf{a}$), Equation 38 can be rewritten in vector notation in the following way:

$$p(q|\mathbf{x}) = \frac{p(q)}{p({}^{q}\mathbf{a})}\, p({}^{q}\mathbf{a}|\mathbf{x})\ . \tag{40}$$

In order to evaluate this equation, in addition to the probability of predicting ${}^{q}\mathbf{a}$ from $\mathbf{x}$, the prior probabilities $p(q)$ of the query and the attribute representation thereof, $p({}^{q}\mathbf{a})$, are principally required [Rus+18]. However, in the absence of more specific knowledge, it is reasonable to assume identical priors for the queries and the attribute vectors, which allows ignoring the quotient $\frac{p(q)}{p({}^{q}\mathbf{a})}$ in Equation 40. Additionally, assuming uniform priors prevents any possibly wrong assumptions, as it is not apparent how a prior over the potentially open set of string queries could be defined [Rus+18]. By ignoring the priors, Equation 40 boils down to

$$p(q|\mathbf{x}) = p({}^{q}\mathbf{a}|\mathbf{x}), \tag{41}$$

where the probability of query q given word image $\mathbf{x}$ can be obtained by evaluating the probability of the PHOC representation ${}^{q}\mathbf{a}$ of the query given the word image. Finally, the probability of the i-th attribute being present in word image $\mathbf{x}$ is provided by the neural network as the estimate $\hat{a}_i = p(A_i = 1 \mid \mathbf{x})$, where $\hat{a} \in \mathbb{R}$ is a scalar in the range $(0, 1)$. As each attribute is considered to be a binary random variable and the behavior is described by the Bernoulli distribution with conditional independence among attributes, the PRM metric is defined as

$$p\left(q \mid \mathbf{x}\right) \;=\; p\left({}^{q}\mathbf{a} \mid \hat{\mathbf{a}}\right) \;=\; \prod_{i=1}^{D} \hat{a}_i{}^{\,a_i} \cdot (1 - \hat{a}_i)^{(1-a_i)}\ . \tag{42}$$

The similarity between query q and word image $\mathbf{x}$ is evaluated as a product of Bernoulli probabilities by obtaining the query attribute vector ${}^{q}\mathbf{a}$ via the PHOC algorithm and estimating $\hat{\mathbf{a}}$ through a neural network. At this point, the decision whether image $\mathbf{x}$ is relevant for a query q only depends on the correct estimation of $\hat{\mathbf{a}}$ for a given word image, as ${}^{q}\mathbf{a}$ is obtained in a deterministic way.

6.2 ESTIMATING ATTRIBUTE PROBABILITIES

Since the PRM is based on probability estimates, the best ranking is achieved by estimating the correct attribute probabilities defined by the corresponding PHOC representation. This section presents the steps for obtaining a statistical model that provides optimal estimates for binary attribute probabilities following a Bernoulli distribution. Section 6.2.1 introduces the *generelized linear models* (GLMs) framework with an example of obtaining the appropriate loss function for a given distribution. Afterward, the GLM is applied to the PRM in Section 6.2.2, deriving the loss function used to train the neural network to estimate the attribute probabilities given a word image.

6.2.1 *Generalized Linear Models*

In statistics, the GLMs provides a very broad and popular family for statistical analysis [Agr15]. The umbrella term GLM encompasses many regression models by allowing the linear model to be related to the response variable via a so-called link function. The GLM is a flexible generalization of linear regression which was originally formulated by Nelder *et al.* [NW72] as a way of unifying various statistical regression models, including linear, logistic and Poission regression. In general, regression is a approach for modeling the relationship between a dependent variable Y and a independent variable $\mathbf{X}$ to obtain the estimate of a conditional expected value $\hat{\mathbb{E}}[Y|\mathbf{X} = \mathbf{x}]$ [FWS06]. The GLM framework is based on two important assumptions: First, Y follows a distribution from the natural exponential family, and second, the expected value $\mathbb{E}[Y]$ depends on a linear transformation of $\mathbf{X}$. Therefore, the GLM consists of three elements, namely the *response variable distribution*, a *linear predictor* and *link function*. The key assumption in the GLM is that the response variable distribution is a member of the exponential family of distributions, which includes the normal, binomial, Poisson, gamma and others [MPV06]. A distribution belongs to the exponential family if its PDF (or PMF for discrete distributions) can be written in the form:

$$f_{exp}(\mathbf{x}|\boldsymbol{\theta}) \;=\; h(\mathbf{x}) \cdot \; exp\Big[\; \boldsymbol{\theta}^T \cdot \; T(\mathbf{x}) - A(\boldsymbol{\theta}) \;\Big] \;, \tag{43}$$

where $\boldsymbol{\theta}$ is called the natural parameter of the distribution and $\mathbf{x}$ is a realization of $\mathbf{X}$. The function $h(\mathbf{x})$ is called the base measure, $T(\mathbf{x})$ is the sufficient statistics, and $A(\boldsymbol{\theta})$ denotes the log-partition function. The linear combination of $\mathbf{X}$ can be expressed as an inner product that is known as the *linear predictor* in the context of GLMs:

$$\eta(\mathbf{x}) \;=\; \mathbf{w}^T\mathbf{x} \;. \tag{44}$$

The *link* between the linear predictor $\eta(\mathbf{x})$ and the expected value $\hat{\mathbb{E}}[Y|\mathbf{X} = \mathbf{x}]$ of the response variable distribution is provided by the link function $g(\cdot)$ in the following way:

$$\eta(\mathbf{x}) \;=\; \mathbf{w}^T\mathbf{x} \;=\; g(\hat{\mathbb{E}}[Y|\mathbf{X} = \mathbf{x}]) \;. \tag{45}$$

The conditional expected value can be recovered by inverting the canonical link function $g(\cdot)$ in order to get

$$g^{-1}\left(\eta(\mathbf{x})\right) \;=\; g^{-1}\left(\mathbf{w}^T\mathbf{x}\right) \;=\; \hat{\mathbb{E}}[Y|\mathbf{X}=\mathbf{x}]\;. \tag{46}$$

The expected value is now expressed as a linear transformation of the weight parameters $\mathbf{w}$ and the input variable $\mathbf{x}$ that is transformed by the inversion of the canonical link function $g^{-1}(\cdot)$. Each exponential family distribution has a *canonical link function* for which a number of statistical characteristics can be shown [Agr15]. In general, any function can serve as the canonical link function as long as its inverse is defined, and the linear predictions of $\eta(\mathbf{x})$ are projected onto a range of values suitable for the distribution of Y.

A canonical link function is obtained by rearranging the PDF into the standard exponential family shown in Equation 43. The rearrangement below demonstrates how to obtain the canonical link function for the response variable Y, which follows a normal distribution. The PDF of a normal distribution is defined as

$$f_{\mathcal{N}}\left(y \;\middle|\; \mu, \sigma^2\right) \;=\; \frac{1}{\sqrt{2\pi\sigma^2}}\;\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)\;. \tag{47}$$

As the GLM does not estimate the variance σ^2, it must be defined by the user. Typically, it is assumed that $f_{\mathcal{N}}$ has a unit variance, so σ^2 is set to 1. Using a normal distribution with a unit variance, $f_{\mathcal{N}}$ can be rearranged in the following way:

$$\begin{aligned} f_{\mathcal{N}}\left(y \;\middle|\; \mu, \sigma^2=1\right) \;&=\; \frac{1}{\sqrt{2\pi}}\;\exp\left(-\frac{(y-\mu)^2}{2}\right) \\ &=\; \frac{1}{\sqrt{2\pi}}\;\exp\left(-\frac{1}{2}\left(y^2+\mu^2-2y\mu\right)\right) \\ &=\; \frac{1}{\sqrt{2\pi}}\;\exp\left(-\frac{y^2}{2}-\frac{\mu^2}{2}+y\mu\right) \\ &=\; \frac{1}{\sqrt{2\pi}}\;\exp\left(-\frac{y^2}{2}\right)\;\cdot\;\exp\left(\mu y-\frac{\mu^2}{2}\right)\;. \end{aligned} \tag{48}$$

Now, the resulting form in Equation 48 corresponds to the standard form as shown in Equation 43, and the elements of the GLM mentioned above are directly assigned as follows

$$\begin{aligned} h(\mathbf{x}) \;&=\; \frac{1}{\sqrt{2\pi}}\;\exp\left(-\frac{y^2}{2}\right) \\ \boldsymbol{\theta}^T \;&=\; \mu \\ T(\mathbf{x}) \;&=\; y \\ A(\boldsymbol{\theta}) \;&=\; \frac{\mu^2}{2}\;. \end{aligned}$$

The canonical link function $g_{\mathcal{N}}$ for the variable Y that follows a normal distribution with a unit variance is defined by the identity function $\boldsymbol{\theta}^T = \mu$. From this, it follows that the inverse of the canonical link function $g_{\mathcal{N}}^{-1}$ is likewise the identity function:

$$g_{\mathcal{N}}(\mu) \;=\; \mu \;=\; g_{\mathcal{N}}^{-1}(\mu) \;.$$

Considering the transformation from Equation 45 to Equation 46 leads to the linear predictor $\eta(\mathbf{x})$ that directly estimates the expected value

$$\eta(\mathbf{x}) \;=\; \hat{\mathbb{E}}_{\mathcal{N}}\left[\, Y \mid \mathbf{X} = \mathbf{x} \,\right] = \mu \;. \tag{49}$$

The linear predictor model requires the optimal parameters $\mathbf{w}$ to estimate the expected value $\hat{\mathbb{E}}_{\mathcal{N}}$. Consequently, the GLM is trained in a supervised fashion, where the optimal parameters are obtained by applying the MLE. Training a model in a supervised setup requires the annotated dataset $\mathbf{S}$ that is defined as

$$\mathbf{S} = \left\{ \left(\mathbf{x}^{(i)}, y^{(i)} \right) \right\}_{i=1}^{\mathsf{N_S}} \;. \tag{50}$$

Here, $\mathbf{x}^{(i)}$ denotes the i-th independent variable (or *training sample*), y represents the dependent variable (or *label*) and $\mathsf{N_S}$ is the number of training samples. The likelihood of the model given the data is defined as

$$L\left(\mathbf{w} \mid \mathbf{S}\right) = \prod_{i=1}^{\mathsf{N_S}} f_{\mathcal{N}}\left(y^{(i)} \;\middle|\; \mu = \hat{y}^{(i)}, \sigma^2 = 1 \right) \;. \tag{51}$$

Analog to the previous example, it is assumed that all y follow a normal distribution with unit variance (i.e., $\sigma^2 = 1$). The expected values of the normal distribution μ are obtained from the model output $\hat{y}$ and are defined as

$$\hat{\mathbb{E}}_{\mathcal{N}}\left[\, Y \mid \mathbf{X} = \mathbf{x} \,\right] = \mathbf{w}^T \mathbf{x} = \hat{y}\left(\mathbf{x} \mid \mathbf{w}\right) \;. \tag{52}$$

For the sake of simplicity, the expression $\hat{y}(\mathbf{x} \mid \mathbf{w})$ is abbreviated to $\hat{y}$ in Equation 51. The optimal model parameters $\hat{\mathbf{w}}$ now need to be chosen such that $L\left(\mathbf{w} \mid \mathbf{S}\right)$ is maximized. A standard approach for solving MLE problems is to minimize the negative logarithm of the likelihood

$$\hat{\mathbf{w}} \;=\; \underset{\mathbf{w}}{\operatorname{argmax}} \; L\left(\mathbf{w} \mid \mathbf{S}\right) = \underset{\mathbf{w}}{\operatorname{argmin}} \; -\log L\left(\mathbf{w} \mid \mathbf{S}\right) \;, \tag{53}$$

which is also known as NLL. The NLL of the normal distribution is obtained by applying the negative of the natural logarithm to the PDF from Equation 51, which results in

$$\begin{aligned} -\log\ L(\mathbf{w} \mid \mathbf{S}) &= -\log \prod_{i=1}^{\mathsf{N_S}} f_{\mathcal{N}}\left(y^{(i)} \,\middle|\, \hat{y}^{(i)}, \sigma^2 = 1\right) \\ &= -\log \prod_{i=1}^{\mathsf{N_S}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y^{(i)} - \hat{y}^{(i)})^2}{2}\right) \\ &= -\sum_{i=1}^{\mathsf{N_S}} \log \frac{1}{\sqrt{2\pi}} \cdot \exp\left(-\frac{(y^{(i)} - \hat{y}^{(i)})^2}{2}\right) \\ &= -\sum_{i=1}^{\mathsf{N_S}} \log\left(\frac{1}{\sqrt{2\pi}}\right) - \sum_{i=1}^{\mathsf{N_S}} \log\left(\exp\left(-\frac{(y^{(i)} - \hat{y}^{(i)})^2}{2}\right)\right) \\ &= -\frac{\mathsf{N_S}}{2} \log\left(\frac{1}{\sqrt{2\pi}}\right) + \frac{1}{2} \sum_{i=1}^{\mathsf{N_S}} (y^{(i)} - \hat{y}^{(i)})^2 \ . \end{aligned} \tag{54}$$

Analogous to Equation 48, the variance σ^2 is set to 1. This results in the term $-\frac{\mathsf{N_S}}{2} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)$ being transformed into a constant value, which is then ignored in the subsequent optimization. As a consequence, the NLL from Equation 54 boils down to

$$\begin{aligned} \hat{\mathbf{w}} &= \operatorname*{argmin}_{\mathbf{w}} \quad -\log\ L(\mathbf{w} \mid \mathbf{S}) \\ &= \operatorname*{argmin}_{\mathbf{w}} \quad \frac{1}{2} \sum_{i=1}^{\mathsf{N_S}} \left(y^{(i)} - \hat{y}^{(i)}\right)^2 \ . \end{aligned} \tag{55}$$

In the context of neural networks, the resulting negative log-likelihood is well known as the Euclidean loss function. A GLM with the identity function as the inverse of the link function is equivalent to a Perceptron using a linear output function in the last layer. Training a Perceptron with the Euclidean Loss while applying a linear output function results in a model that estimates the expected value of a random variable sampled from a normal distribution with unit variance. The example in this section demonstrates how the NLL results from the assumption about the distribution of the response variable Y. Similarly, the output and loss function applied to train a Perceptron model are closely related.

6.2.2 *Loss Function for Attribute Probabilities*

The insights gained from the previous section are used to determine a suitable GLM that meets the requirements of the PRM from Section 6.1. The probabilistic retrieval model assumes that the binary attributes follow a Bernoulli distribution and require the estimates of attribute probabilities. The Bernoulli distribution has the beneficial property that the expected value corresponds to the probability of the random variable Y as follows

$$\mathbb{E}_{\mathcal{B}}[Y] = P(Y = 1) \cdot 1 + P(Y = 0) \cdot 0 = p \cdot 1 + q \cdot 0 = p \ . \tag{56}$$

As a consequence, estimating the expected value $\mathbb{E}_{\mathcal{B}}[Y]$ is equivalent to estimating the probability p. This section aims to derive a GLM that estimates the expected value of a Bernoulli-distributed random variable. Considering the assumption of the PRM that PHOC attributes are likewise Bernoulli-distributed random variables, the GLM obtained should be capable of directly estimating the attribute probabilities of the PHOC representations. Therefore, the PMF of the Bernoulli distribution is rearranged into the standard exponential form (cf. Equation 43)

$$\begin{aligned} f_{\mathcal{B}}\left(Y = y \mid p \right) &= p^y \cdot (1-p)^{(1-y)} \\ &= \exp\left(\log\left(p^y \, (1-p)^{(1-y)} \right) \right) \\ &= \exp\left(y \cdot \log p \; + \; (1-y) \cdot \log(1-p) \right) \\ &= \exp\left(y \cdot \log p \; - y \cdot \log(1-p) \; + \; \log(1-p) \right) \\ &= \exp\left(\log\left(\frac{p}{1-p} \right) \cdot y \; + \; \log(1-p) \right) . \end{aligned} \tag{57}$$

The canonical link function is the logarithm of the odds describing the ratio between the probability of event occurrence to its non-occurrence and is commonly known as the *logit*[1] [Agr15] function:

$$g_{\mathcal{B}}(p) \; = \; \log\left(\frac{p}{1-p} \right) . \tag{58}$$

Considering that the probability p is equal to the expected value $\mathbb{E}_{\mathcal{B}}[Y]$ of the random variable Y, the relationship from Equation 45 can be applied. Now, the inverse of the canonical link function can be obtained as follows

$$\begin{aligned} \eta(\mathbf{x}) &= g_{\mathcal{B}}(p) \\ \eta(\mathbf{x}) &= \log\left(\frac{p}{1-p} \right) \\ \exp(\eta(\mathbf{x})) &= \frac{p}{1-p} \\ \frac{\exp(\eta(\mathbf{x}))}{1+\exp(\eta(\mathbf{x}))} &= p \, . \end{aligned} \tag{59}$$

In the context of neural networks, the resulting inverse is known as the *logistic* function and is mostly referred to as sigmoid function. Considering the relationship $\eta(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, the logistic function can be further rearranged to obtain a form that is commonly used to express the sigmoid activation function.

$$\begin{aligned} \operatorname{sigm}(\eta(\mathbf{x})) &= g_{\mathcal{B}}^{-1}(\mathbf{w}^T\mathbf{x}) \\ &= \frac{\exp\left(\mathbf{w}^T\mathbf{x}\right)}{1+\exp\left(\mathbf{w}^T\mathbf{x}\right)} \cdot \frac{\exp\left(-\mathbf{w}^T\mathbf{x}\right)}{\exp\left(-\mathbf{w}^T\mathbf{x}\right)} \\ &= \frac{1}{1+\exp\left(-\mathbf{w}^T\mathbf{x}\right)} \, . \end{aligned} \tag{60}$$

1 logit is the abbreviation for **log**istic un**it**

This function defines the relationship between the output of the linear predictor $\eta(\mathbf{x})$ and the estimated expected value $\hat{\mathbb{E}}_{\mathcal{B}}[Y \mid \mathbf{X} = \mathbf{x}]$ by projecting the output value to the range $(0, 1)$. Moreover, the relationship between the input variable $\mathbf{x}$ and the response variable Y can be expressed as

$$\hat{\mathbb{E}}_{\mathcal{B}}[Y \mid \mathbf{X} = \mathbf{x}] = \text{sigm}(\mathbf{w}^T\mathbf{x}).$$

The reason is that the sigmoid function is equal to the inverse of the canonical link function (Equation 60) and the expected value $\hat{\mathbb{E}}_{\mathcal{B}}$ of the Bernoulli-distributed random variable Y corresponds to the probability p of this distribution (Equation 56). If the inverse function estimates the expected value, the sigmoid function estimates the Bernoulli probability equivalently.

Having derived the sigmoid function as the inverse of the link function and thereof the relation between $\mathbf{x}$ and Y, the next step is to obtain optimal model parameters. Similar to the example in Section 6.2.1, finding the optimal parameter set can be achieved by maximizing the likelihood or equivalently minimizing the negative log-likelihood of the PMF of the Bernoulli distribution. Concerning the PRM, the goal is to estimate attribute probabilities defined by the PHOC representation. Therefore, two modifications are required with respect to the response variable Y and the model parameters $\mathbf{w}$. Until now, it was assumed that Y is a scalar, but the PRM models the PHOC representation as a random variable $\boldsymbol{A}$ which is a vector of individual attribute probabilities, each following a Bernoulli distribution. Hence, the first modification replaces the scalar random variable $Y \in \{0, 1\}$ with the vector random variable $\boldsymbol{A} \in \{0, 1\}^M$, where M is the number of attributes (i.e., PHOC dimensionality). The second modification extends the model parameters from a vector $\mathbf{w} \in \mathbb{R}^N$, where N denotes the dimensionality of the input variable $\mathbf{x}$, to a matrix of parameters $\mathbf{W} = [\, \mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M \,]$. The expected value for the i-th attribute can now be estimated using the following expression

$$\mathbb{E}\,[\, A_i \mid \mathbf{X} = \mathbf{x} \,] = \text{sigm}\left(\mathbf{w}_i^T \mathbf{x} \right) \ ,$$

where A_i is the scalar random variable for the i-th attribute and $\mathbf{w}_i^T$ denotes the corresponding row vector of model parameters. The relation between the expected values of the attributes $\mathbf{A}$ and the input variable $\mathbf{x}$ can now be rewritten as a function of attribute estimates $\hat{\mathbf{a}}$ of input $\mathbf{x}$ given the model parameters $\mathbf{W}$ using the matrix notation

$$\mathbb{E}\,[\, \mathbf{A} \mid \mathbf{X} = \mathbf{x} \,] \ = \ \text{sigm}\left(\mathbf{W}^T \mathbf{x} \right) = \ \hat{\mathbf{a}}\,(\, \mathbf{x} \,|\, \mathbf{W}\,) \ . \tag{61}$$

In order to find the optimal parameters through maximum likelihood estimation, a set of annotated training samples is required. Therefore, the dataset $\mathbf{S}_{\mathcal{B}}$ is defined as

$$\mathbf{S}_{\mathcal{B}} = \left\{ \left(\mathbf{x}^{(i)}, \mathbf{a}^{(i)} \right) \right\}_{i=1}^{\mathsf{N_S}} \ , \tag{62}$$

where $\mathbf{x}^{(i)}$ is the i-th vector from a set of $\mathsf{N_S}$ training samples and $\mathbf{a}^{(i)} \in \{0, 1\}^M$ the corresponding PHOC vector consisting of M binary attribute values, each following a Bernoulli

distribution. The final step is to choose the optimal model parameters $\hat{\mathbf{W}}$ that maximize the likelihood $\mathrm{L}_{\mathcal{B}}\left(\mathbf{W} \mid \mathbf{S}_{\mathcal{B}}\right)$. In the context of neural networks, it is more appropriate to pursue the equivalent to maximum likelihood estimation by minimizing the negative log-likelihood, as this can be performed by applying stochastic gradient descent. Therefore, the following definition applies:

$$\hat{\mathbf{W}} = \underset{\mathbf{W}}{\operatorname{argmax}}\ \mathrm{L}_{\mathcal{B}}\left(\mathbf{W} \mid \mathbf{S}_{\mathcal{B}}\right) = \underset{\mathbf{W}}{\operatorname{argmin}}\ -\log\ \mathrm{L}_{\mathcal{B}}\left(\mathbf{W} \mid \mathbf{S}_{\mathcal{B}}\right)\ . \tag{63}$$

The negative log-likelihood for the PMF of the Bernoulli distribution is obtained as follows:

$$\begin{aligned}
-\log\ \mathrm{L}_{\mathcal{B}}\left(\mathbf{W} \mid \mathbf{S}_{\mathcal{B}}\right) &= -\log \prod_{i=1}^{\mathsf{N_S}} \prod_{j=1}^{D} f_{\mathcal{B}}\left(y_j = a_j^{(i)} \,\middle|\, p_j = \hat{a}_j^{(i)}\right) \\
&= -\sum_{i=1}^{\mathsf{N_S}} \sum_{j=1}^{D} \log\ f_{\mathcal{B}}\left(a_j^{(i)} \,\middle|\, \hat{a}_j^{(i)}\right) \\
&= -\sum_{i=1}^{\mathsf{N_S}} \sum_{j=1}^{D} \log\left(\ \hat{a}_j^{(i)\,(a_j^{(i)})}\ \cdot\ (1-\hat{a}_j^{(i)})^{(1-a_j^{(i)})}\ \right) \\
&= -\sum_{i=1}^{\mathsf{N_S}} \sum_{j=1}^{D} a_j^{(i)}\ \log\left(\hat{a}_j^{(i)}\right)\ +\ \left(1-a_j^{(i)}\right) \log\left(1-\hat{a}_j^{(i)}\right)\ .
\end{aligned} \tag{64}$$

Here, $\mathsf{N_S}$ denotes the number of training samples, and D is the dimension of a given attribute vector $\mathbf{a}^{(i)}$ from dataset $\mathbf{S}_{\mathcal{B}}$. The model's attribute prediction $\hat{a}_j^{(i)}\left(\mathbf{x} \mid \mathbf{w}_j\right)$ is abbreviated by $\hat{a}_j^{(i)}$, representing the estimate of the j-th attribute for the i-th sample given the corresponding parameter vector $\mathbf{w}_j$. In the PHOC representation, the j-th attribute is denoted by $a_j^{(i)}$, indicating whether a particular character is present. A significant benefit of the Bernoulli distribution is that the predicted expected value $\mathbb{E}\left[\, A \mid \mathbf{X} = \mathbf{x} \,\right]$ corresponds to the estimated attribute probability p. If the GLM is interpreted as a fully connected layer of a neural network with the sigmoid function at the output $f\left(\mathbf{x}\,,\, \hat{\mathbf{W}}\right) = \mathrm{sigm}\left(\hat{\mathbf{W}}^T\mathbf{x}\right)$, obtaining the optimal model parameters $\hat{\mathbf{W}}$ by minimizing the negative log-likelihood from Equation 64 results in a neural network model that estimates the expected values of the attribute representations $\mathbb{E}\left[\, A \mid \mathbf{X} = \mathbf{x} \,\right]$. This insight, in turn, means that the network directly predicts the individual attribute probabilities $p_j\left(a_j \mid \mathbf{x}\right)$, which are required by the probabilistic retrieval model.

7

CUNEIFORM SIGN SPOTTING

In addition to retrieval of handwritten words in document collections, spotting is adapted to cuneiform tablets. This chapter presents a novel approach for cuneiform sign spotting based on wedge expressions. While query-by-string is a well-established retrieval scenario in word spotting, no such query modality exists for retrieving cuneiform signs. The presented methodology enables the retrieval of cuneiform signs by encoding alphanumeric expressions that refer to the positions and constellations of particular wedge impressions. This retrieval scenario is referred to as *query-by-expression QbX* and was partially evaluated by the author of this thesis in [Rus+20; Rus+21]. Retrieving signs in the query-by-expression scenario is the equivalent of retrieving textual strings in the query-by-string scenario. The elaborated cuneiform signs encoding and its resulting attribute representations are the main contribution. These attributes refer to binary-valued semantics representing wedge impressions, their counts, and particular constellations. Additionally, wedge positions are encoded, applying a pyramidal splitting of cuneiform signs. This chapter presents the two approaches that yield attribute representations for cuneiform sign spotting in the QbX retrieval scenario. Therefore, the ensuing Section 7.1 introduces the cuneiform writing system. Subsequently, related cuneiform sign recognition and retrieval research are reviewed in Section 7.2. Finally, the proposed method for cuneiform sign spotting based on wedge expressions is presented in Section 7.3.

7.1 INTRODUCTION TO CUNEIFORM

The cuneiform script is one of the earliest attested writing systems in history, consisting of a continuously growing estimated minimum of 550 000 inscribed objects and tablet fragments [BRF21]. It was developed in Mesopotamia at the end of the 4th millennium BC and was used in the course of more than three millennia until the 1st century CE in large parts of the ancient Near East. The writing system was consequently adapted to represent a variety of local languages [Kra81]. Most contemporary writing systems are written with ink on paper, excluding digital texts, which can be seen as a 2 dimensional text representation. In contrast, cuneiform is a truly 3 dimensional script. It was primarily written by pressing a stylus into moist clay tablets to create signs composed of wedge-shaped impressions, henceforth called wedges [Kra81]. A typical tablet reassembled from joining fragments is visualized on the left side of Figure 22. The fragments were partially damaged through the millennia, showing severe attrition in some places. The right side of Figure 22 shows a section from the respective tablet image on the left side. This cutout section shows various cuneiform signs consisting of triangular-like wedge impressions. A cuneiform sign is composed of one or several wedge impressions, and the reader may identify a sign according

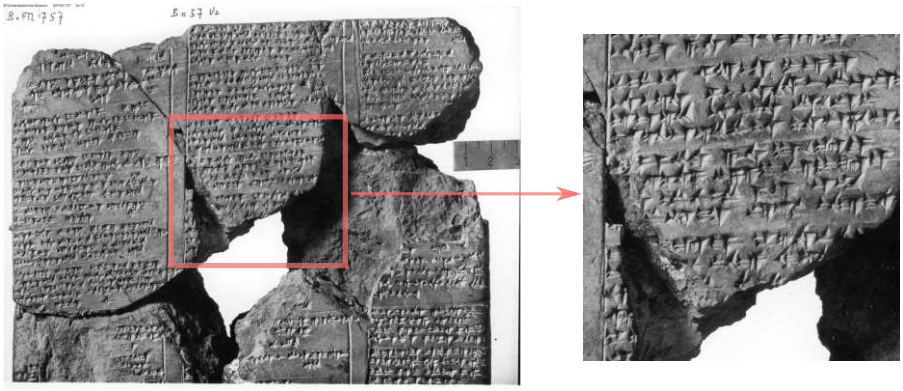


Figure 22: The left side shows a whole tablet reassembled from several fragments. The right side shows an enlarged section from the tablet. Image is provided by the *Hethitologie Portal Mainz* [Müloo].

to the direction, number, and positioning of these wedges, similar to brush strokes in a Chinese character [MO16]. As cuneiform signs may have syllabic or logographic values, the writing system contains significantly more signs than an alphabet, with an approximate maximum of 900. The use of this writing system in a broad temporal and geographic range, its adaptation to at least 7 languages, as well as the personal preferences of the scribe, both the inventories of regularly used signs and the form of the signs used, henceforth to be referred as sign variants, are highly heterogeneous [MO16]. Figure 23 visualize examples of six different cuneiform signs. Each row shows the name, a hand draw prototype and multiple visual examples for each sign class. The visual variability and quality regarding the wedge impression can be tremendous. Some signs have a well-preserved surface, and the wedge impressions are sharply visible. However, other signs have a damaged surface and exhibit significant aging artifacts.

7.2 RELATED RESEARCH

Although the cuneiform writing system is less known in the document analysis community, the first approaches have been proposed for cuneiform sign recognition and retrieval. Due to the 3D nature of cuneiform scripts, it is evident to model tablets and fragments as 3D representations. Therefore, it is worth noting that the first methods for computer-aided (visual) analysis of cuneiform scripts were proposed in the field of 3D graphics. Reviewing publications from this field is far beyond the scope of this thesis. Hence, this chapter focuses on the methods concerning recognizing and retrieving cuneiform signs. For further studies of computer-aided cuneiform analysis in the 3D domain, the works from Mara [Mar12] and Fisseler [Fis19] are referred.

This chapter categorizes the reviewed approaches according to the visual domains and the necessity of annotations and segmentation. Unfortunately, established datasets for cuneiform scripts do not exist. Consequently, all the methods reviewed in this chapter use individual cuneiform data and evaluation protocols, making comparing the approaches

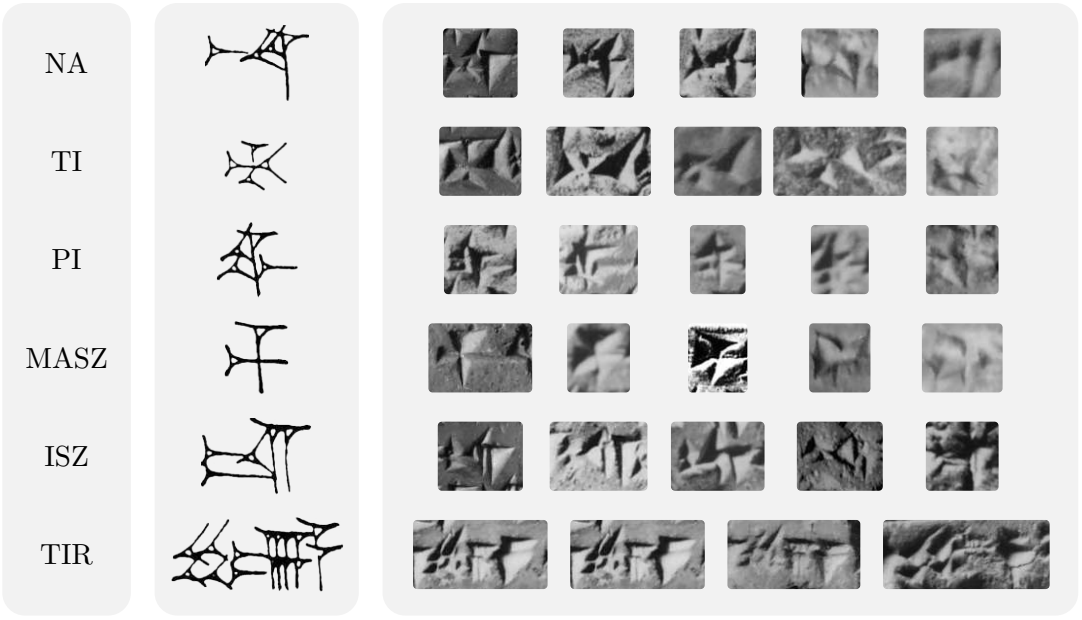


Figure 23: Visual examples of six cuneiform signs. Each row shows the sign classes *NA*, *TI*, *PI*, *MASZ*, *ISZ* and *TIR*, respectively. Column (a) shows the sign names, (b) visualize hand drawn prototypes and (c) depicts corresponding visual examples.

impossible. Nevertheless, this section provides an overview of the efforts that have been made in the field of cuneiform sign recognition and retrieval.

The first work for cuneiform sign spotting was presented by Rothacker *et al.* [Rot+15] and is based on the Bag-of-Features Hidden Markov Models (BoF-HMMs) approach [RWF21]. The sequence models have successfully been applied on different scripts like Roman [RMF13] or Bangla [Rot+13]. The proposed approach performs segmentation-free sign spotting in a QbE retrieval scenario without the necessity of annotations. Therefore, rasterized 2D images are obtained from 3D scans applying a differential operator [Mey+03] on triangle meshes and subsequently projecting the output on a 2D representation. Local image features are obtained by extracting dense SIFT descriptors followed by a quantization step with a vocabulary of 4096 visual words. Finally, retrieval is performed in two steps. The first step estimates the query HMMs using the Baum-Welch algorithm (cf. [PF09]). The second step computes the similarities between a given query and a grid of image patches through probabilistic inference with the Viterbi algorithm (cf. [PF09]).

A graph similarity approach for annotation-free retrieval of segmented cuneiform signs is provided by Bogacz *et al.* [BGM15b]. The signs are represented as strokes via the SVG format in order to obtain a graph representation. These strokes are hand-drawn tracings of wedge impressions provided by human experts (e. g. , Assyriologists) using a vector graphic editor. Once the graph representation has been obtained, cuneiform sign similarity is evaluated by applying three different graph matching approaches, namely the Weisfeiler-Lehmann graph kernel [She+11], spectral decomposition [LCH03] and the random walk graph kernel [Vis+10]. Additionally, the authors extend all three matching algorithms by

a Delaunay triangulation. The results show that the Delaunay triangulation increases the performance of spectral decomposition. While the same triangulation applied to Weisfeiler-Lehmann or random walk kernels significantly decreases retrieval performance. The benefit of this approach is that no annotations are required. However, obtaining the wedge strokes depends on hand-crafted tracings by human experts, which is a tremendous drawback of this method. In their work [BGM15a], the authors enhance the graph representation derived from wedge strokes and conduct segmentation-based sign retrieval using two different wedge assignment strategies. The assignment costs are used to measure the similarity between a given query and the candidates. The assignment strategies involve a simple wedge assignment along the x direction, or a minimization problem solved using the Hungarian algorithm [Mun57]. Moreover, the authors provide a comparison between the proposed approach and three word-spotting methods based on DTW [RM07], word warping [KBS11] and HMMs. Therefore, the wedge strokes are rasterized from the spline obtained from the vector graphics. Furthermore, the authors provide a baseline based on the ICP algorithm [BM92; Zha94]. This algorithm requires point clouds obtained from sampling along the splines. The results reported show consistently higher performance for the graph-based approach followed by the ICP algorithm. In contrast, all word-spotting methods perform inferior. The authors demonstrate that a graph-based representation can achieve higher retrieval performance than representations based on image features. However, wedge tracings must be provided by human experts, as techniques for direct wedge spline extraction are missing. A first approach for extracting SVGs from hand-drawn autographies is proposed by Massa *et al.* [Mas+16]. The method extracts skeletonized wedge strokes based on the potrace[1] algorithm and Voronoi diagrams [Kir79], followed by a heuristic extraction and pruning of possible wedge candidates. The authors demonstrate promising results for the extraction of graph-based wedge representations. However, the method is based on several heuristic assumptions which only fit some autographies. Moreover, the application of this method is limited to hand-drawn autographies rather than 2D photographs. In [BHM16], Bogacz *et al.* propose a method for segmentation-free cuneiform sign spotting. Therefore, the authors adapt the method of part-structured inkball models from Howe [How13]. The wedge strokes are extracted from spline keypoints representing a wedge's head, tail, and arms. After the wedge representations have been obtained, cuneiform signs are modeled as tree structures of connected wedges using the part-structured approach. The results compare to the BoF-HMMs approach from Rothacker *et al.* [Rot+15], concluding that the part-structured keypoints approach performs slightly better than BoF-HMMs. The authors in [BM18] present a hybrid method for segmentation-free sign spotting in the 3D domain. Although retrieval is performed in the 2D domain, the proposed approach demonstrates a possible bridge between the 3D and 2D domains. Therefore, local surface features are computed using MSII [MK13]. Then, radial geodesic patches are extracted from each vertex and embedded into a polar coordinate system. Subsequently, a rectangular sampling scheme converts the patches into rasterized images. Finally, spotting is performed using

1 The potrace algorithm was developed by Peter Selinger and is used for vectorizing bitmapped images: http://potrace.sourceforge.net/

HOG and ORB features, which are extracted from the image patches. As the data used is not annotated, only qualitative results are provided. The authors conclude that the proposed method does not achieve satisfying results regarding spotting but rather represents a text detector.

All previously mentioned works fall into the category of annotation-free methods. As mentioned, acquiring annotations for cuneiform scripts requires expert knowledge. Consequently, the annotation process requires human expert knowledge. In [BKM17], Bogacz *et al.* provide a method for automated transliteration using weakly-annotated data. Therefore, the authors exploit web-based[2] cuneiform data of autographies and the corresponding transliterations. The training process is carried out as an alignment problem to learn the correspondence between image descriptors and their corresponding tokens. First, the text lines are segmented using projection profiles [San+09]. Subsequently, dense HOG descriptors are extracted from the text lines. After the lines have been extracted, annotated training samples consisting of descriptors and tokens are generated. Following the assumption that the number of tokens equals the number of signs in a text line, tokens are assigned to image descriptors using the nearest neighbor upsampling. Finally, transliteration is performed by classifying the sequences of descriptors using HMMs. A very similar problem is approached by Dencker *et al.* [Den+20]. The authors propose a segmentation-free cuneiform sign recognition method based on neural networks. The presented method learns to localize and recognize cuneiform signs by aligning text lines and their corresponding transliterations using an iterative learning approach under weak supervision. A training iteration is applied in three steps, starting with the localization of transliterated signs in the tablet using detected lines and aligned detections as reference points (*sign placement*). In the second step, the sign detector is trained using the generated aligned and placed detections as sign annotations in order to obtain raw detections (*sign detector training*). Finally, these raw detections are refined in the last step by optimizing the alignment between the transliteration and tablet image (*image-transliteration alignment*). The lines are segmented using a pre-trained RPN, followed by a post-processing step applying the Hough transformation. The neural network is initially pre-trained on 410 text lines from 38 annotated tablet images. Afterward, further text lines are segmented in the remaining tablet images using the resulting network model. These automatically segmented text lines are aligned to the transliteration following a naive approach, where the first segmented text line is assigned to the first line from the transliteration. Finally, the center position of each sign is determined along the segmented text lines by computing the bounding box sizes using statistics obtained from Unicode fonts [Van18]. Although the evaluation shows compelling results for weakly supervised cuneiform sign detection, the authors state that the approach struggles with classifying unfrequent signs, and, thus, additional annotations of rare cuneiform signs are still required.

2 The following databases are used
Cuneiform Digital Library Initiative (CDLI): `http://cdli.ucla.edu/`
Open Richly Annotated Cuneiform Corpus (ORACC): `http://oracc.museum.upenn.edu/`
Cuneiform Commentaries Project (CPP): `http://ccp.yale.edu/`

In [Kri+18], the authors approach the problem of directly classifying cuneiform signs in the 3D domain using GNNs. The wedges are represented as graphs and extracted from 3D scans obtained from the approach in [Fis+13]. The resulting wedge model consists of four vertices (i.e., depth, tail, right, and left), one of the three wedge types (i.e., vertical, horizontal, or Winkelhaken), and the spatial location in the two-dimensional Euclidean space. The authors apply a SplineCNN [Fey+18] that directly operates on the graph-based inputs. The network is trained end-to-end and transforms a graph representation into an arbitrary feature vector. Subsequently, the feature vector is mapped to a one-hot encoded vector using a fully connected layer. The results compare the GNN and a heuristic approach based on a graph edit distance carried out as the nearest neighbor classification. As both approaches achieve nearly equal performance, the authors conclude that more annotated data is required as the networks remain "data hungry". Another work using GNNs in the 3D domain is presented by Bogacz *et al.* [BM20]. The authors propose a method for time period classification of cuneiform tablets. 3D cuneiform tablets are provided by the Heidelberg Cuneiform Benchmark Dataset (HeiCuBeDa) [MB19]. In contrast to Kriege *et al.* [Kri+18], graph-based wedge representations are obtained using MSII [MK13]. The proposed network architecture is based on the SplineCNN [Fey+18], followed by a pooling strategy from the PointNet++ [Qi+17]. Similar to Kriege *et al.* [Kri+18], classification is carried out as a one-hot encoding, and the network is trained in an end-to-end fashion using the categorical cross-entropy loss. Most recently, Rest *et al.* [Res+21] presented a method for illumination-based data augmentation to improve the classification of cuneiform signs on 2D photographs. Usually, experts use only a single illumination source during the digitization process of cuneiform tablets. As a result, photographs only contain a single illumination source. This technique improves the visualization of wedge impressions in photographs. An example of illumination-induced shadow spots is shown in Figure 22 from Section 7.1. The authors use a ResNext [Xie+17] architecture for sign classification in a one-hot encoded fashion. The provided results indicate that augmenting the data based on illumination improves the performance. However, the authors note that noteworthy improvements are achieved if the synthetic illumination fits the illumination angle of the target data. Furthermore, annotated data from 3D cuneiform tablets is required before the illumination-based augmentation can be applied.

7.3 WEDGE-BASED SIGN SPOTTING

Several works [Rot+15; BGM15b; BGM15a] reviewed in the previous section provide methodologies for cuneiform sign spotting. However, the query modality is limited to a QbE retrieval scenario. This section presents an alphanumeric encoding similar to chemical expressions. This encoding refers to predefined wedge impressions, their counts, and constellations. Representing cuneiform signs through their wedges leads to attribute-based representations and enables the QbX retrieval scenario. Cuneiform signs are made by pressing a stylus into wet clay tablets, which leaves a wedge-shaped imprint. The types, counts, and patterns of wedge impressions define each cuneiform sign's meaning. These impressions

serve as the basic building blocks of the cuneiform writing system, much like the characters of the Latin alphabet. In the context of word spotting, these characters represent an extensive vocabulary through binary-valued attribute vectors. These attribute-based representations enable the QbS retrieval [Alm+14b]. The approach presented in this section adapts the principal for QbS retrieval in order to enable the QbX cuneiform sign spotting. Therefore, the alphanumeric encoding is explained in the following section (Section 7.3.1). The attribute vector obtained from this encoding holistically represents wedge impressions of cuneiform signs. Two approaches for the pyramidal division of cuneiform signs are shown to enrich this representation with spatial information and encode the positions of particular wedges. The first approach, discussed in Section 7.3.2, is inspired by the *spatial pyramid* [LSP06] and splits the signs according to a *grid-based* pyramidal division. For the second approach, wedge impressions are annotated according to their sequential order, inspired by the order of characters forming a word in Latin scripts. A primary advantage of *sequence-based* encoding is that the PHOC algorithm can directly be applied to obtain the attribute representation as shown in Section 7.3.3.

7.3.1 *Gottstein System*

In [Got13], Gottstein presented the idea of a unified cuneiform sign representation based on commonly occurring wedge types, which is referred to as the *Gottstein-System*. The author argues that sign lists and lexica used by researchers are less suitable for representing signs in databases, especially for retrieval tasks. This representation should be applicable to anyone without expert knowledge of the sign lexicon and the specific language used in databases. In order to simplify the retrieval process, Gottstein proposed an alphanumeric encoding of cuneiform signs based on 4 commonly occurring wedge types. The cuneiform signs are decomposed into wedge types denoted by a, b, c, and d. Figure 24 shows hand-drawn examples of the four wedges types from the Gottstein-System. A wedge type is defined according to the pointing direction of its wedge tail. Type a denotes a *vertical* wedge tending from top to bottom and vice-versa. b represents a *horizontal* wedge type, tending from *left-to-right*. Here, the variant *right-to-left* does not exist [Kra81]. The type c represents three different variants of wedges, namely the "Winkelhaken" and an oblique wedge tending diagonally from the upper left to the lower right as well as from the lower right to the upper left direction. Finally, d represents a perpendicular wedge type to c, tending from the lower left to the upper right direction and vice-versa. Furthermore, the Gottstein-System indicates each wedge type's count for a cuneiform sign. An example of

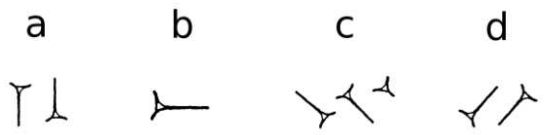


Figure 24: Four hand drawn examples of wedge types from the Gottstein-System and their corresponding alphanumeric encodings.

three hand-drawn cuneiform signs is depicted in Figure 25, showing an increasing complexity of signs from left to right. A simple sign of two a and one b wedge types is shown on the far left. The sign in the middle shows three different wedge types (a, b and c) and a wedge count of $2+2+1=5$ in total. Finally, the sign on the far right side shows a more complex wedge constellation consisting of three wedge types and a count of $10+2+4=16$ wedges.

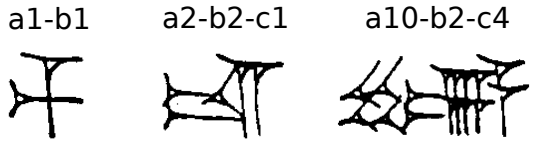


Figure 25: Visualization of three hand drawn cuneiform signs and their respective Gottstein encodings, showing a simple sign on the left side and a more complex sign on the right side [Rus+21].

7.3.2 *Spatial Pyramid Gottstein Representation*

Although the Gottstein System provides a holistic representation of a particular cuneiform sign, this simplified description exhibits a shortcoming. As some sign classes consist of the same wedge types and counts, it is impossible to distinguish between these signs using the Gottstein-System. Figure 26 shows the drawback of the Gottstein representation that results in ambiguous mappings between sign classes and their alphanumeric encodings. Here, the three different cuneiform signs *NA*, *TI*, and *PI* are described by the identical Gottstein encoding. Although all three signs have different visual appearances, they are mapped to the same alphanumeric encoding ($a1$ $b1$ $c2$). The Gottstein system cannot cover the whole set of cuneiform signs regarding a one-to-one mapping between classes and attribute vectors. This ambiguous many-to-one assignment is resolved by decomposing the signs using pyramidal partitioning. In Latin scripts, writings are intuitively partitioned according to their reading direction from left to right, while single characters are represented as binary attributes. Although the common reading direction for cuneiform texts tends from left to right, human experts perceive the signs holistically. Hence, an apparent wedge arrangement in the reading direction is missing. Despite the lack of this apparent arrangement of wedges, cuneiform signs are represented through a pyramidal scheme by applying a *spatial pyramid* pattern, much like the spatial pyramid approach presented in [LSP06]. Figure 27 shows an example of the proposed partitioning of the sign *NA*. This sign is split pyramidal using the levels 2 and 3 in the horizontal and vertical directions, respectively. The

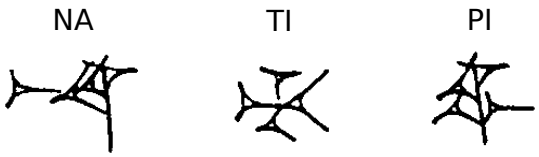


Figure 26: Visualization of three different cuneiform signs consist of the same wedge types in different constellations, represented by the same Gottstein encoding $a1$ $b1$ $c2$ $d0$ [Rus+21].

partition effectively results in left-right and top-bottom splits on the second level, while on the third level, the sign is split into left-middle-right and top-middle-bottom parts. Assigning annotations to splits based on the visual appearance entails an issue. In contrast to the PHOC representation, where word strings are assigned to splits according to their characters, cuneiform signs do not exhibit this "natural" partitioning. Hence, the annotation process remains fully manual. While some wedge constellations can be distinctively assigned to a certain split, many wedges are located between two splits, and a clear affiliation to a certain split is missing. Therefore, two assignment rules are included to resolve this ambiguity. First, wedges are assigned to splits according to the location of their head. Second, wedge heads intersecting a split boundary are assigned to both respective splits.

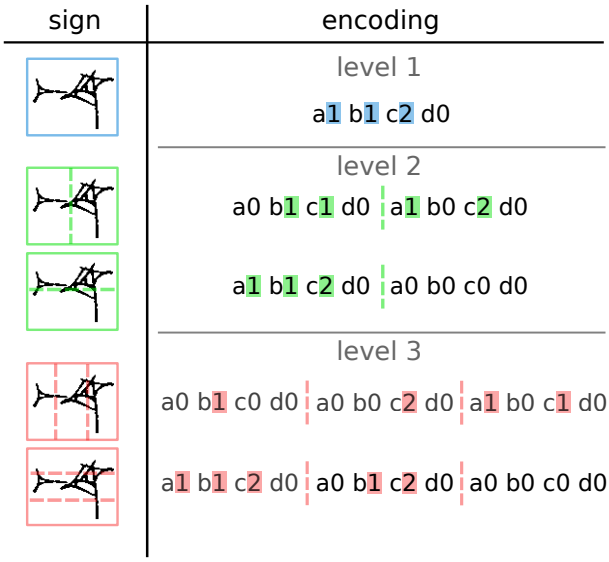


Figure 27: Visualization of a pyramidal splitting of the sign *NA* in the horizontal and the vertical direction for the levels 1, 2 and 3, respectively.

7.3.3 *Temporal Pyramid Gottstein Representation*

The grid-like annotation approach supplements the Gottstein system with spatial information regarding the wedge positions and their respective counts. However, assigning the alphanumeric wedge encoding to a particular split within the predefined grid-like pattern leads to ambiguous annotations. This approach forces the annotator to assign the wedge encoding to predefined splits. Moreover, it is possible to assign a visual wedge constellation to two separate attribute semantics as shown in Figure 27. For the horizontal splits on level 2, the encoding $c2$ is divided into $c1$ on the left side and $c2$ on the right side. The two Winkelhaken wedges are now referred to by two different semantics, one indicating two c-type wedges ($c2$) and the other indicating only one c-type wedge ($c1$). However, the $c1$ wedge can have a considerably different visual appearance from other cuneiform signs.

An additional annotation approach is introduced on these grounds, where cuneiform signs are annotated following a *sequence-like* pattern. In contrast to the grid-based approach, vertical information is dropped, and instead, cuneiform signs are annotated ac-

cording to their wedge positions in a horizontal direction from left to right. This approach is inspired by the pyramidal concept of the PHOC representation, where textual strings are decomposed according to their sequential position. Figure 28 shows six examples of the cuneiform signs previously displayed in Figure 26 and 25. Each sign contains color boxes that emphasize the relative position of wedges and correspond to the color boxes shown in the respective alphanumerical encoding. The dash symbol separates the sequential position on an annotated encoding. For example, in the upper left corner of Figure 28, the encoding $b1\,c2\,a1$ represents a sequence of three wedge encodings as they visually appear in the cuneiform sign. Compared to the grid-based annotations, the sequence-based approach provides more flexibility. For example, if a cuneiform sign does not show a clearly sequential arrangement of wedges, such as in the upper right example of Figure 28, the annotation is not forced to indicate a sequence. Instead, the wedge semantics $a1b1$ can be annotated. Some cuneiform signs show vertical wedge constellations, like the sign in the middle left of Figure 28. In this case, vertical sign positions are not explicitly encoded, and instead, the encoding $b1{-}a1c1{-}c1$ is annotated, indicating $b1$ on the left, $a1c1$ in the middle, and $c1$ on the right side.

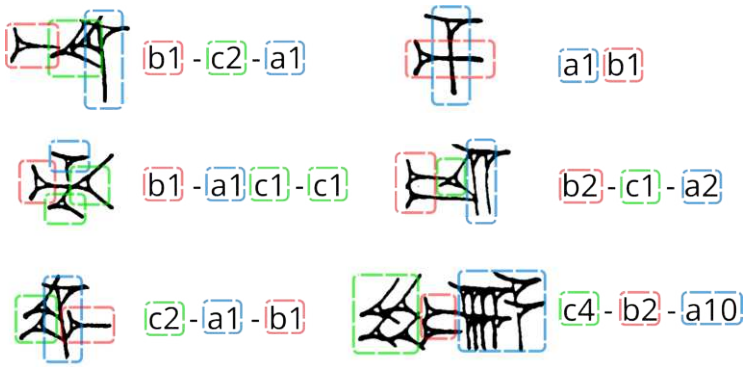


Figure 28: Visualization of six visual samples and their corresponding annotations. The color boxes emphasis the visual area represented by the corresponding wedge encoding.

Besides the sequential arrangement, visual ambiguity exists for equal wedge counts. Both signs in the upper and lower left corners of Figure 28 contain such ambiguity for the wedge encoding $c2$. These wedges are arranged horizontally in the upper and vertically in the lower sign (green boxes). Four additional wedge semantics are introduced to resolve this ambiguity, as shown in Figure 29. These semantics indicate a crossing (x) between two wedges, a vertical (v) or horizontal (h) wedge constellation and wedges arranged as a grid (g). Similar to Figure 28, the color boxes show the visual position and the corresponding information type. The two cuneiform signs mentioned above are augmented by the information indicating horizontal and vertical constellations such that their encodings result in $b1{-}c2\boldsymbol{h}{-}a1$ and $c2\boldsymbol{v}{-}a1{-}b1$.

The TPG representation is obtained by first constructing the sequence of attribute vectors. For example, the encoding $b1{-}c2h{-}a1$ is converted into a sequence of three vectors, each representing the respective encoding b1, c2h, and a1. The target encoding

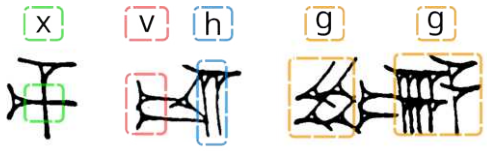


Figure 29: Visual examples of the additional wedge annotations indicating a crossing (x), a vertical (v) or horizontal (h) positions, and grid (g) constellations

is indicated by setting the corresponding vector element to 1 and all other elements to 0. This process is repeated for the remaining vectors until the desired sequence of vectors is obtained. The sum of maximum wedge counts determines the dimensionality of the vectors. For example, if the wedge types a, b, c, and d have their respecitive maximum counts of 10, 10, 9, and 3, the resulting dimensionality is 32. Finally, the PHOC algorithm assigns the sequence of vectors to predefined pyramid levels. Figure 30 gives three examples of how encodings are assigned to the pyramid levels 1, 2, and 3. The third example (from left) shows the resulting encodings for each split on the levels $1, 2$ and 3 for the annotation $b1{-}c2h{-}a1$ mentioned above. All wedge annotations are assigned analog to the grid-based approach on the first level. However, the significant difference is that on higher levels, the wedge counts remain unchanged in contrast to the SPG representation. Although the second pyramid level visually intersects the encoding $c2h$, the count remains unchanged and is assigned to both splits in this level. The main argument for this approach is that, in all splits, the count 2 for wedge type c is represented by the same attribute. This assignment strategy supports the consistency between the visual appearance of wedges and the attributes indicating their presence.

annotation	a1b1x	c2v-a1-b1	b1-c2h-a1	c4g-b2v-a10g
level 1	a1b1x	a1 b1 c2v	a1 b1 c2h	a10g b2v c4g
level 2	a1b1x | a1b1x	a1 c2v | a1 b1	b1 c2h | a1 c2h	b2v c4g | a10g b2v
level 3		c2v | a1 | b1	b1 | c2h | a1	c4g | b2v | a10g

Figure 30: Four examples showing a pyramidal decomposition of the TPG encodings according to the PHOC algorithm on the levels 1, 2 and 3.

8

EVALUATION

The probabilistic retrieval model presented in Chap. 6 is a novel measure assessing the similarity through attribute probabilities between a given query and the estimated representations. Chap. 7 introduces two wedge-based annotation approaches and the resulting binary-valued attribute representations for cuneiform sign spotting. This chapter provides the empirical evaluation of the PRM similarity measure and the two sign representations SPG and TPG. The experiments that are conducted give answers to the following questions derived from the methodology in this thesis.

- **Is the probabilistic retrieval model beneficial for retrieval performance when using binary-valued attribute representations?** Assessing the similarity between a user defined query and all database candidates is the most crucial step in every information retrieval system. The experiments in this section evaluate four different similarity measurements in multiple retrieval scenarios. The main focus is on the comparison between the performance of the probabilistic retrieval model (cf. Section 6.1) and the Cosine similarity (cf. [SF18]). In addition, the Euclidean and Cityblock distances are evaluated and compared.
- **How do architectural design choices impact the retrieval performance?** The network architectures presented for state-of-the-art models [WB16; SF18; KJ19] are based on specific components such as the structure of convolutional layers, a pyramidal pooling of feature maps or the attribute estimates provided by the Multi-Layer Perceptron. This leads directly to the question of how strongly these components influence the performance of these models. The experiments in this chapter evaluate the change in performance if the CNN components are altered.
- **Are the SPG and TPG representations suitable for QbX retrieval? Moreover, is one representation superior to the other?** The novel retrieval scenario QbX can be performed by transforming a cuneiform sign into its numerical representation. Both, the SPG and TPG provide sign presentations based on semantic attributes. This chapter evaluates multiple experiments to quantify the performance of the attribute-based QbX retrieval. This evaluation gives insight into the performance of both grid- and sequence-based cuneiform sign representations on different pyramid levels. The obtained results are compared to answer whether one representation is superior to the other.

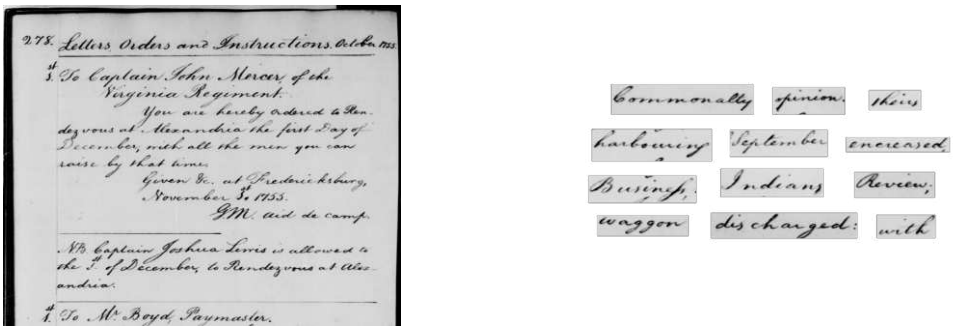
The experiments in this chapter are evaluated on word spotting benchmarks consisting of handwritten documents and photography collections showing cuneiform tablets introduced in Section 8.1. The mAP is utilized to quantify the retrieval performance (Section 8.2.1). A permutation test assesses statistical significance between two numerically close mean

average precision values (Section 8.2.2). Subsequently, Section 8.3 explains the experiment configurations and describes the evaluation protocol, network architectures, and the hyperparameters used to train the convolutional neural networks. Finally, the Section 8.4 and 8.5 present the evaluation results obtained for the word spotting and cuneiform sign spotting benchmarks.

8.1 BENCHMARK DATASETS

The methodology presented in Chap. 6 and 7 is evaluated on six benchmarks. This section gives a brief introduction to the databases used to carry out the experiments. Four different datasets, showing historical and modern handwriting, are used for the evaluation of word spotting. The cuneiform sign spotting experiments are performed on two benchmarks containing scripts from ancient periods.

8.1.1 *George Washington Letters*



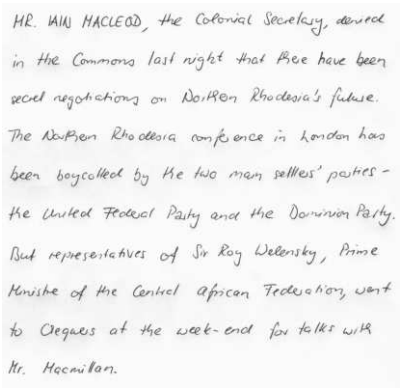
(a) Page sample (b) Word image samples

Figure 31: Visual page and word image examples from the George Washington database.

The GW database is one of the most widely used benchmarks for evaluating word spotting methods [Gio+17]. The letters originate from the George Washington Papers collection at the Library of Congress in the United States. The collection contains approximately 77 000 items, including correspondences, diaries, financial records, and more, divided into 9 series according to different topics [Was23][1]. The word spotting benchmark comes from *Series 2, Letterbooks,* 1754-1799 [Was54][2] and contains copies of mail correspondence between George Washington and his associates. The document pages are from the *Letterbook 1*, which is a later re-copy [SF18]. It is assumed that only a single writer has reproduced the letters. Thus, the writing style across the document pages is very homogeneous. In the context of word spotting, the dataset has been described for the first time in [KLP01]. Later, a version of the benchmark containing 20 document pages and 4 860 annotated word images have been proposed and is often referred to as *GW 20* [LRM04][3]. The 20 document pages consist of a lexicon covering 1 124 unique transcriptions. Figure 31a shows a cutout from a document page of the collection, and 31b depicts samples of individual word images.

1 https://www.loc.gov/collections/george-washington-papers/about-this-collection/
2 https://www.loc.gov/item/mgw2.001/
3 Available at the University of Massachusetts https://ciir.cs.umass.edu/downloads/old/data_sets.html

8.1.2 *IAM Handwriting Database*



(a) Page sample

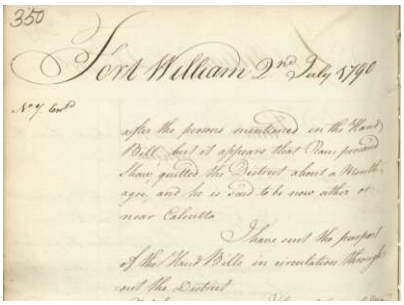
(b) Word image samples

Figure 32: Visual page and word image examples from the IAM handwriting database.

The IAM-DB was initially designed for handwriting recognition tasks [MB99]. This database consists of document pages showing the modern handwritten text of the *Lancaster-Oslo/Bergen* corpus [JLG78]. Several extensions have been proposed [MB02], mainly focusing on adding more handwritten document pages to the database. In the most recent version 3.0 available at the University of Bern[4], 657 different writers contribute to the creation of the IAM-DB with 1 539 handwritten document pages. The database consists of 5 685 isolated sentences, 13 353 labeled text lines and 115 320 word-level annotations. The word images have been segmented automatically and subsequently verified manually [ZB02]. Thus, some of the annotations are labeled as erroneous. All document pages are scanned with modern devices and are available as high-resolution images. Due to the modern equipment used to create this database, the handwritten texts have a high visual quality and aging artifacts such as faded ink or bleed-through are missing. However, the major challenge provided by the official partitioning is that a writer has contributed exclusively to the training set or the test set only. In the last decade, the word spotting community has adopted the official (writer-independent) partition [Fis+10; Fis+12; Alm+14b]. Especially Almazán *et al.* [Alm+14b] proposed a benchmark for segmentation-based word spotting, which is commonly used in recent works [WB16; SF18; KJ19]. Visual examples are depicted in Figure 32 showing a whole page on the left side (a) and word images from different pages on the right side (b). The word images, picked from multiple pages written by different writers, visualize the enormous diversity in handwriting styles.

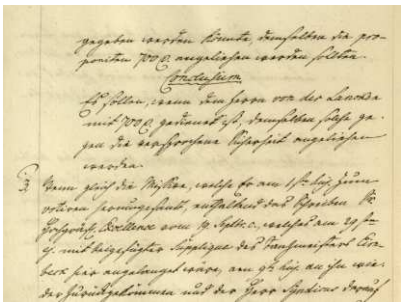
4 Research Group on Computer Vision and Artificial Intelligence INF, University of Bern: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database

8.1.3 *Botany and Konzilsprotokolle*



(a) Botany page sample

(b) Botany word image samples



(c) Konzilsprotokolle page sample

(d) Konzilsprotokolle word image samples

Figure 33: Visual page and word image examples from the Botany and Konzilsprotokolle databases.

The two benchmarks *Botany in British India* (Botany) and *Alvermann Konzilsprotokolle* (Konzilsprotokolle) have been used in the word spotting competition at the *ICFHR 2016* [Pra+16]. The Botany collection was written in the 19th century and contains manuscripts covering botanical topics. The original collection is archived at the *British Library*[5] in London, UK. Many document pages contain different writing styles and unique visual appearances. The script is written in English *Latin* and exhibits the typical aging artifacts for historical documents, such as faded ink and bleed-through. The Konzilsprotokolle collection dates back to the 18th century and shows protocol notes of formal meetings held by the central administration. The collection belongs to the University Archives and was digitized and provided by the *University Library Greifswald*[6], Germany. In contrast to Botany, the script is written in German *Kurrent* and contains around 18 000 documents. The writing style on the document pages is relatively homogeneous compared to Botany. However, the significant challenges are the characteristics of the Kurrent script, as the visual appearances of some characters are very similar. Analogous to Botany, documents from the 18th century are not immune to aging and show the same artifacts. Visual page and word image samples from both databases are displayed in Figure 33. The Botany dataset contains 25 432 word image samples annotated on 154 document pages. These word images represent 6 919 unique transcriptions. For Konzilsprotokolle, 21 009 word image samples have been annotated using a collection of 85 document pages which consist of 6 298 unique transcriptions.

5 https://www.bl.uk/collection-guides/botany-in-british-india
6 https://www.digitale-bibliothek-mv.de/viewer/index/

8.1.4 *Hittite and Old Assyrian Cuneiform Databases*

(a) Hittite tablet sample

(b) Hittite sign image samples

(c) Old Assyrian tablet sample

(d) Old Assyrian sign image samples

Figure 34: Visual examples of cuneiform tablets and signs from the Hittite and Old Assyrian databases.

The Cuneiform databases used in the evaluation of this chapter include annotated tablets of Hittite and Old Assyrian scripts. Both databases have been prepared in the context of the research project *Computer-unterstützte Keilschriftanalyse* (CuKa)[7]. Tablets containing Hittite cuneiform scripts are provided by the *Hethitologie Portal Mainz* (HPM)[8], and the Old Assyrian scripts are accessible on the webpage of the *Cuneiform Digital Library Initiative* (CDLI)[9]. The Hittite cuneiform script dates back to around 3 500 BC, while the Old Assyrian tablets were created around 1 950 to 1 850 BC. These clay tablets display various cuneiform text, including correspondence and bureaucratic documents related to trade and stocks. Figure 34 shows visual examples of both scripts. The Hittite database consists of 261 tablet images containing 316 unique cuneiform signs distributed across 100 370 annotated samples. In contrast, the collection of Old Assyrian tablets only contains 24 images consisting of 5 745 annotated samples representing 161 unique signs.

7 Akademie der Wissenschaften und der Literatur Mainz: https://www.adwmainz.de/projekte/computer-unterstuetzte-keilschriftanalyse-cuka/beschreibung.html
8 Hethitologie Portal Mainz (HPM): https://www.hethport.uni-wuerzburg.de/HPM/index.php
9 Cuneiform Digital Library Initiative (CDLI): https://cdli.mpiwg-berlin.mpg.de/

8.2 PERFORMANCE MEASURES

Assessing the performance of the methodology described in Chap. 6 and Chap. 7 involves both qualitative and quantitative analyses. Qualitative analysis is based on a user's impressions and considers only a small set of visual examples. On the other hand, quantitative analysis evaluates performance by considering a large number of examples. When comparing the performance of different approaches, it's preferable to use quantitative analysis as qualitative analysis relies heavily on the user's assessment and the selection of individual examples.

8.2.1 *Mean Average Precision*

Word spotting is an information retrieval problem [Gio+17; Alm+14b], and the same applies to cuneiform sign spotting [Rot+15]. Information retrieval deals with the problem of extracting relevant information from a database. The retrieval system issues a query to the database and returns a list of items, often called retrieval lists. This list contains candidates from the database that are either *relevant* or *irrelevant* concerning the query. A user is particularly interested in the relevant candidates from the retrieval list. However, in information retrieval, the relevance of candidates needs to be globally defined and strongly depends on the application domain. In word spotting, relevance is typically assigned based on the word classes (i.e., transcriptions) of the candidates. This approach of relevance assignment is widely used in the word spotting literature and can be considered as the de facto standard [Gio+17]. The performance of retrieval systems is evaluated using performance measures, which quantify the retrieval quality and allow for comparison between different systems. These measures assess three performance indicators for a retrieval list. The first two indicators evaluate the number of relevant items in the list and the proportion of relevant items retrieved out of the total number of relevant items. The third indicator assesses the ranking order of the list items. The first two indicators are quantified using popular metrics known as *precision* and *recall*, which are widely used in the fields of statistics and machine learning [BR11]. Precision represents the fraction of relevant items in the retrieval list out of all the items retrieved, and is defined as:

$$\text{precision} = \frac{\text{relevant items in list}}{\text{all items in list}}. \tag{65}$$

Conversely, recall computes the fraction between relevant items in the retrieval list and the total number of relevant items in the database. The recall is defined as:

$$\text{recall} = \frac{\text{relevant items in list}}{\text{all relevant items}}. \tag{66}$$

Usually, precision and recall are not discussed in isolation as both are not particularly useful metrics when used singularly, especially for information retrieval tasks. Both are closely related measures and capture different aspects of the retrieval list [BR11]. There

is an inverse relationship in which it is possible to increase precision in exchange for low recall values and vice versa. For instance, returning only a single candidate with the highest likelihood of being relevant will likely result in a precision of 100% but lead to a small recall value if more than one relevant candidate is in the database. Returning all candidates from the database, on the other hand, leads to a recall of 100% and a vast number of irrelevant candidates, decreasing the precision. For that reason, both are usually combined into a single-valued measure. Therefore, the first approach could be to use the well-known *accuracy* or even the more sophisticated *harmonic mean of precision and recall* better known as *F-measure* [MRS08]. However, a user demands the retrieval system to present all relevant candidates at the top of a list. Unfortunately, neither the accuracy nor the F-measure is sensitive to the ranking order of a retrieval list and is thus incapable of quantifying the third performance indicator.

In this regard, the most commonly used measure that takes the ranking order into account is the *AP* [BR11; MRS08]. This measure considers both precision and recall simultaneously by evaluating the *area under* the *precision-recall-curve* sometimes referred to as *AUPRC* [MRS08]. The precision-recall curve is obtained by evaluating the precision values pr at different recall levels r, which can be expressed as $pr(r)$. For classification-related tasks (for example, object detection [Zou+19]), recall levels are typically defined throughout 11 standard values, i.e., $k \in \{0.0, 0.1, 0.2 \dots 1.0\}$. In contrast, precision and recall values are assessed at each position along the retrieval list. The positions are defined according to the ranking order $k \in \{1, 2, 3, \dots K\}$, where K is the number of elements in the retrieval list. Computing the precision at every position k is known as *precision@k* [BR11, p.140]. As a result, precision pr and recall r at position k can be defined as:

$$pr(k) = \frac{\sum_{i=0}^{k-1} rel(i)}{k} \qquad r(k) = \frac{\sum_{i=0}^{k-1} rel(i)}{|\mathbf{R}|} \ , \tag{67}$$

where both $pr(k)$ and $r(k)$ yield respective values if only the top-k candidates of a retrieval list are considered. The recall is normalized by the cardinality of set $\mathbf{R}$, which represents the number of all relevant items in the database. The function $rel(i)$ indicates the relevance of a candidate at position i by evaluating 1 if relevant and 0 otherwise. The average precision is obtained by computing the area under the precision-recall curve as a finite sum:

$$AP = \sum_{k=1}^{K} p(k) \times \Delta r(k) \ . \tag{68}$$

Every precision value at position k is multiplied by the *change rate* of the recall $\Delta r(k)$, which quantifies the difference between two consecutive recall values defined as:

$$\Delta r(k) = \mid r(k-1) - r(k) \mid \ . \tag{69}$$

Since the indicator function $rel(\cdot)$ evaluates to only two values, either 1 or 0, the change rate $\Delta r(k)$ consequently takes only two values as well, namely 0 or $\frac{1}{|\mathbf{R}|}$. In Equation 68,

the precision values at different positions contribute to the average precision only if the recall value changes, i.e., $\Delta r(k) = \frac{1}{|\mathbf{R}|}$. This consideration leads to an equivalent expression of the average precision by replacing the recall change rate with the relevance indicator function and normalizing the sum by the number of relevant candidates in the database:

$$AP = \frac{\sum_{k=1}^{K} p(k) \times rel(k)}{|\mathbf{R}|} \ . \tag{70}$$

For the evaluation over a query set $\mathbf{Q} = \{1, 2, ...\mathsf{Q}\}$ the average precision values obtained for each query $q \in \mathbf{Q}$ are averaged in the following way:

$$\text{mAP} = \frac{1}{\mathsf{Q}} \sum_{q=1}^{\mathsf{Q}} AP(q) \ . \tag{71}$$

Averaging the $AP(q)$ values results in the performance measure known as mAP, which is widely used for evaluating retrieval systems [BR11, p.159]. As a result, mean average precision has become the standard measure for evaluating word spotting model performance [Gio+17]. This thesis assesses the methodology's performance using mean average precision and compares the results with those reported in the word-spotting literature.

8.2.2 *Significance Testing*

Performance differences between two retrieval systems are compared by evaluating the mean average precision values achieved by both systems using the same evaluation protocol (cf. Section 8.3.1). Using a single value measure to assess the superiority of two systems involves a certain degree of uncertainty, especially for small differences. The difference between two mAP values could stem from significant differences in just a few APs or variations across many AP values. In the first case, the difference would be caused by the selected query set, while in the latter case, it is more likely that the difference results from a methodological advantage. Specifically, training neural networks involves stochastic processes such as the initialization of network parameters, the distribution of sampled training samples (i.e., the progress of SGD), or regularization techniques like dropout (cf. Section 2.2.4). Therefore, it is important to have additional evidence to support a statement about performance differences.

In this case, the common practice is to apply a *statistical hypothesis test*. As the name suggests, these tests statistically assess whether a hypothesis made about two random variables is unlikely enough to be rejected. The hypothesis is referred to as a *null hypothesis* and typically assumes that the means of these variables are equal. Consequently, the opposite assumption is that both means are unequal, known as the *alternative hypothesis*. The null hypothesis is typically rejected if the probability of observing its assumption is below a certain threshold. This probability is expressed by the *p-value* and results from a *test statistic*, i.e., distribution of a particular measure, individually defined by the tests. The threshold is a meta-parameter known as the *significance level* and has to be defined by the user beforehand. If the null hypothesis is rejected, the result of a test is called *statistically*

significant. It is important to note that the null hypothesis can only be rejected. A test never accepts the null hypothesis. Instead, the test fails to reject. That is the case if insufficient statistical evidence exists to make a confident decision about rejecting the null hypothesis. In the context of the mean average precision, a test fails to reject if the sets of average precision obtained from two retrieval systems are too close in the numerical sense to reject the null hypothesis. For that case, it is assumed that the average precision values of the two sets compared represent an almost equal performance.

In the scope of this thesis, statistical significance is tested by applying the permutation test [SAC07; ULH19; Goo04]. As suggested by [SAC07], the permutation test is suitable for testing the statistical significance of information retrieval systems. This test has also been applied in other works to analyze word spotting systems [SF18; RWF21]. In [SAC07], different tests are compared to determine the significance of the performance difference between two information retrieval systems. Among them are the *Student's paired t-test* and the *bootstrap test* [ET93]. All three tests mentioned show similar performance in the experiments reported by [SAC07]. However, the paired t-test assumes that the performance values of both systems are randomly sampled from a normal distribution, which implies "normality". The bootstrap test [ET93], on the other hand, assumes that the performance values from both systems are randomly sampled from the same distribution. In contrast, the permutation test is *non-parametric* and makes no assumption about the random sampling from a population [SAC07].

In the context of information retrieval, the basic idea of the permutation test is that if system A and B perform equally, the large majority of permutations of their respective sets of average precisions ($\mathbf{AP}_\mathrm{A}$ and $\mathbf{AP}_\mathrm{B}$) obtained from the same set of queries $\mathbf{Q} = \{q_1, q_2, q_3, ..., q_\mathsf{Q}\}$ will lead to (almost) equal mean average precision values. The permutation test evaluates the probability of permutations that yield equal or greater mAP differences than the *observed* difference $\mathrm{mAP}_{obs} = \mathrm{mAP}_\mathrm{A} - \mathrm{mAP}_\mathrm{B}$. This probability represents the p-value, and the test rejects the null hypothesis if the p-value drops below the significance level α. In other words, if the observed difference is unlikely to occur by chance, then both performances can be considered significantly different. A test statistic is required to compute the p-value. As shown in Equation 72, the test statistic d_{ts} for a two-sided test is defined as the absolute difference between mAP_A and mAP_B which are obtained from the two systems to be tested [SAC07].

$$d_{ts}\left(\mathbf{AP}_\mathrm{A}, \mathbf{AP}_\mathrm{B}\right) = \left| \frac{1}{\mathsf{Q}} \sum_{i=1}^{\mathsf{Q}} \mathrm{AP}_\mathrm{A}(q_i) - \frac{1}{\mathsf{Q}} \sum_{j=1}^{\mathsf{Q}} \mathrm{AP}_\mathrm{B}(q_j) \right| . \tag{72}$$

The probability given by the p-value is statistically modeled through a distribution obtained from applying the test statistic to all possible permutations of the two sets $\mathbf{AP}_\mathrm{A}$ and $\mathbf{AP}_\mathrm{B}$. Therefore, both sets are joined into a new set $\mathbf{AP}_\mathrm{AB}$ of size $2\mathsf{Q}$ which contains all average precision values from $\mathbf{AP}_\mathrm{A}$ and $\mathbf{AP}_\mathrm{B}$ as shown in the following:

$$\mathbf{AP}_\mathrm{AB} = \{\mathrm{AP}_A(q_1), ..., \mathrm{AP}_A(q_\mathsf{Q}), \mathrm{AP}_B(q_1), ..., \mathrm{AP}_B(q_\mathsf{Q})\} .$$

In each iteration, two disjoint permutations $\mathbf{P}_\mathrm{A}$ and $\mathbf{P}_\mathrm{B}$ of size Q are sampled and subsequently the test statistic $d_{ts}\left(\mathbf{P}_\mathrm{A}, \mathbf{P}_\mathrm{B}\right)$ is applied. It is important to emphasis that both sets $\mathbf{P}_\mathrm{A}$ and $\mathbf{P}_\mathrm{B}$ are disjoint and only contain unique set items from $\mathbf{AP}_\mathrm{AB}$. So that, for example, $\mathrm{AP}_\mathrm{A}(q_1)$ is contained either in $\mathbf{P}_\mathrm{A}$ or $\mathbf{P}_\mathrm{B}$ but never in both sets within the same iteration. The exact permutation test requires all possible permutations to be evaluated. The total number of permutations, however, grows rapidly with the sample size, i.e., $2\mathsf{Q}^{2\mathsf{Q}}$. In practice, excessive computations are avoided, and the p-value is approximated by sampling an adequate number of permutations. According to [SAC07], an estimate $\hat{p}$ of the actual p-value can be derived following a Monte-Carlo sampling. The standard deviation p_{std} of estimate $\hat{p}$ is directly related to the number of permutations K with p being the *true* p-value:

$$\hat{p}_{std}^2 = \frac{p\,(1-p)}{\mathsf{N}_{perm}} \;\Rightarrow\; \mathsf{N}_{perm} = \frac{p\,(1-p)}{\hat{p}_{std}^2}\;. \tag{73}$$

A small $\hat{p}_{std}$ indicates an accurate estimation of p, which in turn requires a large number of permutations [SAC07]. While in Equation 73, the true p-value is unknown, the minimum number of permutations required to stay below the standard deviation $\hat{p}_{std}$ for any p-value can be computed by using $p = 0.5$ which maximizes $p\,(1-p)$. The total number of permutations N_{perm} given a desired standard deviation $\hat{p}_{std}$ can then be obtained by applying the following equation:

$$\mathsf{N}_{perm} \;=\; \frac{1}{4\,\hat{p}_{std}^2}\;. \tag{74}$$

For the experiments in this thesis, the same parameterization is used as described in [Sud18; Rot19]. A precise approximation of the p-value is obtained by setting the maximum standard deviation to $\hat{p}_{std} = 0.001$ [Sud18]. This results in a total number of $\mathsf{N}_{perm} = 250\,000$ permutations. Algorithm 1 summarizes the general procedure in assessing the significance of performance difference between two retrieval systems used by the permutation test. First, the observed mAP difference d_{obs} between the two given average precision sets $\mathbf{AP}_\mathrm{A}$ and $\mathbf{AP}_\mathrm{B}$ is computed (line 3). The algorithm iterates $\mathsf{N}_{perm} - 1$ times in order to obtain two permutation sets $\mathbf{P}_\mathrm{A}$ and $\mathbf{P}_\mathrm{B}$ which have never been sampled in previous iterations. This behavior is ensured by hashing the concatenated byte representation h of both sets (line 9) and appending this hash value to the hash value set $\mathbf{H}$ (line 11). The algorithm leaves the while loop as soon as a new byte representation is found (lines 10-13) and subsequently computes the mAP difference d_{perm} between the two sampled permutations (line 15). The counter n is increased if the difference in mAP between the permutations d_{perm} is equal or greater than the difference observed for the actual two retrieval systems d_{obs} (line 16-18). After all N_{perm} permutations have been evaluated, the estimated p-value $\hat{p}$ is computed (line 20) and compared to the significance level α (line 21). If $\hat{p}$ is smaller than the predefined significance level α, the null hypothesis is rejected, which means that the observed difference d_{obs} is significantly different from the statistical point of view. A $\hat{p}$ value above the significance level α indicates that either d_{obs} is likely enough to be reproduced by random permutations of both AP sets or that insufficient statistical evidence exists to reject the null hypothesis. In both cases, the test fails to reject.

Algorithm 1 Determining the significance of difference in mAP between two word spotting methods using the permutation test

Initialize counter $n \leftarrow 0$
Initialize empty hash set $\mathbf{H}$
Compute observed mAP difference $d_{\text{obs}} \leftarrow d_{ts}\left(\mathbf{AP}_{\text{A}}, \mathbf{AP}_{\text{B}}\right)$
for i in $\{1, 2, 3, ...\mathsf{N}_{perm}\}$ **do**
    while true **do**
        $\mathbf{P}_{\text{AB}} \leftarrow$ shuffle $\mathbf{AP}_{\text{AB}}$
        $\mathbf{P}_{\text{A}} \leftarrow$ take first half of elements from $\mathbf{P}_{\text{AB}}$
        $\mathbf{P}_{\text{B}} \leftarrow$ take second half of elements from $\mathbf{P}_{\text{AB}}$
        $h \leftarrow$ compute a hash value for the concatenated byte representation of $\mathbf{P}_{\text{A}}$ and $\mathbf{P}_{\text{B}}$
        if h not in $\mathbf{H}$ **then**
            $\mathbf{H} \leftarrow$ appends hash value h to hash set $\mathbf{H}$
            leave while loop
        end if
    end while
    Compute mAP difference for permutations $d_{\text{perm}} \leftarrow d_{ts}\left(\mathbf{P}_{\text{A}}, \mathbf{P}_{\text{B}}\right)$
    if $d_{\text{perm}} >= d_{\text{obs}}$ **then**
        $n \leftarrow n + 1$
    end if
end for
Compute the estimated p-value $\hat{p} \leftarrow \frac{n}{\mathsf{N}_{perm}}$
if $\hat{p} < \alpha$ **then**
    reject null hypothesis $\rightarrow$ difference in mAP is statistically significant
else
    fail to reject
end if

8.3 EXPERIMENTS CONFIGURATIONS

This section presents the configurations used for the experiments in Section 8.4 and 8.5. The training, retrieval, and query sets used for the word spotting and cuneiform sign spotting benchmarks are described in Section 8.3.1. After that, the convolutional neural network architectures used in the experiments are discussed in Section 8.3.2. Finally, Section 8.3.3 explains the preprocessing applied to the images, the regularization techniques included in the training procedure, and the hyperparameters set for the optimizer.

8.3.1 *Evaluation Protocol*

In order to make the results presented in Section 8.4 comparable to other approaches, the allocation of train, test, and query samples is chosen to correspond precisely or as closely as possible to other works from the literature. The George Washington database has no official partitioning. Therefore, the allocation of word image samples proposed by Almazán *et al.* [Alm+14b] is applied. The authors randomly assign the database with 4860 word images in 4 equally sized sets of 1215 samples and perform a four-fold cross-validation.

All word instances that occur at least twice are considered queries for query-by-example retrieval. In the query-by-string scenario, all unique word transcriptions from the retrieval set are used as queries. The IAM-DB [MB99] provides an official partitioning into training, validation, and test set. For the experiments in Section 8.4, all samples from the 115 320 annotated word images labeled as erroneous are discarded, which results in a total amount of 96 422 word images. The authors in [Alm+14b] use the official *writer independent text line* partitioning that splits the document pages in 6 161 training, 1 840 validation, and 1 861 handwritten text lines for testing purposes. In this thesis, only the training and test line sets are used, which yields 60 453 samples for training and a retrieval set size of 13 752. Table 1 shows the number of training samples used to obtain the network model parameters and the retrieval set size for the GW splits and IAM-DB.

Table 1: Number of training, retrieval and query samples for the cross-validation split on the GW database and IAM-DB.

set	GW_1	GW_2	GW_3	GW_4	IAM-DB
training	3 645	3 645	3 645	3 645	60 453
retrieval	1 215	1 215	1 215	1 215	13 752
QbE	910	888	910	913	4 030
QbS	471	488	481	471	2 743

For the two benchmarks, Botany and Konzilsprotokolle, the word spotting competition aims to evaluate the word spotting methods on different scripts, primarily focusing on the required amounts of training data [Pra+16]. Therefore, the competition defines benchmarks for the two collections, each consisting of 20 document pages. For segmentation-based word spotting, the retrieval sets of Botany and Konzilsprotokolle contain 3 318 and 3 891 annotated word images, respectively. Both benchmarks provide 101 textual string queries for the QbS retrieval scenario. For QbE, 150 and 200 word images are selected as visual queries for Botany and Konzilsprotokolle, respectively. Three sets with a growing number of training samples are provided to evaluate retrieval models trained on different amounts of training samples. Table 2 shows the total number of training samples, the corresponding retrieval set, and the respective amount of QbE and QbS queries.

Table 2: Number of word images for the respective training, retrieval and query partitions of the Botany and Konzilsprotokolle datasets.

set	Botany	Konzilsprotokolle
Train I	1 683	1 849
Train II	5 289	7 816
Train III	21 964	16 918
retrieval	3 318	3 891
QbE	150	200
QbS	101	101

Both cuneiform benchmarks, Hittite and Old Assyrian, do not provide an official partitioning. A custom assignment for training, retrieval, and query sets is applied for the experiments in Section 8.5. Analogous to the GW benchmark, the partitioning scheme assigns images of cuneiform tablets to one out of four splits to perform a four-fold cross-validation. Therefore, the assignment algorithm iterates through all tablet images and assigns all annotated samples from one image to the cross-validation split containing the fewest sign samples. This approach guarantees to assign one cuneiform tablet page to only one cross-validation split while at the same time yielding an approximately uniform distribution of samples across the slits. Table 3 shows the resulting training, retrieval, and query samples obtained from the custom assignment.

Table 3: Number of cuneiform sign images for the respective training, retrieval and query partitions in each of the four cross-validation splits for the Hittite and Old Assyrian datasets.

set	Hittite				Old Assyrian			
	1	2	3	4	1	2	3	4
training	75 430	75 459	75 102	75 119	4 470	4 485	3 997	4 283
retrieval	24 940	24 911	25 268	25 251	1 275	1 260	1 748	1 462
QbE	24 913	24 885	25 247	25 233	1 248	1 246	1 731	1 441
QbS	267	273	262	263	122	115	122	122

In the word spotting benchmarks, word images or textual strings are relevant concerning the query if the Levenshtein [Lev66] (or string edit) distance is zero. All queries and word labels are converted to their lowercase, as only case-insensitive retrieval scenarios are performed. For all experiments, only uni-grams are defined as semantic attributes, while higher n-grams, such as bi- or tri-grams, are not included. These uni-grams are extracted from all unique transcriptions for a given benchmark. Table 4 show the number of uni-grams for each word spotting benchmark. The word images and textual strings of the GW benchmark are represented by 36 uni-grams consisting of 24 characters from the Latin alphabet and 10 digits. The uni-gram set for IAM-DB additionally contains the special character "–" representing punctuation marks. Botany is represented by 51 uni-grams, including the Latin alphabet, 10 digits, and 15 special characters. Similar to Botany, Konzilsprotokolle contains 52 unique uni-grams consisting of the Latin alphabet, 10 digits, 11 special characters, and additionally the German letters *ä,ö,ü,ß*.

Table 4: The number of unique uni-grams extracted from the word string annotations of each word spotting benchmark.

	GW	IAM-DB	Botany	Konzilsprotokolle
# unigrams	36	37	51	52

A sign is considered relevant for the cuneiform benchmarks according to their annotated sign class. This consideration means that two different sign classes are irrelevant to each other despite being represented by the same attribute vector. A binary-valued semantic attribute represents one count of a wedge type as described in Section 7.3. These individual wedge counts obtained from the expression-based annotations (cf. Section 7.3.1) of the Hittite and Old Assyrian databases are shown in Table 5.

Table 5: Individual wedge counts (a,b,c,d) and additional attributes (e) in the SPG and TPG encodings of the Hittite and Old Assyrian databases.

wedge semantic	SPG		TPG	
	Hittite	Old Assyrian	Hittite	Old Assyrian
a	1−10	1−9	1−10	1−9
b	1−10	1−9	1−6, 8, 9	1−7, 9
c	1−9, 11, 12	1−9	1−9	1−6
d	1, 2	1−3	1,2	1, 2
e	–	–	1−4	1−4

The retrieval performances of CNN models are displayed as mean average precision values obtained for a given benchmark (see Section 8.2.1). As described in Section 8.2.2, a permutation test is applied to determine the significance between two sets of AP values obtained from two different models. For that, a standard deviation of 0.001 is selected for all experiments presented in Section 8.4 and Section 8.5. Using this value results in a probability more than 99% for the true p-value being in the interval $\hat{p} \pm 0.003$. The test requires 250 000 random permutations to obtain the desired standard deviation (cf. Section 8.2.2). In order to perform the significance test, the individual average precision values are required. Therefore, only the results obtained from the experiments in this thesis are evaluated for their performance significance.

8.3.2 *Model Architectures*

The convolutional neural network architectures used in the experiments (Section 8.4 and Section 8.5) are based on the PHOCNet proposed in [SF18] and the PHOCResNet as presented in [Sud18]. The PHOCNet is inspired by the VGG_{16} architecture [SZ15] with 13 convolutional layers, followed by a multi-layer Perceptron with two fully connected layers. The network consistently uses a kernel size 3×3 for all convolutional layers with an increasing number of feature maps 2×64, 2×128, 3×256, 3×256, 3×512. In contrast to the 512 feature maps in the original VGG_{16} architecture, the PHOCNet only uses 256 feature maps in the penultimate convolutional layers block. The number of subsampling (i.e., poolings) applied in the convolutional part of the network is significantly different. Sudholt *et al.* argues that preserving the spatial position of characters is crucial. Thus, maximum pooling is applied only after the second and fourth convolutional layers. For

comparison, the original VGG architectures subsample the feature maps five times after the second, fourth, seventh, tenth, and thirteenth convolutional layer [SZ15]. As presented in [Sud18], the PHOCResNet make use of the ResNet_{50} [He+16] architecture with 49 convolutional layers. This network model uses the same number of feature maps and sizes for the convolution kernels. Similar to the PHOCNet, pooling is only applied in the first convolutional layer using a stride of 2, followed by a maximum pooling equal to the original ResNet architecture, which effectively subsamples the input image by a factor of 2.

For word spotting, the feature maps obtained from the convolutional layers are transformed through a *temporal pyramid pooling* (TPP) [SF17] using the levels 1, 2, 4, and 8 to obtain a feature vector of fixed dimensionality (cf. Section 5.2). The TPP yields a 7 680 dimensional vector if the VGG architectures and the smaller versions of the ResNet architecture (18 and 34) are used. For the larger ResNet models with 2 048 feature maps in the last convolutional layer, a 30 720 dimensional feature vector is obtained. In the case of cuneiform sign spotting, SPP [He+15b] is applied using a grid-like partitioning of 1×1, 2×2 and 3×3, yielding 14 pyramid splits. The reason is that cuneiform signs do not strictly follow a sequential order in the horizontal direction as handwritten texts in Latin scripts. The SPP feature vectors have a dimensionality of 7 168 when using the SPP-PHOCNet.

Afterwards, the Multi-Layer Perceptron converts the feature vector into the attribute representation. The Multi-Layer Perceptron is made up of two fully connected layers, each containing 4 096 neurons. In Section 8.3.2, the MLP is substituted with a Single-Layer Perceptron to compare the performance difference between using only one layer and using three fully connected layers.

8.3.3 *Training Setup*

All convolutional neural networks used in the experiments (Section 8.4 and 8.5) are trained end-to-end given the word or cuneiform images and their respective label vectors. No pre-training or fine-tuning is applied. All word and cuneiform sign images are converted to grayscale, as color does not contribute more information about an image's visual content. The image pixels are divided by 255 to rescale their value range from $[0, 255]$ to $[0.0, 1.0]$. In addition, images of handwritten words are inverted using $1.0-x$, where x represents a pixel value. The images are inverted based on the assumption that handwritten texts mainly consist of a white paper background and a black ink foreground. If one is interested in a model that learns to infer text consisting of darker-colored pen strokes, inverting pixel values yields images where pen strokes are represented by values close to 1.0 while the paper background is represented by values close to 0.0. Due to their large amount of trainable parameters, network models tend to overfit. For this reason, neurons are dropped with a probability of 50% in the fully connected layers (cf. Section 2.2.4). Dropout is applied to the pyramid pooling output using the SLP and to the output of both hidden layers if the MLP is used. In addition to dropout, different augmentation techniques are applied in order to increase the visual variability of the word and cuneiform sign images. When manipulating image pixels, it is essential to make sure that the semantic content is

Table 6: Total number of training iterations carried out per benchmark. The initial learning rate is 10^{-4}. The right column shows the iteration at which the learning rate is set to the value displayed by the column label. All iteration values are shown in thousands [$\times 10^3$].

benchmark	training iters.	learning rate	
		10^{-5}	10^{-6}
GW	100	50	
IAM-DB	200	50	150
Botany	120	50	100
Konzilsprotokolle	120	50	100
Hittite	150	50	100
Old Assyrian	150	50	100

preserved. Therefore, transformations like *flipping* or *mirroring* are not applied. Instead, images are slightly *sheared*, *rotated*, *scaled*, or transformed in their *perspective*. Moreover, random areas of the images are *cut out* (cf. Section 2.2.4). The augmentation techniques are applied to 50% of the training samples while the rest remains unchanged. All convolutional neural network models are trained using the *Adam* [KB15] optimizer. Adam is an established optimization algorithm and has been successfully utilized in various training scenarios [GG16; Kes+17; ZL17; SF18]. This optimizer combines the advantages of both *adaptive gradient* (AdaGrad) [DHS11] and *root mean square propagation* (RMSProp) [TH+12]. Following the recommendation from [KB15], both betas β_1 and β_2 are set to 0.9 and 0.999, respectively. The *momentum* value is set to 0.9 and weight decay to 10^{-5}. The *learning rate* is initialized with 10^{-4} and divided by 10 after a certain amount of training iterations, depending on the benchmark. Table 6 displays the *total number of training iterations* executed for each benchmark in order to obtain one CNN model. The two columns on the right side in Table 6 show the training iteration at which the learning rate is divided by 10. Smaller datasets, like GW, require fewer training iterations to converge, while large ones, such as IAM-DB, require a longer training run. The total amount of iterations required is determined empirically through preliminary experiments. For all benchmarks, additional training does not improve the retrieval performance beyond the iterations shown in Table 6. All CNN models are trained with a batch size of 10; however, the images are not rescaled to uniform sizes, which is required to perform batch training. Handwritten word images have considerable variability in height and width. Forcing these images into an equal size will squash words containing many characters or stretch word images with few characters. Instead, the 10 images are forwarded through the network separately while the resulting gradients are accumulated and divided by 10. This approach emulates a training with a batch size of 10.

8.4 RESULTS FOR WORD SPOTTING

This section presents the performance results obtained from the experiments conducted on four benchmarks. The following section (Section 8.4.1) shows results obtained applying the PRM and presents a comparison to other similarity measures commonly used for word spotting. Subsequently, in Section 8.4.2 qualitative results are presented. Section 8.4.3 investigate the influence of different components from the CNN architecture. Finally, Section 8.4.4 compares the achieved performances to the results reported in related works from the literature.

8.4.1 *Comparison of Similarity Measures*

For retrieval-based systems, similarity measurements are essential components. To compare the influence on performance using different measures, a TPP-PHOCNet is trained to estimate PHOC representations using the levels 1, 2, 4, and 8. Two network models are trained on the GW and IAM-DB datasets using the Almazan protocol (cf. Section 8.3.1). The best performance values are highlighted in bold, and significant differences are indicated in italics. Both models are trained using the BCE loss and estimate the PHOC

Table 7: Comparison of QbE and QbS retrieval performance obtained for the GW and IAM-DB benchmarks using a TPP-PHOCNet trained with the BCE loss. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

similarity	GW		IAM-DB	
	QbE	QbS	QbE	QbS
PRM	**97.81**	**98.71**	81.52	**92.98**
Cosine	97.65	97.64	**81.93**	*90.21*
Euclidean	97.69	97.86	81.62	*90.20*
Cityblock	97.51	*96.65*	*79.67*	*84.72*

representations in the query-by-example and query-by-string retrieval scenarios for their respective benchmark. Once these representations are obtained, four different measures, the probabilistic retrieval model (PRM), Cosine similarity, Euclidean distance, and Cityblock distance, are applied to retrieve the relevant word images. In word spotting, the Cosine similarity and the Euclidean distance are well-known measurements often used to perform retrieval tasks [Gio+17; Alm+14b; WB16; SF17; KDJ18] (cf. Chap. 5). In addition to the similarity measures mentioned, the Cityblock distance is used to assess similarities. The results displayed in Table 7 show that the PRM achieves superior performance compared to the other measures. Generally, the results are similar except for the performance achieved for IAM-DB using the Cityblock distance (79.67% and 84.72%) in both retrieval

scenarios. For QbE, the performance achieved by the PRM is almost equal to the one using the Cosine similarity or Euclidean distance on both benchmarks. However, for QbS, the results obtained using the PRM diverge by nearly one percentage point (98.71 to 97.64/97.86) on the GW and by about two percentage points (92.98 to 90.21/90.20) on the IAM-DB benchmark compared to the performance achieved by the Cosine similarity or Euclidean distance. The permutation test validates that the PRM similarity performs significantly better on IAM-DB in the QbS retrieval compared to the other measures. From the results presented in Table 7, it can be concluded that the retrieval performance of binary-valued attribute representations benefits from the PRM similarity measure. As

Table 8: Comparison of QbE and QbS retrieval performance obtained for the GW and IAM-DB benchmarks using a TPP-PHOCNet trained with corresponding loss functions and similarity measures. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

loss	similarity	GW		IAM-DB	
		QbE	QbS	QbE	QbS
BCE	PRM	**97.81**	**98.71**	**81.52**	**92.98**
Cosine	Cosine	97.70	97.40	81.24	*90.33*
MSE	Euclidean	97.29	97.59	80.33	*90.87*
MAE	Cityblock	97.42	*95.76*	*66.02*	*76.41*

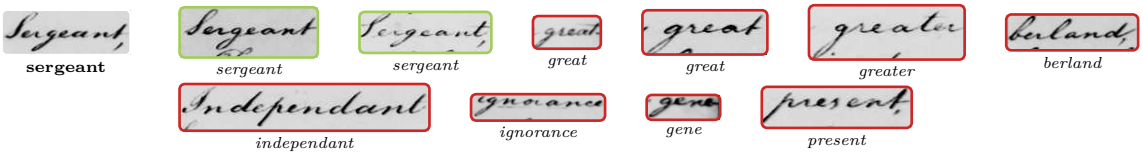
previously explained in the methodology (Section 6.2), it is possible to derive a suitable loss function for training a neural network given the similarity measure. For the probabilistic retrieval model, the binary cross-entropy can be considered as the corresponding loss function (cf. Section 6.2.2). Sudholt *et al.* [SF18] show that the Cosine loss corresponds to the Cosine similarity. For this reason, the evaluation performed in Table 8 presents different performance results obtained from eight TPP-PHOCNet models (four per benchmark) trained using four different loss functions and their respective similarity measures. In addition to the BCE and Cosine loss, the *mean squared error* (MSE)[1] as well as the *mean absolute error* (MAE)[1] is used in combination with the Euclidean and Cityblock distance, respectively. It is important to note that only the combination BCE/PRM uses a sigmoid function to scale the network output, while the other combinations do not scale the estimated output values. The results from Table 8 sketches a similar pattern as shown in Table 7. The best results are achieved if the network is trained using binary cross-entropy and the probabilistic retrieval model similarity measure. Especially for the QbS scenario, the differences in performance are significantly higher.

1 Both loss functions are implemented in the Python machine learning framework *PyTorch* (www.pytorch.org) version 1.12.

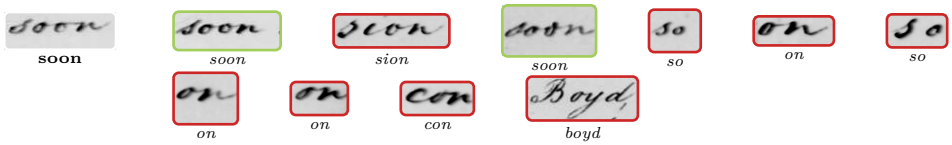
8.4.2 *Qualitative Results and Attribute Statistics*

In addition to the quantitative performance numbers presented in the previous section, qualitative results and attribute statistics are discussed in the following. On the Pages 104 to 105, Figures 35 to 38, visualize selected results for the QbE and QbS retrieval scenario achieved on the GW and IAM-DB benchmarks. The results are obtained by the TPP-PHOCNet models shown in Table 7 trained using the BCE loss and assessing similarities by applying the PRM. All Figures show the top 10 retrieved candidates for a given query, with relevant candidates marked by green boxes and irrelevant candidates marked by red boxes. The queries are selected from higher and lower average precision values to present a broader range of retrieval results. Regarding the average precision, the two influencing factors are the position and the number of occurrences of all relevant candidates (cf. Section 8.2). The influence is clearly visible for the two queries *soon* and *greatly*. For *soon*, the second candidate is wrongly retrieved at the third position. This deviation by one position decreases the AP to 83.33%. This effect is even higher for the query *greatly* as only one relevant candidate has to be retrieved. The AP is halved to 50%, as the relevant word image is retrieved on the second position in the list.

Figure 39 on page 106 shows attribute statistics for the retrieval sets from the GW and IAM-DB benchmarks. The blue bars indicate the mean absolute differences between the attribute label a and the estimated value $\hat{a}$, denoted as $|a - \hat{a}|$. It is important to note that only attributes labeled as 1 are evaluated. The reason is that the absence of attributes (i.e., 0) by far dominates the frequencies, and the network models estimate zero-labeled attributes very well. The absence of attributes will lead to statistics showing significantly better predictions and distorting the actual estimation quality. The orange bars quantify the maximum normalized frequency of attributes $\|fr(a)\|_{max}$. Figure 39a and 39c visualize the statistics per attribute across all pyramid splits while Figure 39b and 39d depict mean attribute values from the different pyramid splits of the levels $1, 2, 3$ and 4. In general, it is noticeable that in Figure 39a and 39c, letters occur more often than digits (orange bars) and are better estimated (blue bars) as well. However, the network struggles to estimate particular letters such as k, q, and x on the GW benchmark, as well as the letters j and q on the IAM-DB benchmark. Investigating the statistics per pyramid split, Figure 39b and 39d show that attributes across different splits are estimated approximately equal (blue bars). For the GW benchmark (Figure 39b), the prediction quality is lower for the most right spits in the pyramid levels. The reason is that the last letters within a word are either written sloppy or similar between many different word classes. For the IAM-DB dataset, the quality between attribute estimates on different pyramid splits is almost equal and differs only marginal.

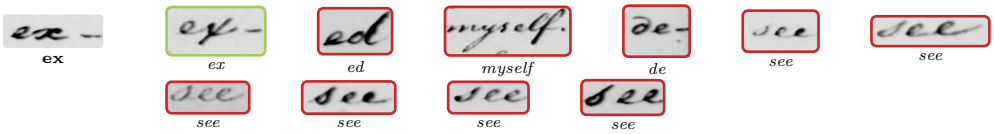


(a) query: *sergeant*, number of relevant candidates: 3, AP: 100.00%,



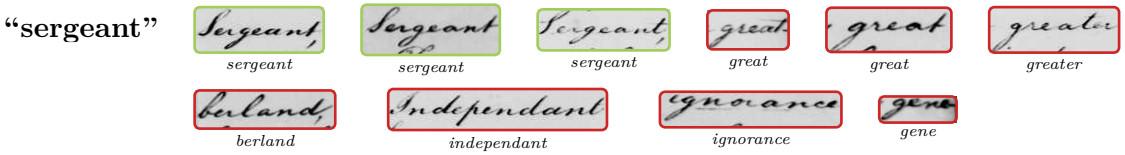
(b) query: *soon*, occurrences: 2, AP: 83.33%

(c) query: *greatly*, occurrences: 1, AP: 50.00%

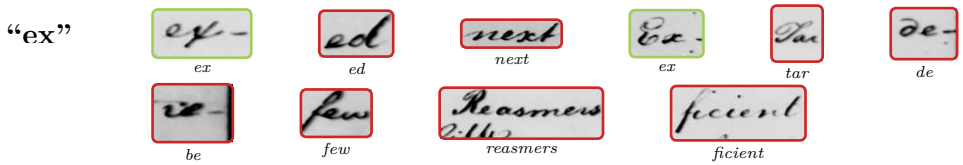


(d) query: *ex*, occurrences: 2, AP: 8.33%

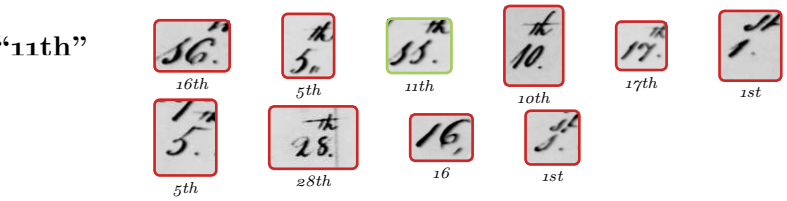
Figure 35: Visualization of QbE retrieval results from the GW database.



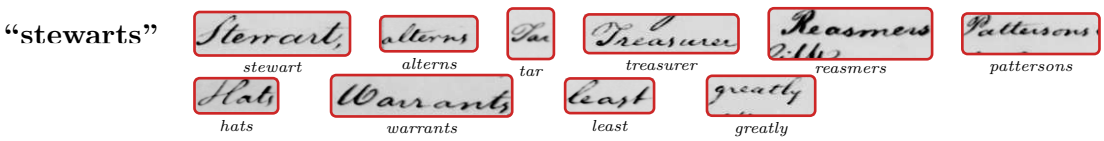
(a) query: *sergeant*, occurrences: 3, AP: 100.00%,



(b) query: *ex*, occurrences: 2, AP: 75.00%

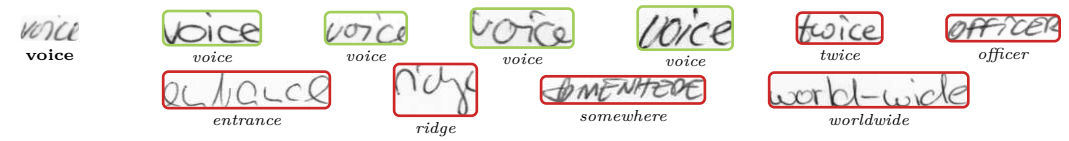


(c) query: *11th*, occurrences: 1, AP: 33.33%

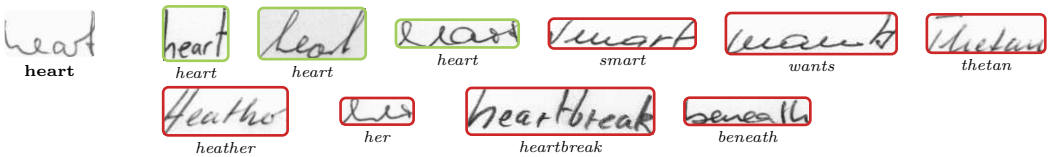


(d) query: *stewarts*, occurrences: 1, AP: 4.00%

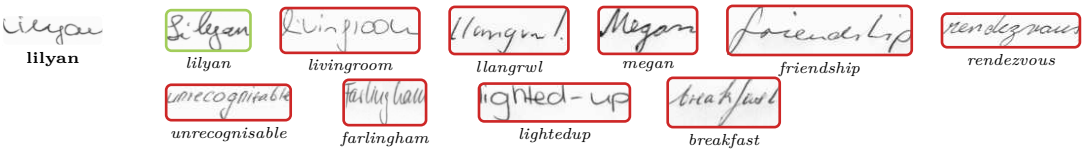
Figure 36: Visualization of QbS retrieval results from the GW database.



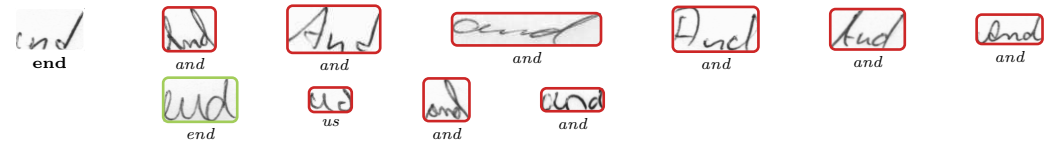
(a) query: *voice*, occurrences: 4, AP: 100.00%,



(b) query: *heart*, occurrences: 4, AP: 79.00%



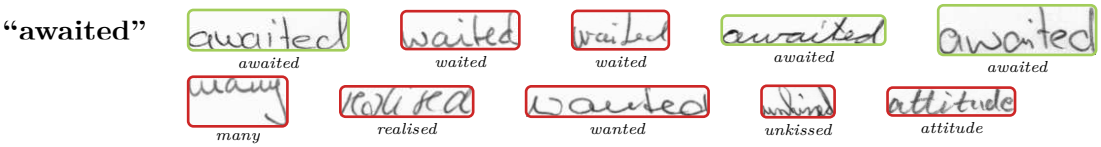
(c) query: *lilyan*, occurrences: 3, AP: 38.52%



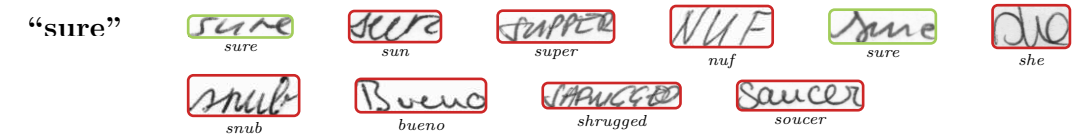
(d) query: *end*, occurrences: 3, AP: 5.30%

Figure 37: Visualization of QbE retrieval results from the IAM-DB database.

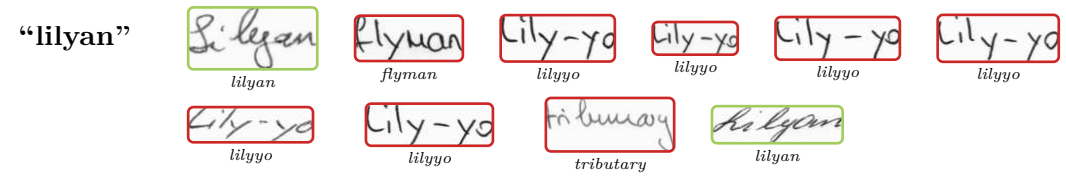
(a) query: *across*, occurrences: 3, AP: 100.00%,



(b) query: *awaited*, occurrences: 3, AP: 70.00%

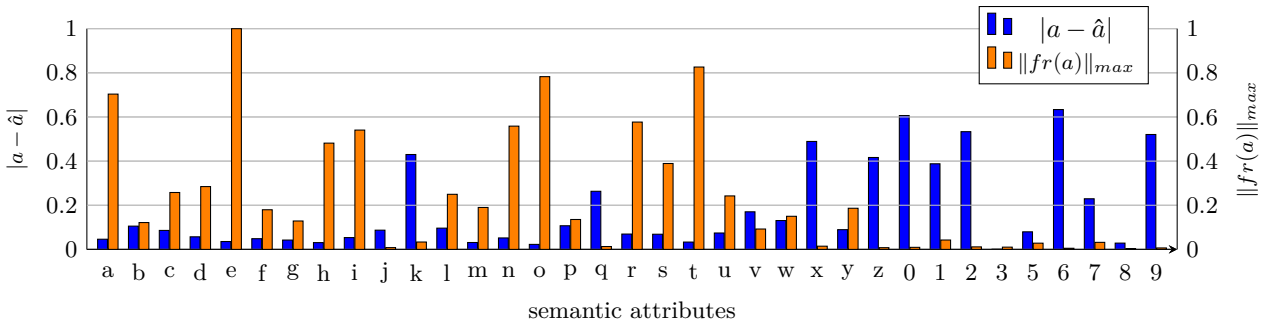


(c) query: *sure*, occurrences: 3, AP: 47.40%

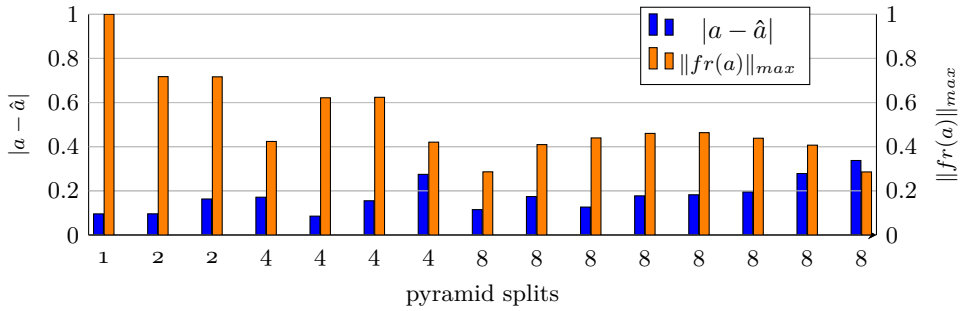


(d) query: *lilyan*, occurrences: 2, AP: 38.58%

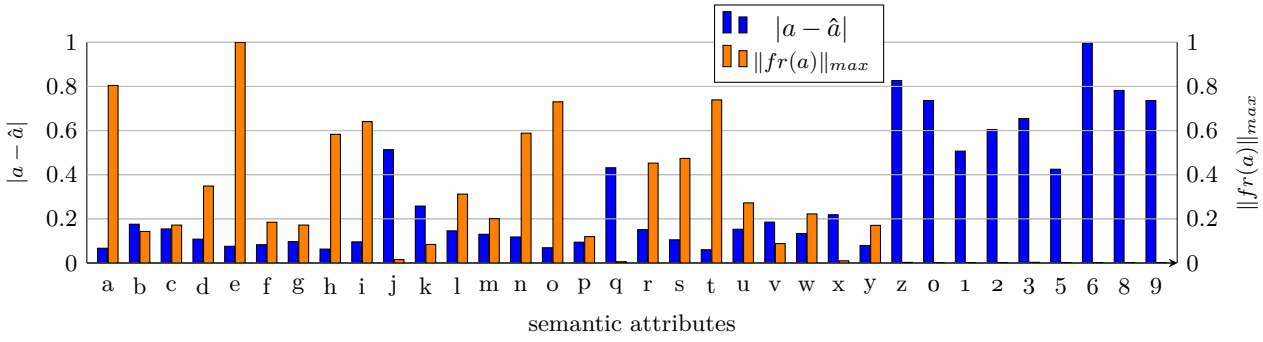
Figure 38: Visualization of QbS retrieval results from the IAM-DB database.



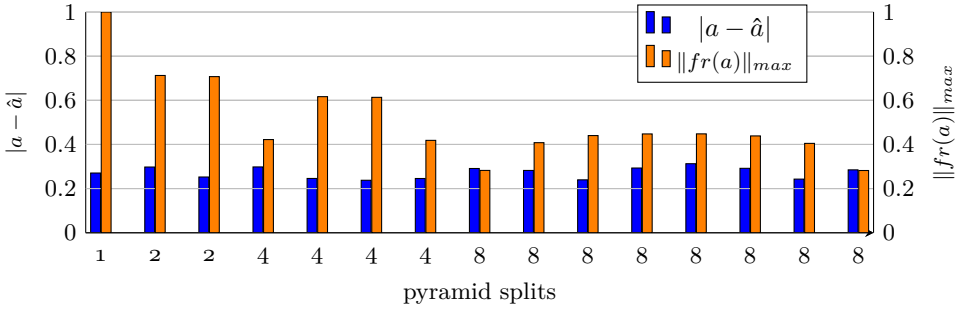
(a) GW database.



(b) GW database.



(c) IAM-DB.



(d) IAM-DB.

Figure 39: Bar chart shows the mean absolute error $|a-\hat{a}|$ between label a and estimate $\hat{a}$ (blue bars) as well as the maximum normalized attribute frequency $\|fr(a)\|_{max}$ (orange bars).

8.4.3 *Influence of Architectural Components*

The quality of retrieval primarily depends on predicting attributes. Convolutional neural networks provide this predictive quality by transforming a given image into its corresponding attribute representation. This section evaluates three crucial components of a CNN architecture: the structure of convolutional layers, the number of subsampling typically applied to reduce the feature dimensionality between two consecutive convolutional layers, and the importance of pyramid pooling and the necessity of a Multi-Layer Perceptron.

Convolutional Neural Network Architecture

Table 9 show the results achieved for different versions of the VGG [SZ15] and ResNet [He+16] architecture. Both architectures are modified to resemble the TPP-PHOCNet used in Section 8.4.1. For that, all subsampling layers except the first two are discarded from the VGG and ResNet networks. Equal to the PHOCNet architecture, a maximum pooling after the first and second block of convolutional layers is used with a kernel size of 2×2 and a stride of 2×2 (cf. [SF16; SF17]). The ResNet architecture subsamples the feature dimensions in width and heights by applying a stride of 2×2 in the first convolutional layer, followed by a maximum pooling using a kernel window of 3×3 shifted by 2×2 positions in both directions (cf. [He+16]). For both architectures, a temporal pyramid pooling followed by an Multi-Layer Perceptron is applied after the convolutional layers described in Section 8.3.2. All networks in Table 9 are trained according to the Almazan protocol (see Section 8.3.1) while the word images are presented by PHOC vectors using the pyramid levels $1, 2, 4$ and 8.

The results presented in Table 9 show that the architectural choice on the GW benchmark does not influence the performance for query-by-example and query-by-string retrieval, at least not for the selected CNN architectures. The representational power of the $\text{VGG}_{\text{A(11)}}$ network (with the fewest parameters) is enough to achieve similar results compared to larger networks such as $\text{VGG}_{\text{E(19)}}$ or the ResNet models. According to the permutation test, the retrieval performance differences are not significant. In contrast, the results obtained on the IAM-DB benchmark show significant differences between the model sizes and architectures. If the depth and number of parameters are increased in the VGG architecture, the performance is significantly improved for both retrieval scenarios. On the other hand, looking at the results obtained from the ResNet architecture, the opposite picture emerges. Using deeper network models, such as the ResNet with 49, 100, or even 151 convolutional layers, leads to distinct inferior results compared to models with lower depth, like 17 or 33. It is important to note that additional training iterations beyond 200 000 (cf. Section 8.3.3) do not improve the performance for the larger ResNet models. Moreover, different configurations concerning the learning rate and weight decay did not boost the performance. Although it cannot be completely ruled out that a combination of training iterations and hyperparameters leads to superior results, the architectural structure is assumed to be responsible for the decline in performance. First

Table 9: Comparison of QbE and QbS retrieval performance obtained for the GW and IAM-DB benchmarks altering the network architectures. All models are trained using the BCE loss. Retrieval is perform applying the PRM as similarity measure. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

arch.	# conv. layers	GW		IAM-DB	
		QbE	QbS	QbE	QbS
$VGG_{A(11)}$	8	97.89	97.50	*76.99*	*91.06*
$VGG_{B(13)}$	10	98.14	98.17	*78.60*	*91.69*
$VGG_{D(16)}$	13	98.20	**98.32**	81.81	93.22
$VGG_{E(19)}$	16	98.42	98.22	**82.91**	**94.03**
$ResNet_{18}$	17	**98.46**	98.15	*80.01*	*92.62*
$ResNet_{34}$	33	97.41	97.18	82.21	93.09
$ResNet_{50}$	49	98.00	98.08	*78.35*	*91.37*
$ResNet_{101}$	100	98.24	97.42	*77.64*	*91.86*
$ResNet_{152}$	151	98.16	97.34	*72.08*	*88.90*

of all, the smaller models, such as $ResNet_{18}$ and $ResNet_{34}$ use two convolutional layers in each residual block, while the larger models make use of bottleneck layers to compress and expand the feature dimensions in every residual block (cf. Section 2.4). Furthermore, the number of feature maps after the last convolutional layers differ by a factor of 4 between the 512 feature maps in the smaller models and 2048 in the larger ones. Using a temporal pyramid pooling at the levels $1, 2, 4$ and 8 leads to feature vectors with a dimensionality of 7 680 in the smaller and 30 720 in the larger models. This, in turn, results in the first fully connected layer containing over 31.4 and 125.8 million parameters, respectively. It may be asked why the larger ResNet models achieve the same outcome as the smaller versions on the GW benchmark but show significantly lower performance on the IAM-DB. The difference in performance is due to the consistent visual appearance of word images in the GW database. In contrast, the IAM-DB contains handwritten text from multiple writers, leading to significant variation in the word images. Both the smaller and larger ResNet models are expected to show more intense overfitting on both datasets. However, overfitting to a specific visual appearance of a handwritten word impacts the retrieval performance in the IAM-DB negatively, while the same model behavior does not reduce retrieval performance on the GW benchmark.

Additional experiments are carried out to confirm this claim. As a result, the ResNet architecture is adjusted to reduce overfitting. The first modification involves reducing the number of pyramid levels in the TPP layer, using only levels one and two. For the second

Table 10: Comparison of QbE and QbS retrieval performance obtained for the GW and IAM-DB benchmarks using the modified ResNet architectures. All models are trained with the BCE loss. Retrieval is perform applying the PRM as similarity measure. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

arch.	# conv. layers	GW		IAM-DB	
		QbE	QbS	QbE	QbS
ResNet_{50} $(1,2)$	49	98.02	97.98	82.56	93.34
ResNet_{101} $(1,2)$	100	**98.12**	97.92	*84.15*	94.03
ResNet_{152} $(1,2)$	151	98.07	97.14	**84.58**	93.64
ResNet_{50} (drop)	49	98.05	**98.12**	*83.99*	**94.33**
ResNet_{101} (drop)	100	98.01	97.94	*83.93*	94.31
ResNet_{152} (drop)	151	98.06	98.01	81.95	93.63

modification, Dropout is applied to the output of the TPP layer. Table 10 shows the results achieved by the modified ResNet models. The suffix $(1,2)$ denotes the networks with reduced pyramid levels, and (drop) refers to the models where the additional Dropout is applied. Compared to Table 9, the performance achieved on the GW benchmark remains the same. However, significantly performance differences are achieved on the IAM-DB benchmark. Here, all models show distinct higher values compared to Table 9. These results clearly indicate that parameter overfitting is the reason for the lower performance.

Subsampling Convolutional Features

Convolutional neural networks typically contain subsampling steps to reduce the dimensionality of the feature tensors between two consecutive convolutional layers. In the context of CNNs, subsampling is widely known as pooling (cf. Section 2.3.2). In the following, subsampling and pooling are used as synonyms. The original VGG [SZ15] and ResNet [He+16] architectures apply five subsamplings, each dividing the feature dimensions in height and width by a factor of 2. This subsampling effectively reduces the input size by 2^5. However, in the original PHOCNet architecture [SF16; SF17], pooling is only applied after the first two blocks of convolutional layers, which effectively shrinks the input size by 2^2. One argument for using only two pooling steps in the PHOCNet architecture is to preserve the spatial information of a word image and divide the feature tensor of the last convolutional layer using the pyramidal pooling scheme. Another argument is the height and width of handwritten word images, which are typically smaller in size than image samples from other computer vision tasks.

The influence of subsampling on the retrieval performance is shown in Table 11. These results are obtained using the same TPP-PHOCNet architecture and PHOC representations

Table 11: Comparison of QbE and QbS retrieval performance obtained for the GW and IAM-DB benchmarks altering the number of subsampling steps. The TPP-PHOCNet is trained with the BCE loss. Retrieval is performed using the PRM as similarity measure. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

pooling positions	PHOCNet			
	GW		IAM-DB	
	QbE	QbS	QbE	QbS
1	96.89	*95.59*	*75.26*	*89.92*
1, 2	**97.81**	**98.71**	*81.52*	*92.98*
1, 2, 3	97.64	98.01	86.57	95.61
1, 2, 3, 4	96.69	97.29	**87.64**	**95.66**
1, 2, 3, 4, 5	96.85	*97.06*	*86.12*	95.30

described in Section 8.4.1. All models are trained using the evaluation protocol presented in Section 8.3.1 and a training setup described in Section 8.3.3. As the VGG architecture inspires the TPP-PHOCNet, subsampling is applied at the same positions as initially proposed for the $\text{VGG}_{D(16)}$ [SZ15]. These positions are shown in the far left column with the label *pooling positions*. For example, the numbers $1, 2, 3$ indicate that pooling is applied at the first, second, and third positions in the convolutional part of the network. The results obtained for the GW database show that pooling at position 1 and 2, as proposed for the PHOCNet [SF16; SF17], achieves the best performance. However, according to the permutation test, no significance could be determined between the performances in a QbE scenario and only two significantly different mAP values for QbS retrieval (i.e., 95.59% and 97.06%). Like the results presented in Table 9, the network models perform almost equally on the GW benchmark, independent of the number of pooling steps. In contrast, the mAP values obtained on IAM-DB show a more considerable variance. The first observation is that more pooling steps lead to significantly higher performance for both QbE and QbS. On the IAM-DB benchmark, retrieval performance is boosted by over 6 respectively 3 percentage points by simply applying two additional pooling steps at the positions 3 and 4 compared to the original TPP-PHOCNet architecture (QbE: 81.51% to 87.64% and QbS: 92.98% to 95.66%). These results are remarkable because the network architecture is not extensively modified.

Pyramid Pooling and Fully Connected Layers

Table 12 shows the results of different configurations concerning the pyramidal pooling and the number of fully connected layers used to map the feature vectors to their corresponding attribute representations. Every row depicts mAP values achieved for a specific

Table 12: Comparison of QbE and QbS retrieval performance obtained for the GW and IAM-DB benchmarks altering the pyramid pooling levels as well as the number of fully connected layers. All models are trained with the BCE loss. Retrieval is perform applying the PRM as similarity measure. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics. Underlined values are significantly lower compared to neighboring value along the row.

pyramidial pooling	GW				IAM-DB			
	QbE		QbS		QbE		QbS	
	SLP	MLP	SLP	MLP	SLP	MLP	SLP	MLP
1	*93.01*	98.00	*87.49*	97.35	*59.21*	*76.98*	*78.58*	*88.85*
1, 2	*96.37*	98.27	97.37	98.26	*68.00*	*80.88*	*87.95*	92.61
1, 2, 4	97.59	98.16	98.63	98.35	*71.23*	82.57	*89.92*	93.74
1, 2, 4, 8	97.49	97.81	98.57	**98.71**	**72.76**	*81.52*	**91.22**	92.98
1, 2, 3	97.42	**98.45**	98.26	98.32	*70.71*	*81.77*	*90.18*	**93.85**
1×6	97.62	*96.97*	**98.64**	97.92	71.82	**83.60**	*90.79*	93.76
2×3	**97.77**	97.68	98.49	97.57	*71.18*	82.91	*90.78*	93.51

partitioning scheme in the pyramidal pooling. Numbers separated by a comma denote a temporal pyramid pooling scheme where feature maps are only divided along the horizontal direction. The two bottom rows show a grid-like partitioning of the feature maps. For example, 2×3 describes a grid of heights 2 and width 3. The consideration behind the evaluation shown in the lower part of table Table 12 (three bottom rows) is to compare the retrieval performance between three models applying a pyramidal pooling that divides the feature maps in 6 splits, which results in a feature vector dimensionality of 3 072. In other words, all three models have the exact same number of parameters; however, they use a different partitioning scheme for the pooling step between the last convolutional and the first fully connected layer. The upper part of Table 12 (top four rows) depicts the influence on performance if a TPP with a growing number of levels is applied. It is important to note that more pyramid levels lead to more splits and, consequently, to more parameters in the first fully connected layer. Every column in Table 12 shows the achieved results either using an SLP or an MLP for the QbE and QbS retrieval scenario on both benchmarks. The SLP directly maps the output of the pyramid pooling layer to the attribute vector without any intermediate representations. Thus, if a PHOCNet uses the SLP, the model contains far fewer parameters than a model using the MLP. For example, using a TPP with the levels 1, 2, 4, and 8 and a representation containing 540 attributes, the parameters between the two mentioned models would differ by 46 299 136. The highest mAP value within a column is indicated in bold, and significantly lower performances are displayed in italics. Moreover, as the presented results are compared between the SLP and

MLP models, directly neighbored numbers are underlined if their performance significantly differs according to the permutation test. Performances obtained on the GW database are almost equal for both retrieval scenarios independent of the pooling scheme or the number of fully connected layers. For QbE, the MLP shows the same results except for the 1×6 pooling where the permutation test determines a significant difference between 96.97% and the best achieved result of 98.45%. Using the SLP to predict the attributes shows nearly identical results as the MLP. However, choosing a low number of pyramid levels (i.e., 1 or $1, 2$) harms the retrieval performance significantly (93.01% and 96.37%) compared to a higher number of pyramid levels such as $1, 2, 4$ or $1, 2, 4, 8$ achieving a mAP of 97.59% and 97.49%, respectively. Results obtained in the query-by-string scenario show more robustness across different pyramid pooling schemes using both the SLP and MLP. The SLP exceeds 98% mAP for all configurations except for a pyramidal pooling using the levels 1 or 1,2. Compared to the MLP, the SLP depends more on the pyramidal pooling as shown by the QbS results in the first row of Table 12. If no pyramidal pooling is applied, the retrieval performance of the PHOCNet models using an SLP drops considerably by about 10 percentage points from 97.37% to 87.49%. Examining the results obtained on the IAM-DB, a different picture emerges. A general observation is that the performance of a SLP is significantly lower for all pooling configurations compared to models using the MLP. The lack of multiple fully connected layers is particularly evident in the QbE scenario. All performance results between the SLP and MLP differ remarkably by about or even more than 10 percentage points (columns 5 and 6). The same behavior is observed for the query-by-string scenario. However, the differences between SLP and MLP are considerably lower and partially comparable like in the row 4 of Table 12 (91.22% and 92.98%). Due to different writers and the heterogeneous visual appearance of the word images, the PHOCNet models show a higher dependence on the pyramidal pooling of the feature maps. The retrieval performance on the IAM-DB benchmark drops significantly if no pyramid pooling is applied compared to the results obtained on the GW benchmark.

8.4.4 *Comparison to selected results from the literature*

This section compares related approaches discussed in Section 5.2. The results are compared on the GW and IAM-DB benchmark as shown in Table 13. In addition, Table 14 and 15 compare retrieval performances achieved on the Botany and Konzilsprotokolle benchmarks. For all three tables, the highest mean average precisions achieved by the models are indicated in bold, while significantly lower values are written in italics. If approaches from the literature achieve the best results, the values are marked in bold and underlined to emphasize the overall best performance. The best performance results achieved by the own models are indicated only in bold. However, it is important to note that statistical significance can only be determined between two sets of average precision values. As these sets are not available for the related approaches, significance is only determined between the results of the models presented in this thesis. The TPP-PHOCNet model is used for

Table 13: Comparison between selected results presented in the literature on the GW database and IAM-DB for QbE and QbS. Results are shown in mAP [%]. Results marked with an (*) use additional data for pre-training and fine-tuning.

model	loss/sim.	GW		IAM	
		QbE	QbS	QbE	QbS
TPP-PHOCNet	BCE / PRM	**97.81**	**98.71**	**81.52**	**92.98**
TPP-PHOCNet	Cos / Cos	97.70	97.40	81.24	*90.33*
SPP-PHOCNet [SF18]	BCE / Cos	97.58	95.58	85.50	92.38
SPP-PHOCNet [SF18]	Cos / Cos	97.72	97.44	75.85	91.12
TPP-PHOCNet [SF18]	BCE / Cos	97.90	96.73	84.80	92.97
TPP-PHOCNet [SF18]	Cos / Cos	97.96	97.92	82.74	93.42
Triplet-CNN(*) [WB16]	triplet / Cos	93.61	–	70.81	–
PHOC(*) [WB16]	Cos / Cos	98.00	92.23	77.24	89.49
DCToW(*) [WB16]	Cos / Cos	97.98	93.69	76.98	85.33
HWNet_{v1}(*) [KDJ16]	Att-SVM / Euc	94.41	92.84	84.25	91.58
HWNet_{v2}(*) [KDJ18]	m.t. / Cos	98.39	98.94	91.57	96.21
HWNet_{v3}(*) [KDJ23]	m.t. / Cos	**99.48**	**99.80**	**93.22**	**97.53**

all experiments in this section to estimate the attribute representation given the word image. Temporal pyramid pooling is applied on the levels $1, 2, 4$ and 8, and the PHOC representations are deduced from the textual strings using the same pyramid levels. The network models are trained under the evaluation protocol as described in Section 8.3.1 using the BCE loss in combination with the PRM as well as the Cosine loss combined with the Cosine similarity.

Table 13 shows selected performance results from related approaches [SF18; WB16; KDJ23] reported on the GW and IAM-DB benchmarks. Especially the results presented in [SF18] display an interesting comparison to the performance numbers reported in Section 8.4.1. The experimental structure regarding the evaluation protocol and the PHOC representation is identical to the one used in [SF18]. However, a significant difference lies in the Python frameworks used to implement the experiments. Sudholt *et al.* use the Caffe[10] library to carry out the experiments in [SF18]. In contrast, this thesis implements the experiments using the PyTorch[11] framework. Besides this, the SPP-PHOCNet (rows 3 and 4) applies a spatial pyramid pooling using the levels $1, 2$ and 3, and the TPP-PHOCNet (rows 5 and 6) use the levels $1, 2, 3, 4$ and 5. The results obtained on the GW database are almost identical between the TPP-PHOCNet from this thesis, and the version proposed in [SF18]. Although the performance of a model trained using the Cosine loss is slightly lower than the binary cross entropy loss (rows 1 and 2), a significance could not be determined.

10 https://caffe.berkeleyvision.org/
11 https://pytorch.org/ version 1.12

Furthermore, the combination BCE/PRM achieves notably higher results in the QbS scenario on the GW database (row 1) compared to the own model trained with the Cosine loss (row 2) as well as the models from [SF18] (rows 3 to 6). On IAM-DB, the results are equal for the QbS scenario comparing the combination BCE/PRM (row 1) and the numbers reported in [SF18] (rows 3 to 6). However, 90.33% mAP are achieved by the combination Cos/Cos (row 2) shows significantly lower performance than the combination BCE/PRM with 92.98% (row 1). For the QbE scenario, both models (rows 1 and 2) achieve inferior performance compared to the SPP- and TPP-PHOCNet trained with the BCE loss (rows 3 to 5). Although the permutation test cannot be applied, performance differences presented in Tables 12 and 11 indicate that the mAP values of 85.50% and 84.80% (rows 3 and 5 in Table 13) are likely statistically significant compared to the 81.25% (row 1 in Table 13). The performance of the TPP-PHOCNet can be improved by simply applying two more pooling steps as shown in Table 11 (row 4). Two more pooling steps lead to mAP values of 87.64% and 95.66% in the QbE and QbS retrieval scenarios, respectively. These differences are significantly superior. The lower part of Table 13 displays numbers achieved by related approaches reported in [WB16; KDJ16; KDJ18; KDJ23]. Wilkinson *et al.* [WB16] present a comparison between the PHOC and the self-created DCToW attribute representation. The authors additionally demonstrate the retrieval capabilities in a QbE scenario utilizing the Triplet loss approach. All three models from [WB16] (rows 7 to 9) make use of the original ResNet_{34} architecture and apply a global pooling after the last convolutional layer. The resulting feature vector is mapped to the corresponding attribute representation through a multi-layered Perceptron. It is important to note that the models are pre-trained on the CVL database [Kle+13] and subsequently fine-tuned on the respective target dataset. Comparing the numbers reported in [WB16] (rows 7 to 9) and the results achieved by the own TPP-PHOCNet models (rows 1 and 2) reveal a distinct higher performance. All results reported in [WB16] show significantly lower performance than the PHOCNet architectures depicted in rows 1 to 6 of Table 13. Only for the QbE scenario on the GW database, both the PHOC and DCToW approaches achieve comparable mAP values (rows 8 and 9). According to the results presented in [WB16], it can be assumed that global pooling is detrimental to performance despite the ResNet_{34} architecture being used. The results proposed by [KDJ16; KDJ18; KDJ23] shown in the rows 10 to 12 follow an approach called deep feature embedding (cf. Section 5.2). In the first step, convolutional neural networks are trained to classify word classes using the cross-entropy loss. Subsequently, the output values of the penultimate fully connected layer are utilized as latent word image representations. Similar to the approach proposed in [Alm+14b], individual PHOC attributes are predicted using the AttributeSVM in the HWNet_{v1} approach [KDJ16]. These predictions are subsequently projected into a common subspace where the Euclidean distance is applied to rank the retrieval list. For this reason, the loss function and similarity measure as used in HWNet_{v1} are denoted as *Att-SVM/Euc* in the second column labeled *loss/sim* in Table 13. The ensuing versions HWNet_{v2} and HWNet_{v3} apply a *multi-task* loss to obtain the model parameters and the Cosine similarity

to rank the retrieval list. Thus, both approaches are listed as *m.t./Cos* in Table 13. All three approaches build on each other and steadily improve performance. Comparing the results between the multi-staged deep embedding approaches (rows 10 to 12) and the end-to-end trained TPP-PHOCNet (rows 1 and 2), only the HWNet_{v1} shows lower retrieval performance. The improved versions achieve higher mean average precisions, particularly on the IAM-DB for QbE and QbS retrieval (rows 11 and 12). Although a permutation test cannot be applied, the results reported on the IAM-DB are likely significantly higher than the performance displayed for the TPP-PHOCNet (rows 1 and 2), considering the significances shown in Table 12. However, one crucial ingredient in these deep embedding approaches is the heavy usage of additional training data for pre-training and fine-tuning the network models.

Table 14: Comparison between selected results presented in the literature on the Botany database for QbE and QbS. Results are shown in mAP [%]. Results marked with an (*) use additional data for pre-training and fine-tuning.

model	loss / sim	Train I		Train II		Train III	
		QbE	QbS	QbE	QbS	QbE	QbS
TPP-PHOCNet	BCE / PRM	25.58	**44.40**	*78.93*	**93.11**	96.70	**99.09**
TPP-PHOCNet	BCE / Cos	**30.43**	42.28	**85.62**	91.69	**97.47**	98.73
TPP-PHOCNet	Cos / Cos	*13.21*	*21.83*	*76.90*	*85.50*	*86.81*	*92.45*
SPP-PHOCNet [SF18]	BCE / Cos	44.56	26.62	78.93	77.22	94.10	95.43
SPP-PHOCNet [SF18]	Cos / Cos	**45.82**	42.95	71.32	81.21	79.90	89.60
TPP-PHOCNet [SF18]	BCE / Cos	39.86	36.47	76.55	84.23	92.89	96.61
TPP-PHOCNet [SF18]	Cos / Cos	38.80	38.19	73.35	84.42	78.39	87.78
HWNet_{v2}(*) [KDJ18]	m.t. / Cos	–	–	–	–	95.54	93.72
HWNet_{v3}(*) [KDJ23]	m.t. / Cos	–	–	–	–	97.13	97.77

Table 14 and Table 15 show the retrieval performance achieved on the Botany and Konzilsprotokolle benchmarks, respectively. Both tables show the results achieved on three differently sized training data partitions defined by the keyword spotting competition [Pra+16] described in Section 8.1.3 and 8.3.1. As shown in Table 14, the TPP-PHOCNet trained with the BCE achieves highest results on Botany for all three training partitions (rows 1 and 2), except in the QbE scenario for partition Train I. Here, the SPP-PHOCNet using the combination Cos/Cos (row 5) shows distinct higher performance. Conspicuously inferior results are achieved by the TPP-PHOCNet trained using the Cosine loss as shown in row 3 in Table 14. Using the TPP-PHOCNet with the combination Cos/Cos show significantly lower results for all retrieval scenarios on all training partitions (rows 1 and 2 vs. 3). However, it cannot be completely ruled out that the poor performance may be due to the hyperparameters, such as the selected Adam optimizer or the chosen learning rate. When the results achieved by the PRM and Cosine similarity are compared, shown

Table 15: Comparison between selected results presented in the literature on the Konzilsprotokolle database for QbE and QbS. Results are shown in mAP [%]. Results marked with an (*) use additional data for pre-training and fine-tuning.

model	loss / sim	Train I		Train II		Train III	
		QbE	QbS	QbE	QbS	QbE	QbS
TPP-PHOCNet	BCE / PRM	84.04	**84.60**	**97.37**	**98.17**	**98.22**	**98.35**
TPP-PHOCNet	BCE / Cos	**84.17**	82.80	97.33	98.00	97.87	98.29
TPP-PHOCNet	Cos / Cos	*71.2*	*74.99*	95.95	95.97	97.68	97.50
SPP-PHOCNet [SF18]	BCE / Cos	84.34	76.45	96.05	95.27	97.08	96.22
SPP-PHOCNet [SF18]	Cos / Cos	88.31	83.62	95.51	93.78	95.54	93.49
TPP-PHOCNet [SF18]	BCE / Cos	86.01	78.23	97.05	96.98	98.11	98.02
TPP-PHOCNet [SF18]	Cos / Cos	**90.97**	**87.37**	96.45	94.80	96.42	94.63
$HWNet_{v2}$(*) [KDJ18]	m.t. / Cos	–	–	–	–	93.16	71.05
$HWNet_{v3}$(*) [KDJ23]	m.t. / Cos	–	–	–	–	96.24	93.64

in rows 1 and 2 of Table 14, the PRM performs better in the QbS scenario, while the Cosine similarity achieves higher results in the QbE retrieval. The $HWNet_{v3}$ [KDJ23] shows comparable results, although for QbS, the difference between 97.77% and 99.09% achieved by the TPP-PHOCNet are noteworthy. In this comparison, the $HWNet_{v2}$ falls slightly behind. In general, the dominance of the HWNet approach (rows 8 and 9) on the Botany dataset in Table 14 is no longer given. The results achieved on the Konzilsprotokolle benchmark, shown in Table 15, draw a similar picture. The TPP-PHOCNet, trained with the BCE loss (rows 1 and 2), achieves the highest results. In contrast to Botany, the PRM measure (row 1) consistently outperform the Cosine similarity on the larger training partitions Train II and III. However, on partition Train I, best results are reported in [SF18] using the TPP-PHOCNet trained with the Cosine loss (row 7). Similar to Botany, the considerable performance gap between the own TPP-PHOCNet models and the models from [SF18] on partition Train I disappears with an increasing number of training samples in the partitions Train II and III. Interestingly, on the partitions Train II and III, the performance gaps between the BCE and Cosine loss are lower. These results affirm the assumption about the suboptimal selection of hyperparameters. Performance-wise, both $HWNet_{v2}$ and $HWNet_{v3}$ fall behind compared to all PHOCNet models. Especially the 71.05% reported for the $HWNet_{v2}$ in the QbS retrieval on partition Train I, are most likely significantly lower compared to all PHOCNet results.

8.5 RESULTS FOR CUNEIFORM SIGN SPOTTING

This section presents the performance results obtained for cuneiform sign spotting on two benchmarks containing Hittite and Old Assyrian scripts. The ensuing Section 8.5.1 presents the retrieval performance achieved for different pyramid levels using the SPG representation. Subsequently, Section 8.5.2 shows the retrieval performance achieved if cuneiform signs are encoded through the TPG representation. A comparison between four similarity measures using the SPG and TPG representations is drawn in Section 8.5.3. Finally, qualitative results are presented in Section 8.5.4.

8.5.1 *Spatial Pyramid Gottstein Representation*

The retrieval results displayed in Tabels 16 and 17 are obtained from SPP-PHOCNet models, each trained on the respective dataset given a different number of pyramid levels. The architecture used is described in Section 8.3.2, and the models are trained following the training setup from Section 8.3.3. The mAP values are determined according to the evaluation protocol for the Hittite and Old Assyrian benchmarks as explained in Section 8.3.1. The network models are trained using the BCE loss, and similarities are obtained by applying the PRM. In Tabels 16 and 17, the first columns labeled *split* denote the respective splits used by the SPG representation to represent the cuneiform sign classes. Here, gr indicates the original Gottstein representation corresponding to the first level in a pyramidal partitioning. The letters h and v denote a splitting in the horizontal and vertical direction, respectively, while the subscripted indices define the number of splits. For example, a horizontal splitting on the pyramid level 2 is denoted by h_2. The pyramidal partitioning indicated by gr, h_i, and v_i corresponds to the visualization in Figure 27 displayed in Section 7.3.2. The column *uni/all* in Tabels 16 and 17 indicate the number of sign classes uniquely represented by the SPG representation in the respective benchmark. For example, the first pyramid level (i.e., gr) can differentiate 177 out of 301 sign classes in the Hittite and 125 out of 154 classes in the Old Assyrian benchmark. As the number of splits increases, the amount of uniquely represented signs grows until each cuneiform sign is encoded by a single attribute vector that only represents the respective sign. The third column labeled *red/full* denotes the reduced and full dimensionalities of the attribute representations used in the respective experiments. A *full* vector is obtained from keeping all attributes of the SPG representation. In contrast, for the *reduced* vector, all attributes from the SPG representation with a frequency of 0 are discarded. For example, in the attribute representation denoted gr, h_2, v_2 in Table 16, only 116 out of 160 attributes occur at least once. Higher pyramid levels are more sparse, as indicated in the column *red/full* of both Table 16 and 17. Thus, the resulting *reduced* representation is proportionally smaller than its *full* counterpart.

Table 16 shows the results achieved on the Hittite benchmark. A general observation is that the mean average precision increases with the number of uniquely represented sign classes in both retrieval scenarios. The retrieval performance achieved for QbE is

Table 16: Comparison of QbE and QbX retrieval performance obtained for different splits on the Hittite benchmark using the SPG representations. The TPP-PHOCNet is trained with the binary cross-entropy loss and retrieval is performed by applying the PRM similarity measure. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

split	uni/all	red/full	Hittite	
			QbE	QbX
gr	177/301	32/32	*62.50*	*42.85*
gr, h_2	263/301	76/96	*76.34*	*61.21*
gr, h_2, v_2	288/301	116/160	*81.59*	66.80
gr, h_2, v_2, h_3	297/301	169/256	*81.87*	69.54
gr, h_2, v_2, h_3, v_3	301/301	220/352	**83.90**	**69.90**

significantly higher using all available pyramid splits (gr, h_2, v_2, h_3, v_3) compared to the attribute representations using fewer splits. The same can also be observed in the QbX scenario. However, the representation without the vertical splitting on the third level (gr, h_2, v_2, h_3) shows almost the same performance. According to the permutation test, all values in the QbE column are significantly lower than the 83.90%. However, in the QbX column, a difference of 3.1 percentage points between the values 66.80 and 69.90 is not significant due to the low number of 1065 queries.

Table 17 shows the results achieved on the Old Assyrian benchmark. Similar to the observations in Table 16, higher performance values are achieved if the number of cuneiform signs represented by the attributes vector is as low as possible. As the Old Assyrian dataset only contains 154 sign classes, a pyramidal splitting on level 2 is enough to obtain attribute vectors that uniquely represent every sign class. The best results are achieved using all available split annotations. However, comparing the mAP values from Tabels 17 and 16 reveals that the Old Assyrian dataset is visually more challenging, although it contains half the number of cuneiform signs. While the performance between the split combinations gr, h_2 and gr, h_2, v_2 (rows 2 and 3) are almost identical for QbE, in the QbX retrieval, a

Table 17: Comparison of QbE and QbX retrieval performance obtained for different splits on the Old Assyrian benchmark using the SPG representations. The TPP-PHOCNet is trained with the binary cross-entropy loss and retrieval is performed by applying the PRM similarity measure. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

split	uni/all	red/full	Old Assyrian	
			QbE	QbX
gr	125/154	30/30	*40.66*	*32.79*
gr, h_2	152/154	73/90	49.82	44.25
gr, h_2, v_2	154/154	112/150	**50.06**	**47.88**

difference of 3.63 percentage points can be observed. However, due to the low number of 471 queries, the distinct difference between 44.25% and 47.88% is not significant according to the permutation test. Using the original Gottstein representation (i.e., *gr*) leads to significantly lower performance in both retrieval scenarios.

8.5.2 *Temporal Pyramid Gottstein Representation*

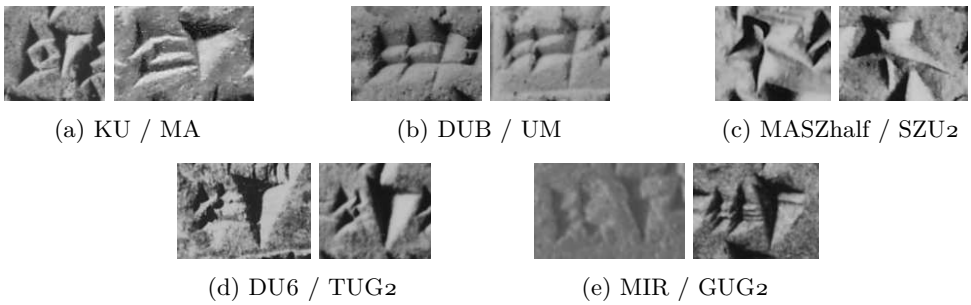
For each row in Tabels 18 and 19, separate SPP-PHOCNet models are trained on four cross-validation splits. These models have the architecture described in Section 8.3.2 and are trained following the training setup from Section 8.3.3. The performance results are obtained following the evaluation protocol described in Section 8.3.1. Similar to Section 8.5.1, the BCE loss is used to train the SPP-PHOCNet models and retrieval lists are ranked by applying the PRM similarity measure. The best performance results are highlighted in bold, while significantly lower values are written in italics. The permutation test determines significant differences between mean average precision values. In Tabels 18 and 19, the *pyramid levels* column denotes the pyramid levels used to obtain the TPG attribute representation. As mentioned in the previous section, the columns *uni/all* and *red/full* show the number of uniquely represented sign classes and the dimensionality of their respective attribute vectors. The far right columns labeled *QbE* and *QbX* show the performance obtained by the SPP-PHOCNet models for different level configurations in the QbE and QbX retrieval scenarios. Both tables are divided into upper and lower parts. The upper part (rows 1 to 5) shows performance results for attribute representations using the four wedge types a, b, c and d. On the other hand, the lower part of Table 18 (rows 6 to 10) shows the mAP values obtained by the SPP-PHOCNet models using the additionally annotated wedge information described in Section 7.3.3.

The results in Table 18 provide a similar picture to the previous section's tables. A general observation is that an increasing number of pyramid levels (column 1) leads to higher-dimensional attribute vectors (column 3), which, in turn, leads to more uniquely represented cuneiform signs (column 2) and, finally, to higher performance values (columns 4 and 5). In the upper part of Table 18, 191 out of 301 cuneiform signs are uniquely represented by the vectors containing 28 attributes on the first pyramid level. The results from this model are noticeably lower compared to the top performance of 82.69% and 69.32% (rows 9 and 10). For QbE, the peak performance of $78,47\%$ is reached using three pyramid levels (i.e., $1, 2, 3$), while higher pyramid levels do not increase the mAP. This is most likely due to the number of 267 uniquely represented cuneiform signs out of 301 sign classes, which is reached using three pyramid levels. However, in the QbS scenario, the model performance benefits from a more fine-grained pyramidal splitting. Although the number of uniquely represented sign classes remains unchanged (267/301), performance increases from 61.26% if three levels are used to 64.02% if the annotations are represented through five pyramid levels. The difference between 64.02% and 61.26% is statistically not significant due to the low number of QbX queries. The lower part of Table 18 shows the results achieved if the additional annotations indicating specific wedge constellations

Table 18: Comparison of QbE and QbX retrieval performance obtained for different pyramid levels on the Hittite benchmark using the TPG representations. The TPP-PHOCNet is trained with the binary cross-entropy loss and retrieval is performed by applying the PRM similarity measure. The additional attributes denotes the wedge annotations x, v, h and g (cf. Section 7.3.3). Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

pyramid levels	uni/all	red/full	Hittite	
			QbE	QbX
without additional attributes				
1	191/301	28/28	*64.59*	*45.13*
1, 2	248/301	82/84	*76.54*	*57.89*
1, 2, 3	267/301	138/168	*78.47*	*61.26*
1, 2, 3, 4	267/301	213/280	*78.70*	63.74
1, 2, 3, 4, 5	267/301	288/420	*78.61*	64.02
with additional attributes				
1	223/301	32/32	*71.01*	*51.91*
1, 2	279/301	94/96	*81.20*	64.47
1, 2, 3	296/301	161/192	*82.10*	68.82
1, 2, 3, 4	296/301	252/320	**82.69**	68.69
1, 2, 3, 4, 5	296/301	334/480	82.57	**69.32**

are used. Using these four additional attributes presents more cuneiform signs uniquely compared to the upper part of the table (column 2). Consequently, level 1 in row 6 of Table 18 performs significantly higher than level 1 in row 1 for both retrieval scenarios. Similar to the upper part of Table 19, peak performance in the QbE scenario is achieved using three or more pyramid levels. For QbX, the difference between 68.82% in row 8 and 69.32% in row 10 is not distinct as the difference in the upper part of Table 19 between the same pyramid levels (rows 3 and 5). Due to the large number of queries in the QbE retrieval, all results except in the last row are significantly lower. In contrast, the low number of queries in the QbE retrieval does not lead to significant differences for most of the results compared to 69.32%. What surprises is that even for a difference of 5.58 percentage points



(a) KU / MA (b) DUB / UM (c) MASZhalf / SZU2

(d) DU6 / TUG2 (e) MIR / GUG2

Figure 40: Five examples from the Hittite database showing pairs of visually similar cuneiform signs from different classes.

(69.32% – 63.74%) significance cannot be determined, although intuitively this number appears to be significant. Furthermore, despite the additional attributes, the maximum number of uniquely represented cuneiform signs is 296 because 5 pairs of sign classes are represented by the same attribute vectors each. These pairs are visualized in Figure 40. Although annotated as different sign classes, the wedge constellations are the same, thus, not distinguishable by the SPG representation.

Table 19 present the results achieved on the Old Assyrian dataset utilizing the sequence-based annotations to represent the cuneiform signs. The upper part of the table (rows 1 to 5) shows the performance values achieved without using additional information regarding the wedge constellations, while the lower part (rows 6 to 10) depicts the results if this information is included. Similar to Table 18, it requires at least the third pyramid level to represent the maximum number of cuneiform signs in the Old Assyrian collection. Adding the fourth and fifth pyramid levels does not increase the number of unique attribute vectors. The Old Assyrian database contains 25 attributes (column 3, row 1) representing the four wedge types and 29 attributes if additional annotations are included (column 3, row 6). As the Old Assyrian dataset contains fewer sign classes than the Hittite database, the first pyramid level represents 123 out of 154 signs as a unique vector. This representation already covers 79.87% of all classes, which leads to a smaller range between the lowest and highest mAP values shown in Table 19. Using the SPG representation with the additional

Table 19: Comparison of QbE and QbX retrieval performance obtained for different pyramid levels on the Old Assyrian benchmark using the TPG representations. The TPP-PHOCNet is trained with the binary cross-entropy loss and retrieval is performed by applying the PRM similarity measure. The additional attributes denotes the wedge annotations x, v, h and g (cf. Section 7.3.3). Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

pyramid levels	uni/all	red/full	Old Assyrian	
			QbE	QbX
without additional attributes				
1	123/154	25/25	*42.30*	*35.37*
1, 2	148/154	73/75	*50.57*	47.33
1, 2, 3	152/154	123/150	*51.31*	46.05
1, 2, 3, 4	152/154	192/250	54.71	**51.89**
1, 2, 3, 4, 5	152/154	240/375	52.99	51.23
with additional attributes				
1	138/154	29/29	*43.74*	*37.23*
1, 2	152/154	84/87	53.18	46.24
1, 2, 3	154/154	144/174	*51.85*	47.28
1, 2, 3, 4	154/154	227/290	51.97	50.27
1, 2, 3, 4, 5	154/154	281/435	**54.87**	50.92

annotations and the pyramid levels $1, 2, 3, 4$ and 5, all 154 sign classes are represented by individual vectors. A general observation is that the additional annotations do not boost the performance on the Old Assyrian benchmark, comparing the rows 3 to 5 and 8 to 10 in Table 19. Contrarily, the performance for QbX in rows 4 and 5 are, although not significantly, higher than those in rows 9 and 10. Similar to QbX, the performance numbers for the QbE retrieval are almost identical, comparing the rows 4 to 5 and 9 to 10. All representations representing at least 152 cuneiform sign uniquely achieve mAP values above 51%. However, there is a considerable range between the lowest 51.31% and the highest 54.87%, where the permutation test determines significance.

8.5.3 *Comparison of Similarity Measures*

The values presented in Tabels 20 and 21 show the results obtained for the SPG and TPG representations on the Old Assyrian and Hittite benchmarks. The architecture of the SPP-PHOCNet models used in the following experiments is described in Section 8.3.2. The performance values are obtained following the evaluation protocol from Section 8.3.1 and the training setup from Section 8.3.3. Both tables have columns labeled SPG or TPG denoting the representation used in the respective experiment. The SPG representation uses all five available splits (gr, h_2, v_2, h_3, v_3) to represent the cuneiform sign classes, while the TPG representation uses the pyramid levels $1, 2, 3, 4$ and 5, and the additional annotations.

Table 20 shows the results of different similarity measures. Given the annotation type (ex. SPG) and the benchmark (ex. Hittite), mAP values are obtained from four SPP-PHOCNet models trained according to the four cross-validation partitions (cf. Section 8.3.1). The numbers in Table 20 show the averaged mAP values. A similarity measure, denoted by the column labeled *similarity*, is applied to the attribute estimates of the four network models obtained from the cross-validation training on one benchmark using one of the two attribute representations (SPG or TPG). Notably, in the QbE retrieval, the Cosine similarity consistently achieves the best results for benchmarks and attribute representations. However, for QbX, all similarity measures are outperformed by the PRM. This result pat-

Table 20: Comparison of QbE and QbX retrieval performance obtained for the Hittite and Old Assyrian benchmarks using the grid-based and sequence-based representations. The TPP-PHOCNet is trained with the binary cross entropy loss. Results are shown in mAP [%]. The best performance values are marked as bold and significantly different results are highlighted in italics.

similarity	SPG				TPG			
	Hittite		Old Assyrian		Hittite		Old Assyrian	
	QbE	QbX	QbE	QbX	QbE	QbX	QbE	QbX
PRM	*83.90*	**69.90**	50.06	**47.88**	*82.57*	**69.32**	*54.87*	**50.92**
Cosine	**85.55**	68.79	**51.88**	46.57	**83.68**	65.01	**57.90**	48.72
Euclidean	85.43	68.21	51.78	46.45	83.24	64.86	56.51	48.11
Cityblock	*84.68*	*62.73*	51.59	44.67	*82.26*	*59.40*	56.06	45.62

tern is similar to the numbers presented for word spotting in Table 7. Additionally, the PRM similarity measure achieves noticeably lower results in the QbE retrieval on both benchmarks and both attribute representations, except in column 6 (82.57%). The numbers in Table 20 support the claim made in Section 8.4 that the performance improves with the PRM similarity measure when queries are provided as binary-valued vectors.

Table 21 shows the results of SPP-PHOCNet models trained using corresponding loss functions and similarity measures. The first two columns labeled *loss* and *similarity* denote the combination of the loss function and its respective similarity measure used. Similar to Section 8.4.1, the mean squared error (MSE) as well as the mean absolute error (MAE) are used to train these models. Despite the combinations, the hyperparameters defined in Section 8.3.3 are applied for training. All experiments in Table 21 use the identical attribute encoding as in Table 20. Similar to Table 2, the PRM achieves the highest results in the QbX retrieval on all benchmarks and attribute representations compared to the other similarity measures. In the case of QbE, no single dominant similarity measure exists. While the PRM is superior in both retrieval scenarios using the TPG representation, the Euclidean distance performs significantly better on the Old Assyrian benchmark. On the Hittite benchmark, best results are achieved by the Cosine similarity utilizing the SPG representation.

Table 21: Comparison of QbE and QbX retrieval performance obtained for the Hittite and Old Assyrian benchmarks using the grid-based and sequence-based representations. The TPP-PHOCNet is trained with corresponding loss functions and similarity measure as denoted in column 1 and 2. Results are shown in mAP [%]. In each column, the highest mAP values are marked as bold and significantly lower results are highlighted in italics.

loss	similarity	SPG				TPG			
		Hittite		Old Assyrian		Hittite		Old Assyrian	
		QbE	QbX	QbE	QbX	QbE	QbX	QbE	QbX
BCE	PRM	83.90	**69.90**	*48.93*	**46.60**	**82.57**	**69.32**	**54.87**	**50.92**
Cosine	Cosine	**83.97**	*63.84*	*48.39*	*38.93*	*80.61*	*57.17*	*47.41*	*42.27*
MSE	Euclidean	83.38	66.56	**53.44**	44.80	*81.42*	*61.42*	53.37	44.80
MAE	Cityblock	*76.86*	*40.35*	*38.00*	*28.07*	*74.11*	*38.46*	*46.75*	*30.91*

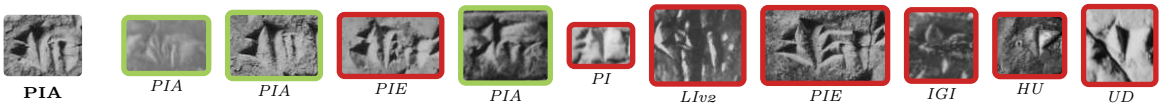
8.5.4 *Qualitative Results and Attribute Statistics*

Similar to Section 8.4.2, quantitative retrieval results and the quality of attribute estimates are presented in this section. Figure 41 to 48 visualize on page 125 to 128 results for the QbE and QbX retrieval scenarios achieved on the Old Assyrian and Hittite benchmarks using the SPG and the TPG representations. The results are obtained by the SPP-PHOCNet models shown in Table 20, each trained with the BCE loss. The retrieval lists are ranked according to the similarity measure of the PRM. In Figure 41 to Figure 48, the top 10 retrieved candidates are depicted, with relevant signs marked by green boxes

and irrelevant ones by red boxes. Queries are selected according to their average precision values in descending order. Similar to the visual results presented for word spotting, the average precision is influenced by the number of relevant candidates and their ranking. For example, the average precision values obtained for the queries $KAxU$ and LUM in Figures 44a and 44a show a 22.29 percentage points difference. This considerable amount suggests significantly different results. However, a visual analysis of the retrieval list reveals a different impression to a human user. In both retrieval lists, the top 3 and top 5 candidates are correctly retrieved, which suggests that the list for the query LUM is of high quality. Regarding this query, one missing candidate is responsible for the 22.29 percentage points gap. Similar differences can be observed for other examples in Figures 41 to 48 as well.

The bar charts on page 129 and 130 show the attribute statistics for the Hittite (Figure 49) and the Old Assyrian (Figure 50) benchmarks. In analogy to Section 8.4.2, the absolute difference $|a - \hat{a}|$ between a present attribute a and its respective estimate $\hat{a}$ is depicted by blue bars. The maximum normalized attribute frequency $\|fr(a)\|_{max}$ is indicated by orange bars. Both representations exhibit similar predictive performance for the different types of wedges (Figure 49a and 49c). However, this performance decreases as the number of wedges increases. In particular, there is a significant disparity in the predictive performance for wedge type a, while this difference is much smaller for the other types. Notably, in Figure 49a, poor predictions are observed for attributes $a8$ and $a10$, while $a9$ is predicted more accurately. These differences can be attributed to the frequency of occurrences and the distribution across sign classes. On the other hand, the highest wedge counts for the other types are estimated very accurately. For instance, the wedge types $c11$ and $c12$ only occur in one category, which may have a distinct visual representation. This unique appearance may make estimating these attributes easier for the neural network. In Figure 49c, the predictions for $a10$ have improved significantly, but the estimation quality for $a5$ and $a6$ have worsened. Furthermore, the difference for $b4$, $b5$, and $b6$ increased as well. A comparison between the pyramid splits in Figure 49b and 49b shows that the prediction quality is almost equal across the splits. Although the representations are not directly comparable, what stands out is the frequency of attributes. In the SPG representation (Figure 49b), attributes are distributed more uniformly across the splits, while in the TPG (Figure 49d) the number of occurring attributes decreases with increasing pyramid levels. The statistical data shows a similar pattern for the Old Assyrian benchmark. Among the different wedge types, attributes representing constellations with fewer wedges are predicted more accurately than those with more wedges. In general, the accuracy of prediction decreases as the number of wedges increases, as seen in both the SPG and the TPG representation (Figure 50b and Figure 50d). However, the attributes $b7, b8, c8$ and $c9$ in Figure 50a, as well as $b7$ and $b8$ in Figure 50c, are exceptions and are predicted accurately. The bar charts for the pyramid splits show that the attributes are predicted with similar accuracy across the various splits.

(a) query: *ALAM*, occurrences: 6, AP: 100.00%,



(b) query: *PIA*, occurrences: 4, AP: 71.38%

(c) query: *ASZ3*, occurrences: 6, AP: 39.26%

(d) query: *NIR*, occurrences: 3, AP: 7.61%

Figure 41: Visualization of QbE retrieval results on the Hittite benchmark using the SPG representation.

(a) query: *ALAM*, occurrences: 6, AP: 100.00%,

(b) query: *GIR4*, occurrences: 4, AP: 61.46%

(c) query: *KIN*, occurrences: 5, AP: 24.88%

(d) query: *SZAB*, occurrences: 3, AP: 14.65%

Figure 42: Visualization of QbX retrieval results on the Hittite benchmark using the SPG representation.

(a) query: *BURU14*, occurrences: 3, AP: 100.00%,

(b) query: *BUR*, occurrences: 5, AP: 74.44%

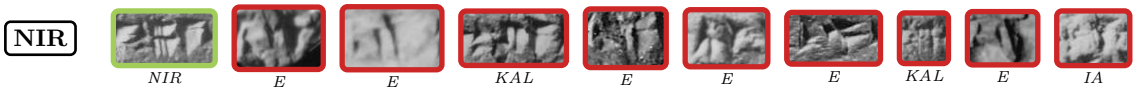
(c) query: *UR2*, occurrences: 4, AP: 38.09%

(d) query: *SU*, occurrences: 7, AP: 8.77%

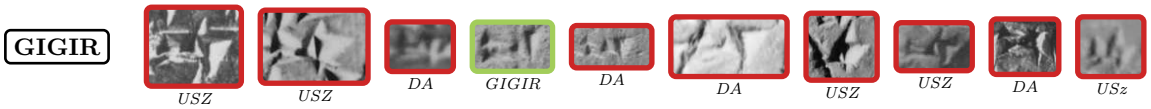
Figure 43: Visualization of QbE retrieval results on the Hittite benchmark using the TPG representation.

(a) query: *KAxU*, occurrences: 3, AP: 100.00%

(b) query: *LUM*, occurrences: 6, AP: 77.71%



(c) query: *NIR*, occurrences: 3, AP: 25.26%

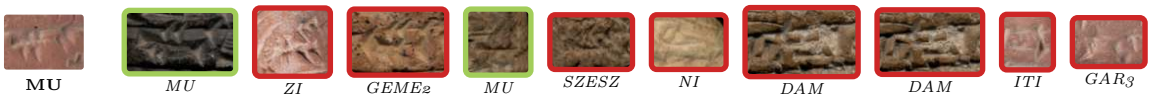


(d) query: *GIGIR*, occurrences: 3, AP: 6.33%

Figure 44: Visualization of QbX retrieval results on the Hittite benchmark using the TPG representation.

(a) query: *DUB*, occurrences: 2, AP: 100.00%,

(b) query: *TUM*, occurrences: 3, AP: 83.33%



(c) query: *MU*, occurrences: 4, AP: 39.66%

(d) query: *SZIR*, occurrences: 5, AP: 7.10%

Figure 45: Visualization of QbE retrieval results on the Old Assyrian benchmark using the SPG representation.

(a) query: *HA*, occurrences: 6, AP: 100.00%,

(b) query: *SUR*, occurrences: 6, AP: 68.04%

(c) query: *IA2*, occurrences: 4, AP: 23.31%

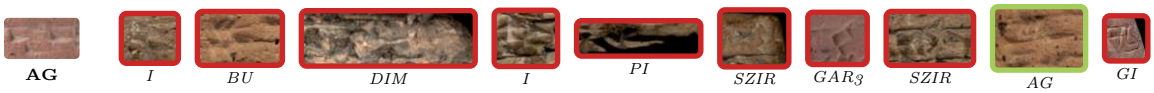
(d) query: *MIN*, occurrences: 3, AP: 5.95%

Figure 46: Visualization of QbX retrieval results on the Old Assyrian benchmark using the SPG representation.

(a) query: *E*, occurrences: 4, AP: 100.00%,

(b) query: *HA*, occurrences: 7, AP: 73.45%

(c) query: *MU*, occurrences: 6, AP: 38.97%



(d) query: *AG*, occurrences: 3, AP: 5.03%

Figure 47: Visualization of QbE retrieval results on the Old Assyrian benchmark using the TPG representation.

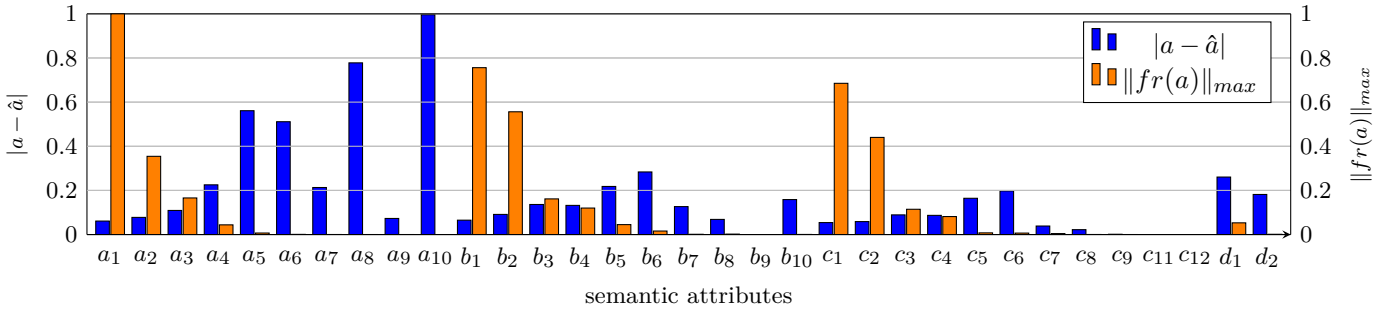
(a) query: *E*, occurrences: 3, AP: 100.00%,

(b) query: *IB*, occurrences: 6, AP: 64.91%

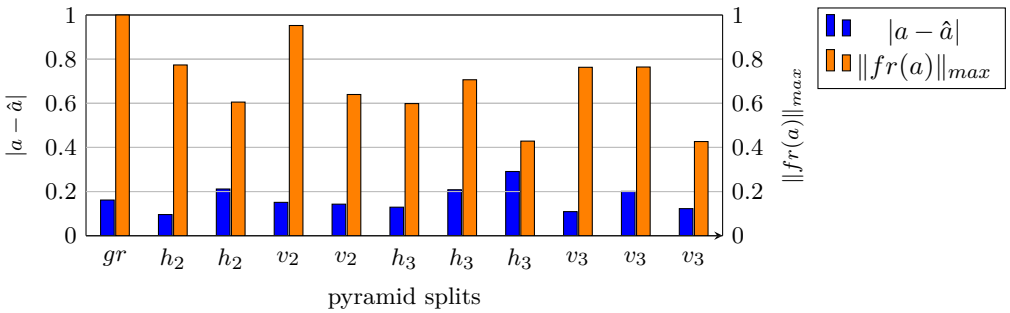
(c) query: *U*, occurrences: 5, AP: 18.46%

(d) query: *TI*, occurrences: 2, AP: 1.79%

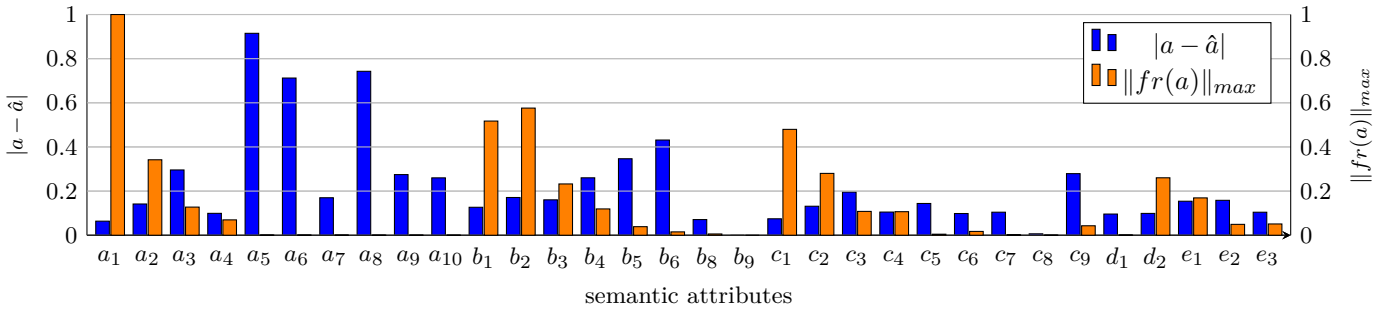
Figure 48: Visualization of QbX retrieval results on the Old Assyrian benchmark using the TPG representation.



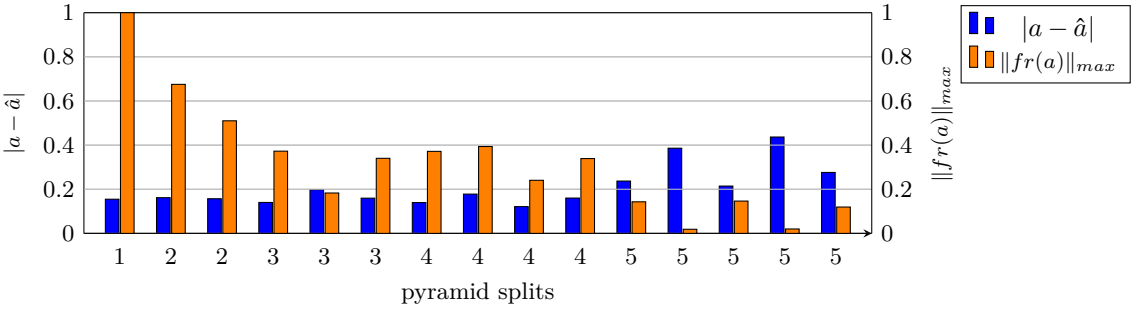
(a) Hittite SPG.



(b) Hittite SPG.

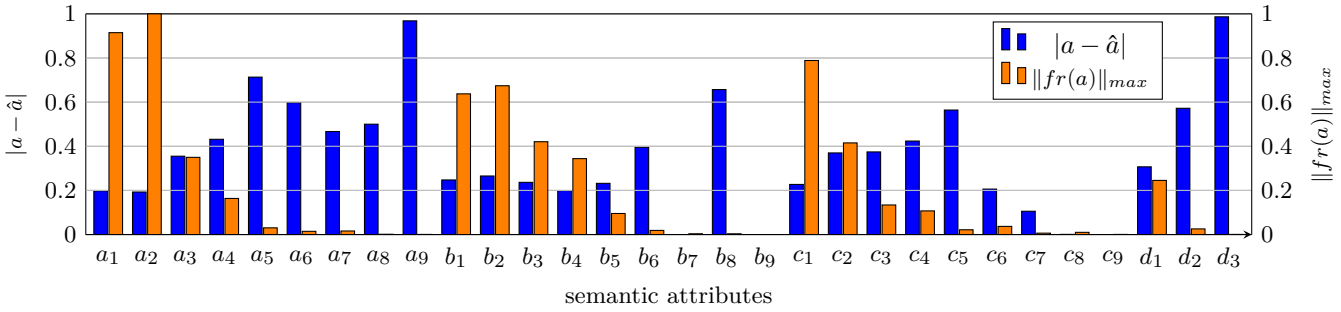


(c) Hittite TPG.

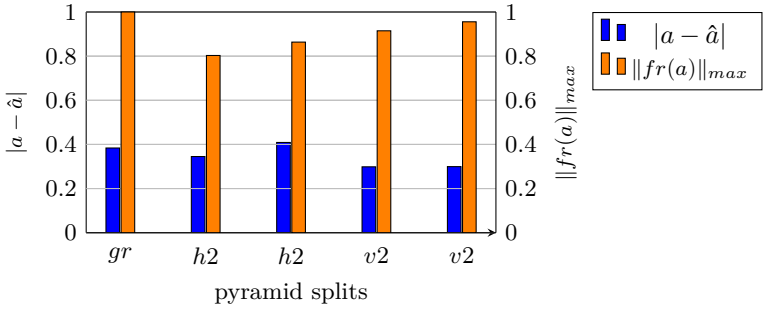


(d) Hittite TPG.

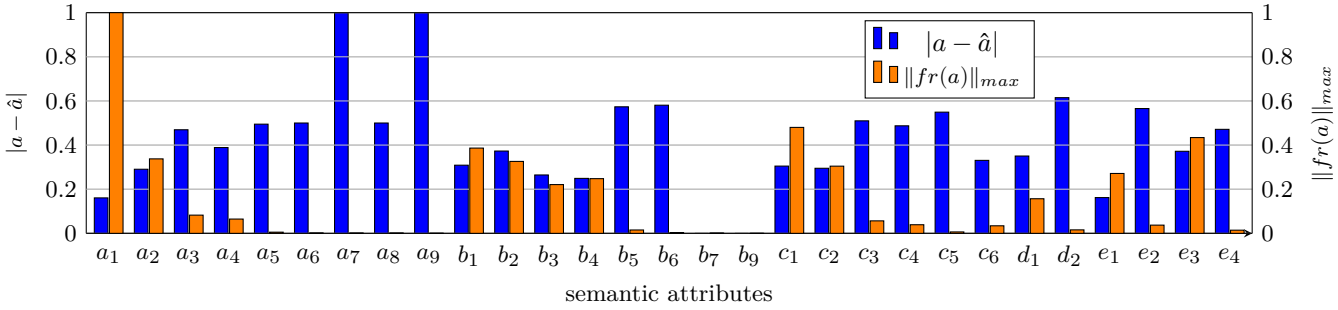
Figure 49: Bar chart shows the mean absolute error $|a-\hat{a}|$ between label a and estimate $\hat{a}$ (blue bars) as well as the maximum normalized attribute frequency $\|fr(a)\|_{max}$ (orange bars).



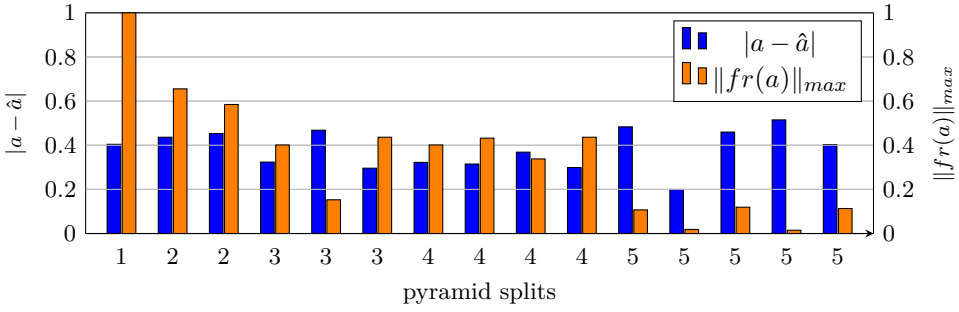
(a) Old Assyrian SPG.



(b) Old Assyrian SPG.



(c) Old Assyrian TPG.



(d) Old Assyrian TPG.

Figure 50: Bar chart shows the mean absolute error $|a-\hat{a}|$ between label a and estimate $\hat{a}$ (blue bars) as well as the maximum normalized attribute frequency $\|fr(a)\|_{max}$ (orange bars).

9 CONCLUSION

This chapter concludes the content of this thesis. Therefore, a short summary of the presented methodology is given in the ensuing Section 9.1. In Section 9.2, findings from the experimental evaluations and the insights derived from them are recapped. The final Section 9.3 reflects thoughts and considerations regarding possible ensuing research and methodological enhancements.

9.1 SUMMARY

This thesis proposes a methodology for *attribute-based word spotting* in handwritten documents and adopts this approach to the ancient writing system named *cuneiform*. The core contribution for attribute-based word spotting is the similarity measure referred to as *probabilistic retrieval model* (PRM). This measure utilize the binary nature of a word embedding technique named *pyramidal histogram of characters* (PHOC) [Alm+14b]. These numerical representations are obtained by decomposing textual strings into their individual semantics following a pyramidal scheme. This embedding is based on the concept of *semantic attributes* [LNH14]. In the context of word spotting, semantics are defined as either individual or a sequence of two or more letters, typically referred to as uni-, bi- or tri-grams. The resulting PHOC vectors are binary valued attribute representations, indicating the presence or absence of encoded semantics (ex. letters) at a certain position within the textual strings. The core idea of the probabilistic retrieval model is the assumption that these binary valued attributes are Bernoulli distributed. Consequently, the probabilities between individual attributes are evaluated given two PHOC representations. The resulting similarity is assigned as the sum of all attribute probabilities.

In order to assess similarities between a query and all word image candidates from a database, the PHOC representations have to be estimated first. Therefore, a convolutional neural network is used to predict the individual attributes of a PHOC vector. The architecture of this network is based on the popular PHOCNet [SF16]. In order to estimate the attributes of a PHOC representation, an optimal set of parameters is required by the neural network. These parameters are obtained by applying an iterative and gradient-based training. A further contribution in this thesis is the derivation of a *suitable loss function* for the training of a neural network. Applying the statistical framework named *generalized linear models* (GLMs), the *binary cross-entropy* (BCE) is derived as the appropriate loss function that corresponds to the probabilistic retrieval model. Using the binary cross-entropy loss to find the optimal set of parameters for a convolutional neural network and subsequently apply the probabilistic retrieval model to assess similarities results in the

superior benefit that the minimization of the BCE loss corresponds to the maximization of the PRM similarity between two equal PHOC representations.

The third major contribution is the *transfer* of the attribute-based approach from the domain of handwritten documents to an ancient writing system named *cuneiform*. In the context of this system, Cuneiform is written by pressing a wedge stylus into moist clay tablets. Individual signs are composed of different wedge impressions and their constellations. The approach in this thesis utilizes the concept of semantic attributes to define these impressions as individual semantics. Therefore, two alphanumeric encodings are presented. The first encoding divides cuneiform signs following a grid-like pattern. The signs are segmented according to a pyramidal scheme with an increasing number of grid cells. The resulting attribute representation is called *spatial pyramid Gottstein* (SPG). For the second encoding, cuneiform signs are annotated in sequential order according to their wedge constellations. Subsequently, these encodings are converted into attribute representations using the PHOC algorithm. As the resulting representations only consider wedge positions in the horizontal direction, these representations are referred to as the *temporal pyramid Gottstein* (TPG). Both the SPG and TPG representations enable the spotting system to perform the novel retrieval scenario named *query-by-expression* (QbX).

9.2 FINDINGS AND INSIGHTS

A number of experiments are conducted in this thesis to evaluate the aforementioned methodology empirically. The probabilistic retrieval model is compared to other measurements such as the Cosine similarity, the Euclidean distance and the Cityblock distance. The first comparison is conducted on the two word spotting benchmarks *GW* and *IAM-DB*. The second comparison is realized on two cuneiform benchmarks containing collections of Hittite and Old Assyrian scripts. The results obtained from the benchmarks of the handwritten documents show that a word spotting system benefits from the PRM similarity measure, especially in the QbS retrieval scenario. Particularly on the IAM-DB benchmark, other similarity measures achieve significantly lower results in the QbS retrieval. These results are confirmed on the cuneiform benchmarks where the PRM measure consistently achieves higher performance for QbX retrieval using both SPG and TPG representations.

The evaluation of architecture components reveals that retrieval performance does not necessarily benefit from network models containing more parameters or layers. Exchanging the VGG-inspired convolutional part of the PHOCNet by the ResNet architecture does not increase retrieval performance on the word spotting benchmarks GW and IAM-DB. However, enlarging the depth of the PHOCNet from 13 to 16 convolutional layers increases the performance obtained on the heterogeneous IAM-DB. The sharpest increase in performance is achieved by including more subsampling steps in the convolutional part of the networks. While the results achieved on the GW database remain constant despite the number of poolings applied, the results on IAM-DB benefit significantly. Altering the configuration of the pyramid pooling does not influence the performance, especially in

combination with the MLP. In particular, the results achieved in the QbS scenario on both word spotting benchmarks show throughout the same performance values. The only exception is applying the pyramid pooling using only level 1, as significantly lower results are achieved for both retrieval scenarios. Moreover, higher performance values are obtained if the attribute representations are estimated using the MLP instead of the SLP. However, this behavior is expected as multiple fully connected layers have a stronger representational power than one layer.

Retrieval results obtained by the TPP-PHOCNet using two combinations of loss functions and similarity measures, namely the BCE together with the probabilistic retrieval model and the Cosine loss combined with the Cosine similarity, are compared to the performance values reported in the related literature. Comparable results are achieved on the benchmarks of the GW and IAM-DB. The mean average precision values are equal to the ones presented by Sudholt *et al.* [SF18] using the same PHOCNet architectures. Only the results reported on IAM-DB for the QbE scenario fall significantly behind compared to the performance in [SF18]. However, considering the results achieved by the own TPP-PHOCNet model applying four instead of two pooling steps, superior performance can be reported on the IAM-DB compared to the results from [SF18]. Additional experiments are conducted on the two benchmarks *Botany* and *Konzilsprotokolle*. For the smallest training data partition on the Botany database, the best results achieved in the QbE retrieval scenario are distinct lower than the highest numbers reported in [SF18]. However, for QbS, higher performance can be reported for the combination using the binary cross-entropy loss and the PRM similarity measure. On the larger two partitions, significantly higher results are achieved in both retrieval scenarios. The following pattern can be observed throughout the data partitions: QbE retrieval benefits if the Cosine similarity is applied, whereas in the QbS scenario, higher performance is achieved using the PRM. A similar conclusion can be drawn from the results obtained on the Konzilsprotokolle benchmark. The noticeable difference is that, compared to the Cosine similarity, the PRM shows slightly higher mAP values for both retrieval scenarios, except on the smallest training partition. However, this difference is negligibly small, and comparing the benchmarks, higher performance values are achieved for Konzilsprotokolle, which suggests that the PRM benefits from better attribute estimations.

In addition to word spotting, several experiments are conducted to evaluate the performance on two cuneiform benchmarks containing Hittite and Old Assyrian scripts. These experiments present the results achieved using the *SPG* and *TPG*. A general observation for both representations is that the performance correlates with the number of uniquely represented cuneiform signs. This ratio between unique sign representations and the total number of cuneiform signs in one dataset is influenced by two factors: the number of splits in the pyramidal structure of the representations and the diversity in semantics. The results show that the higher the number of pyramid splits, the higher the retrieval performance for both scenarios. A comparison of the two representations shows comparable results achieved on the Hittite benchmark using higher pyramid levels and slightly

higher performance values on the Old Assyrian benchmark using the TPG representation. However, comparing the lower pyramid levels (i.e., *gr* vs. level 1) reveals that the TPG representation achieves superior results on both benchmarks. Moreover, the additional information regarding the wedge constellation that enriches the sequence-based annotations improves the retrieval results on the Hittite benchmark. However, the results remain unchanged for Old Assyrian, and the performance numbers are almost equal. These findings can be attributed to the fact that this additional information enables the sequence-based annotations to represent more cuneiform signs uniquely in the Hittite database. On the other hand, the Old Assyrian dataset contains only 154 signs in total. Here, the sequence-based annotations without additional information can already cover 152 cuneiform signs. Including this information will represent only two more signs as a unique attribute vector, which is not noticeable in influencing the performance.

9.3 OUTLOOK

Word spotting is an area in the document analysis community that has been researched since the late 1990s. Accordingly, numerous publications have very different approaches for retrieving individual word images. With the advent of *convolutional neural networks*, the performance has shifted significantly upwards and approaches such as in [KDJ23] show that the retrieval performance is already scratching at the upper limit. It is, therefore, questionable whether it makes sense to continue striving to improve performance in the defined benchmarks for segmentation-based word spotting. Instead, it is more promising to go into the area of *annotation-free* word spotting. Various approaches [WF20; WBF20; WF22a; WF22b] show that there is still much room for improvement in performance in this area. Strictly speaking, annotation-free approaches [RWF21] correspond more to a real-world application and sometimes represent a more significant practical benefit for word spotting on historical documents. The probabilistic retrieval model can make an essential contribution to this, especially when a model makes predictions for non-annotated data and uses these in turn for further training, which is known as *pseudo-labeling* and *self-training* [WF22b]. The PHOC, as an *attribute-based representation*, supports the prediction of unseen word images during training and can be used for the task of *zero-shot retrieval*. Furthermore, the PHOCNet, as presented in this thesis, can be transformed into an approach for segmentation-free word spotting. For this purpose, it can be integrated into a *patch-based* [RWF21] framework or used in combination with the *region-proposal networks* [WLB20].

The ancient writing system cuneiform is less known in the document analysis community, and only a few works provide approaches for recognizing or retrieving cuneiform signs (cf. Chap. 7). The main reason is the lack of annotated data, which requires a human expert to provide it. The research project CuKa made an essential step toward the challenge of annotating cuneiform tablets. This project developed a web-based cuneiform analysis platform where photographs of tablets can be uploaded and subsequently anno-

tated, recognized, or retrieved. Furthermore, the *grid-based Gottstein representations* for cuneiform signs were developed during this project as well. Together with the sequence-based representations presented in this thesis, these attribute-based approaches provide a solid foundation for several possible future works in this area. Due to limited data, ancient document collections pose a challenge, requiring few or zero-shot learning approaches. The attribute-based Gottstein representations support these methods capable of handling a small amount of training data (cf. Chap. 3). Moreover, annotation-free approaches can be adapted to cuneiform as presented for word spotting tasks [RWF21; WF22a; WF22b]. The pseudo-labeling and self-training approaches can be directly adapted to another visual domain. On the other hand, synthetically generated handwriting [KJ16] is not usable for generating cuneiform signs. However, with the massive success of generative models like the *generative adversarial networks* [Cre+18] or the recent *diffusion models* [Cro+23], network models demonstrated their potential to generate images of high quality for a specified visual domain. In this regard, two initial approaches have been proposed by the author of this thesis using *cycle-consistent* generative adversarial networks [Zhu+17] in order to transform sign images between two visual domains [Rus+19] and *impaint* individual signs into cuneiform tablets under context aware conditions [BRF21]. Besides the small data, the attribute-based sign representations can represent hundreds or thousands of different sign classes. This capability opens the field of approaches grouped under the name of *transfer learning* (cf. Chap. 3). Knowledge can be shared across different scripts using only one neural network model to predict the attributes of cuneiform signs. More data from other scripts can partially offset the difficulty of small data sets for a particular script. And last but not least, the segmentation-based cuneiform sign spotting approach in this thesis can be enhanced to provide segmentation-free capabilities. Similar to handwritten documents, a possible starting point could be a patch-based approach for annotation-free cuneiform sign spotting as demonstrated in [Rot+15]. An alternative approach is to utilize region-proposal networks [WLB20]. However, this method requires annotated data. Furthermore, in contrast to typical object detection tasks in computer vision [Zou+19], the main challenge in the cuneiform domain is the number of signs written on a tablet. These approaches must be adapted from predicting a few large objects to many smallish cuneiform signs.

BIBLIOGRAPHY

[Agr15] Alan Agresti. *Foundations of Linear and Generalized Linear Models*. John Wiley & Sons, Incorporated, 2015.

[AHP06] T. Ahonen, A. Hadid, and M. Pietikainen. "Face Description with Local Binary Patterns: Application to Face Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.12 (2006), pp. 2037–2041.

[Aka+16] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. "Label-Embedding for Image Classification." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.7 (2016), pp. 1425–1438.

[Aka+15] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. "Evaluation of output embeddings for fine-grained image classification." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 2927–2936.

[Ald+13] David Aldavert, Marçal Rusiñol, Ricardo Toledo, and Josep Lladós. "Integrating Visual and Textual Cues for Query-by-String Word Spotting." In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, pp. 511–515.

[Ald+15] David Aldavert, Marçal Rusiñol, Ricardo Toledo, and Josep Lladós. "A study of Bag-of-Visual-Words representations for handwritten keyword spotting." In: *International Journal on Document Analysis and Recognition (IJDAR)* 18 (2015), pp. 223–234.

[Alm+12] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. "Efficient Exemplar Word Spotting." In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 67.1–67.11.

[Alm+13] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. "Handwritten Word Spotting with Corrected Attributes." In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 1017–1024.

[Alm+14a] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. "Segmentation-free word spotting with exemplar SVMs." In: *Pattern Recognition* 47 (2014), pp. 3967–3978.

[Alm+14b] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. "Word Spotting and Recognition with Embedded Attributes." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36 (2014), pp. 2552–2566.

[Alz+21] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions." In: *Journal of Big Data* 8 (2021), p. 53.

[AD07] Esra Ataer and Pinar Duygulu. "Matching Ottoman Words." In: *2007 IEEE 15th Signal Processing and Communications Applications*. 2007, pp. 1–4.

[BL15] Artem Babenko and Victor Lempitsky. "Aggregating Local Deep Features for Image Retrieval." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1269–1277.

[BKC16] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. *SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation*. Tech. rep. 2016.

[BR11] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The concepts and technology behind search*. 2nd. Addison-Wesley Publishing Company, 2011.

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[BB15] Abhijit Bendale and Terrance Boult. "Towards Open World Recognition." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1893–1902.

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

[Ben+06] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy Layer-Wise Training of Deep Networks." In: *Advances in Neural Information Processing Systems*. Vol. 19. MIT Press, 2006.

[BM92] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256.

[Bha46] A. Bhattacharyya. "On a Measure of Divergence between Two Multinomial Populations." In: *Sankhyā: The Indian Journal of Statistics (1933-1960)* 7.4 (1946), pp. 401–406.

[Bie87] Irving Biederman. "Recognition-by-components: a theory of human image understanding." In: *Psychological review* 94.2 (1987), pp. 115–147.

[Bis09] Christopher M. Bishop. *Pattern recognition and machine learning*. 8. (corr. printing). Information science and statistics. Springer, 2009.

[BGM15a] Bartosz Bogacz, Michael Gertz, and Hubert Mara. "Character retrieval of vectorized cuneiform script." In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. 2015, pp. 326–330.

[BGM15b] Bartosz Bogacz, Michael Gertz, and Hubert Mara. "Cuneiform Character Similarity Using Graph Representations." In: *2015 20th Computer Vision Winter Workshop (CVWW'15)*. 2015.

[BHM16] Bartosz Bogacz, Nicholas Howe, and Hubert Mara. "Segmentation Free Spotting of Cuneiform Using Part Structured Models." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 301–306.

[BKM17] Bartosz Bogacz, Maximilian Klingmann, and Hubert Mara. "Automating Transliteration of Cuneiform from Parallel Lines with Sparse Data." In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 615–620.

[BM18] Bartosz Bogacz and Hubert Mara. "Feature Descriptors for Spotting 3D Characters on Triangular Meshes." In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2018, pp. 363–368.

[BM20] Bartosz Bogacz and Hubert Mara. "Period Classification of 3D Cuneiform Tablets with Geometric Neural Networks." In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, pp. 246–251.

[BZM07] Anna Bosch, Andrew Zisserman, and Xavier Munoz. "Representing shape with a spatial pyramid kernel." In: *Proceedings of the 6th ACM international conference on Image and video retrieval*. CIVR '07. Association for Computing Machinery, 2007, pp. 401–408.

[Bot91] Léon Bottou. "Stochastic Gradient Learning in Neural Networks." In: *Proceedings of Neuro-Nîmes 91*. EC2, 1991.

[Bot98] Léon Bottou. "Online Algorithms and Stochastic Approximations." In: *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[BB07] Léon Bottou and Olivier Bousquet. "The Tradeoffs of Large Scale Learning." In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2007.

[BRF21] Kai Brandenbusch, Eugen Rusakov, and Gernot A. Fink. "Context Aware Generation of Cuneiform Signs." In: *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I*. Springer-Verlag, 2021, pp. 65–79.

[Bra+10] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. "Visual Recognition with Humans in the Loop." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer, 2010, pp. 438–451.

[Bre96] Leo Breiman. "Bagging predictors." In: *Machine Learning* 24.2 (1996), pp. 123–140.

[Bro+20] Tom Brown *et al.* "Language Models are Few-Shot Learners." In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[CZF06] J. Chan, C. Ziftci, and D. Forsyth. "Searching Off-line Arabic Documents." In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1455–1462.

[CWB93] F.R. Chen, L.D. Wilcox, and D.S. Bloomberg. "Word spotting in scanned images using hidden Markov models." In: *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5. 1993, 1–4 vol.5.

[Cho+14] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. ACL, 2014, pp. 1724–1734.

[Cho16] François Chollet. *Information-theoretical label embeddings for large-scale image classification*. Tech. rep. arXiv, 2016.

[CHL05] S. Chopra, R. Hadsell, and Y. LeCun. "Learning a similarity metric discriminatively, with application to face verification." In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 539–546 vol. 1.

[CUH16] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016.

[CV95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine Learning* 20.3 (1995), pp. 273–297.

[Cre+18] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. "Generative Adversarial Networks: An Overview." In: *IEEE Signal Processing Magazine* 35.1 (2018), pp. 53–65.

[Cro+23] Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. "Diffusion Models in Vision: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.9 (2023), pp. 10850–10869.

[Cur44] Haskell B. Curry. "The Method of Steepest Descent for Non-Linear Minimization Problems." In: *Quarterly of Applied Mathematics* 2.3 (1944), pp. 258–261.

[DT05] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, pp. 886–893.

[Dan80] Per-Erik Danielsson. "Euclidean distance mapping." In: *Computer Graphics and Image Processing* 14.3 (1980), pp. 227–248.

[Dav12] Jean Ponce David A. Forsyth. *Computer Vision: A Modern Approach, 2nd Edition*. 2012.

[Dee+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. "Indexing by latent semantic analysis." In: *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm." In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pp. 1–38.

[Den+20] Tobias Dencker, Pablo Klinkisch, Stefan M. Maul, and Björn Ommer. "Deep learning of cuneiform sign detection with weak supervision using transliteration alignment." In: *PLOS ONE* 15.12 (2020), e0243039.

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[DG11] Jeff Donahue and Kristen Grauman. "Annotator rationales for visual recognition." In: *2011 International Conference on Computer Vision*. 2011, pp. 1395–1402.

[Dos+21] Alexey Dosovitskiy *et al.* "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale." In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[DRS11] Matthijs Douze, Arnau Ramisa, and Cordelia Schmid. "Combining attributes and Fisher vectors for efficient image retrieval." In: *CVPR 2011*. 2011, pp. 745–752.

[DL18] Simon Du and Jason Lee. "On the Power of Over-parametrization in Neural Networks with Quadratic Activation." In: PMLR, 2018, pp. 1329–1338.

[DHS11] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization." In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. Second edition. A Wiley-Interscience publication. John Wiley & Sons, Inc., 2001.

[Dug+00] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. "Incorporating Second-Order Functional Knowledge for Better Option Pricing." In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press, 2000.

[Edw+04] Jaety Edwards, Yee Teh, Roger Bock, Michael Maire, Grace Vesom, and David Forsyth. "Making Latin Manuscripts Searchable using gHMM' s." In: *Advances in Neural Information Processing Systems*. Vol. 17. MIT Press, 2004.

[ET93] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, 1993.

[ESE13] Mohamed Elhoseiny, Babak Saleh, and Ahmed Elgammal. "Write a Classifier: Zero-Shot Learning Using Purely Textual Descriptions." In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 2584–2591.

[Far+09] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. "Describing objects by their attributes." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 1778–1785.

[FWL16] Geli Fei, Shuai Wang, and Bing Liu. "Learning Cumulatively to Become More Knowledgeable." In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2016, pp. 1565–1574.

[FFP06] Li Fei-Fei, Rob Fergus, and Pietro Perona. "One-shot learning of object categories." In: *IEEE transactions on pattern analysis and machine intelligence* 28.4 (2006), pp. 594–611.

[FLP17] Rogerio Schmidt Feris, Christoph Lampert, and Devi Parikh. *Visual Attributes*. Advances in Computer Vision and Pattern Recognition. Springer International Publishing, 2017.

[FLF11] David Fernández, Josep Lladós, and Alicia Fornés. "Handwritten Word Spotting in Old Manuscript Images Using a Pseudo-structural Descriptor Organized in a Hash Structure." In: *Pattern Recognition and Image Analysis*. Lecture Notes in Computer Science. Springer, 2011, pp. 628–635.

[FZ07] Vittorio Ferrari and Andrew Zisserman. "Learning Visual Attributes." In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2007.

[Fey+18] Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. "SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 869–877.

[Fis+10] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. "HMM-based Word Spotting in Handwritten Documents Using Subword Models." In: *2010 20th International Conference on Pattern Recognition*. 2010, pp. 3416–3419.

[Fis+12] Andreas Fischer, Andreas Keller, Volkmar Frinken, and Horst Bunke. "Lexicon-free handwritten word spotting using character HMMs." In: *Pattern Recognition Letters*. Special Issue on Awards from ICPR 2010 33 (2012), pp. 934–942.

[Fis+13] Denis Fisseler, Frank Weichert, Gerfrid G. W. Müller, and Michele Cammarosano. "Towards an interactive and automated script feature analysis of 3D scanned cuneiform tablets." In: *4th Conference Scientific Computing and Cultural Heritage (SCCH 2013), pp. 1-10*. 2013.

[Fis19] Denis Bernd Fisseler. "Contributions to computer-aided analysis of cuneiform tablet fragments." PhD thesis. TU Dortmund University, 2019.

[For+11] Alicia Fornés, Volkmar Frinken, Andreas Fischer, Jon Almazán, Gabriel Jackson, and Horst Bunke. "A keyword spotting approach using blurred shape model-based descriptors." In: *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*. HIP '11. Association for Computing Machinery, 2011, pp. 83–90.

[FWS06] Rudolf J. Freund, William J. Wilson, and Ping Sa. *Regression Analysis : Statistical Modeling of a Response Variable*. Elsevier Science & Technology, 2006.

[FB14] Volkmar Frinken and Horst Bunke. "Continuous Handwritten Script Recognition." In: Springer, 2014, pp. 391–425.

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

[Fuk80] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position." In: *Biological Cybernetics* 36.4 (1980), pp. 193–202.

[FM82] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position." In: *Pattern Recognition* 15.6 (1982), pp. 455–469.

[FMI83] Kunihiko Fukushima, Sei Miyake, and Takayuki Ito. "Neocognitron: A neural network model for a mechanism of visual pattern recognition." In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13.5 (1983), pp. 826–834.

[GG16] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning." In: PMLR, 2016, pp. 1050–1059.

[Gat14] Basilis G. Gatos. "Imaging Techniques in Document Analysis Processes." In: Springer, 2014, pp. 73–131.

[GEB16] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2414–2423.

[GP21] Alexander Gepperth and Benedikt Pfülb. "Gradient-Based Training of Gaussian Mixture Models for High-Dimensional Streaming Data." In: *Neural Processing Letters* 53.6 (2021), pp. 4331–4348.

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

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In: JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks." In: JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[Glu69] Herbert A. Glucksman. *Classification of Mixed-font Alphabetics by Characteristic Loci*. Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force, 1969.

[GRK17] Lluís Gómez, Marçal Rusiñol, and Dimosthenis Karatzas. "LSDE: Levenshtein Space Deep Embedding for Query-by-String Word Spotting." In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 499–504.

[Goo04] Phillip I. Good. *Permutation, Parametric, and Bootstrap Tests of Hypotheses (Springer Series in Statistics)*. Springer-Verlag, 2004.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. Adaptive computation and machine learning. The MIT Press, 2016.

[Goo+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and Harnessing Adversarial Examples." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[Got13] Norbert Gottstein. "Ein stringentes Identifikations- und Suchsystemfür Keilschriftzeichen." In: *Mitteilungen der Deutschen Orient-Gesellschaft zu Berlin* (2013).

[Gra+06] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In: *Proceedings of the 23rd international conference on Machine learning*. ICML '06. Association for Computing Machinery, 2006, pp. 369–376.

[Gra+09] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. "A Novel Connectionist System for Unconstrained Handwriting Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5 (2009), pp. 855–868.

[Hah98] R. L. T. Hahnloser. "On the piecewise analysis of networks of linear threshold neurons." In: *Neural Networks: The Official Journal of the International Neural Network Society* 11.4 (1998), pp. 691–697.

[Hah+00] Richard H. R. Hahnloser, Rahul Sarpeshkar, Misha A. Mahowald, Rodney J. Douglas, and H. Sebastian Seung. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit." In: *Nature* 405.6789 (2000), pp. 947–951.

[HS88] C. Harris and M. Stephens. "A Combined Corner and Edge Detector." In: *Proceedings of the 4th Alvey Vision Conference*. 1988, pp. 147–151.

[He+15a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[He+15b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.9 (2015), pp. 1904–1916.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[Hen+16] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. "Generating Visual Explanations." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer International Publishing, 2016, pp. 3–19.

[HS06] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks." In: *Science* 313.5786 (2006), pp. 504–507.

[HOT06] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." In: *Neural Computation* 18.7 (2006), pp. 1527–1554.

[Hin+12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors." In: *CoRR* abs/1207.0580 (2012). arXiv: 1207.0580.

[Hoc98] Sepp Hochreiter. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.2 (1998), pp. 107–116.

[HA15] Elad Hoffer and Nir Ailon. "Deep Metric Learning Using Triplet Network." In: *Similarity-Based Pattern Recognition*. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 84–92.

[HSW89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." In: *Neural Networks* 2.5 (1989), pp. 359–366.

[How13] Nicholas R. Howe. "Part-Structured Inkball Models for One-Shot Handwritten Word Spotting." In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, pp. 582–586.

[HSS18] Jie Hu, Li Shen, and Gang Sun. "Squeeze-and-Excitation Networks." In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7132–7141.

[HBW07] Gang Hua, Matthew Brown, and Simon Winder. "Discriminant Embedding for Local Image Descriptors." In: *2007 IEEE 11th International Conference on Computer Vision*. 2007, pp. 1–8.

[Hua+17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely Connected Convolutional Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269.

[Hua+15] Junshi Huang, Rogerio Feris, Qiang Chen, and Shuicheng Yan. "Cross-Domain Image Retrieval with a Dual Attribute-Aware Ranking Network." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1062–1070.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: accelerating deep network training by reducing internal covariate shift." In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. JMLR.org, 2015, pp. 448–456.

[Jad+14] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition*. Tech. rep. arXiv, 2014.

[Jad+16] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Reading Text in the Wild with Convolutional Neural Networks." In: *International Journal of Computer Vision* 116.1 (2016), pp. 1–20.

[JVZ14] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. "Deep Features for Text Spotting." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 512–528.

[Joa02] Thorsten Joachims. "Optimizing search engines using clickthrough data." In: *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '02. Association for Computing Machinery, 2002, pp. 133–142.

[JLG78] Stig Johansson, Geoffrey Leech, and Helen Goodluck. *Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers*. Department of English, University of Oslo, Oslo, Norway. 1978.

[Jos13] Florent Perronnin Jose Rodriguez. "Label embedding for text recognition." In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[Joy21] James Joyce. "Bayes' Theorem." In: *The Stanford Encyclopedia of Philosophy*. Fall 2021. Metaphysics Research Lab, Stanford University, 2021.

[KLP01] Shaun Kane, Andrew Lehman, and Elizabeth Partridge. *Indexing George Washington's Handwritten Manuscripts*. Kane2001. 2001.

[KB19] Mahmut Kaya and Hasan Şakir Bilge. "Deep Metric Learning: A Survey." In: *Symmetry* 11.9 (2019), p. 1066.

[KGG97] P. Keaton, H. Greenspan, and R. Goodman. "Keyword spotting for cursive document retrieval." In: *Proceedings Workshop on Document Image Analysis (DIA'97)*. 1997, pp. 74–81.

[KZ11] Andrea Vedaldi Ken Chatfield Victor Lempitsky and Andrew Zisserman. "The devil is in the details: an evaluation of recent feature encoding methods." In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 76.1–76.12.

[KBS11] Douglas J. Kennard, William A. Barrett, and Thomas W. Sederberg. "Word Warping for Offline Handwriting Recognition." In: *2011 International Conference on Document Analysis and Recognition*. 2011, pp. 1349–1353.

[Kes+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima." In: *International Conference on Learning Representations (ICLR)*. 2017.

[KH93] Siamak Khoubyari and Jonathan J. Hull. "Keyword location in noisy document image." In: *In 2nd Annual Symposium on Document Analysis and Information Retrieval*. 1993, pp. 217–231.

[Khu+23] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. "Natural language processing: state of the art, current trends and challenges." In: *Multimedia Tools and Applications* 82 (2023), pp. 3713–3744.

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[Kir79] David G. Kirkpatrick. "Efficient computation of continuous skeletons." In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*. 1979, pp. 18–27.

[Kle+13] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. "CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting." In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, pp. 560–564.

[Kol+00] A. Kolcz, J. Alspector, M. Augusteijn, R. Carlson, and G. Viorel Popescu. "A Line-Oriented Approach to Word Spotting in Handwritten Documents." In: *Pattern Analysis & Applications* 3.2 (2000), pp. 153–168.

[KWD14] Alon Kovalchuk, Lior Wolf, and Nachum Dershowitz. "A Simple and Fast Word Spotting Method." In: *2014 14th International Conference on Frontiers in Handwriting Recognition*. 2014, pp. 3–8.

[KPG15] Adriana Kovashka, Devi Parikh, and Kristen Grauman. "WhittleSearch: Interactive Image Search with Relative Attribute Feedback." In: *International Journal of Computer Vision* 115.2 (2015), pp. 185–210.

[KVG11] Adriana Kovashka, Sudheendra Vijayanarasimhan, and Kristen Grauman. "Actively selecting annotations among objects and attributes." In: *2011 International Conference on Computer Vision*. 2011, pp. 1403–1410.

[Kra81] S.N. Kramer. *History Begins at Sumer: Thirty-Nine Firsts in Recorded History*. University of Pennsylvania Press, Incorporated, 1981.

[Kri+18] Nils M. Kriege, Matthias Fey, Denis Fisseler, Petra Mutzel, and Frank Weichert. "Recognizing Cuneiform Signs Using Graph Based Methods." In: *Proceedings of the International Workshop of Cost-Sensitive Learning, (COST'18)*. PMLR, 2018, pp. 31–44.

[KDJ23] Praveen Krishnan, Kartik Dutta, and C. V. Jawahar. "HWNet v3: a joint embedding framework for recognition and retrieval of handwritten text." In: *International Journal on Document Analysis and Recognition (IJDAR)* 26.4 (2023), pp. 401–417.

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

[KDJ18] Praveen Krishnan, Kartik Dutta, and C.V. Jawahar. "Word Spotting and Recognition Using Deep Embedding." In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. 2018, pp. 1–6.

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

[KJ16] Praveen Krishnan and CV Jawahar. "Generating synthetic data for text recognition." In: *arXiv preprint arXiv:1608.04224* (2016).

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Vol. 25. Curran Associates, Inc., 2012.

[KA94] Shyh-shiaw Kuo and Oscar E. Agazzi. "Keyword Spotting in Poorly Printed Documents using Pseudo 2-D Hidden Markov Models." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 16 (1994), pp. 842–848.

[LNH09] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. "Learning to detect unseen object classes by between-class attribute transfer." In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 951–958.

[LNH14] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. "Attribute-Based Classification for Zero-Shot Visual Object Categorization." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36.3 (2014), pp. 453–465.

[LEB08] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. "Zero-data learning of new tasks." In: *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*. AAAI'08. AAAI Press, 2008, pp. 646–651.

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

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories." In: vol. 2. IEEE, 2006, pp. 2169–2178.

[LeC+89a] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition." In: *Neural Computation* 1.4 (1989), pp. 541–551.

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

[LeC+89b] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, R. Howard, Wayne Hubbard, and Lawrence Jackel. "Handwritten Digit Recognition with a Back-Propagation Network." In: *Advances in Neural Information Processing Systems*. Vol. 2. Morgan-Kaufmann, 1989.

[LeC+98] Yann LeCun, Leon Bottou, Genevieve B. Orr, and Klaus Robert Müller. "Efficient BackProp." In: Lecture Notes in Computer Science. Springer, 1998, pp. 9–50.

[LEN02] Christina Leslie, Eleazar Eskin, and William Stafford Noble. "The spectrum kernel: a string kernel for SVM protein classification." In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing* (2002), pp. 564–575.

[Lev66] Vladimir Iosifovich Levenshtein. "Binary codes capable of correcting deletions, insertions and reversals." In: *Soviet Physics Doklady* 10.8 (1966), pp. 707–710.

[Ley+09] Yann Leydier, Asma Ouji, Frank LeBourgeois, and Hubert Emptoz. "Towards an omnilingual word retrieval system for ancient manuscripts." In: *Pattern Recognition* 42.9 (2009), pp. 2089–2105.

[LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. "Network In Network." In: *International Conference on Learning Representations, Conference Track*. 2014.

[Lin+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 740–755.

[Liu+13] Luoqi Liu, Hui Xu, Junliang Xing, Si Liu, Xi Zhou, and Shuicheng Yan. ""Wow! you are so beautiful today!"" In: *Proceedings of the 21st ACM international conference on Multimedia*. MM '13. Association for Computing Machinery, 2013, pp. 3–12.

[LS07] J. Lladós and G. Sánchez. "Indexing Historical Documents by Word Shape Signatures." In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 1. 2007, pp. 362–366.

[Lla+12] Josep Lladós, Marçal Rusiñol, Alicia Fornés, David Fernández-Mota, and Anjan Dutta. "On the influence of word representations for handwritten word spotting in historical documents." In: *International Journal of Pattern Recognition and Artificial Intelligence* 26.05 (2012), p. 1263002.

[Llo82] S. Lloyd. "Least squares quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137.

[Lod+00] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Christopher Watkins. "Text Classification using String Kernels." In: *Advances in Neural Information Processing Systems*. Vol. 13. MIT Press, 2000.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

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

[LCH03] Bin Luo, Richard C. Wilson, and Edwin R. Hancock. "Spectral embedding of graphs." In: *Pattern Recognition* 36.10 (2003), pp. 2213–2230.

[Luo+16] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks." In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[Maa13] Andrew L. Maas. "Rectifier Nonlinearities Improve Neural Network Acoustic Models." In: *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing (WDLASL 2013)*. 2013.

[MGE11] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. "Ensemble of exemplar-SVMs for object detection and beyond." In: *2011 International Conference on Computer Vision*. 2011, pp. 89–96.

[Man+96] R. Manmatha, Chengfeng Han, E. M. Riseman, and W. B. Croft. "Indexing handwriting using word matching." In: *Proceedings of the first ACM international conference on Digital libraries*. DL '96. Association for Computing Machinery, 1996, pp. 151–159.

[MHR96] R. Manmatha, Chengfeng Han, and E.M. Riseman. "Word spotting: a new approach to indexing handwriting." In: *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1996, pp. 631–637.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[Mar12] Hubert Mara. "Multi-Scale Integral Invariants for Robust Character Extraction from Irregular Polygon Mesh Data." PhD thesis. University of Heidelberg, 2012.

[MB19] Hubert Mara and Bartosz Bogacz. "Breaking the Code on Broken Tablets: The Learning Challenge for Annotated Cuneiform Script in Normalized 2D and 3D Datasets." In: *2019 International Conference on Document Analysis and Recognition (ICDAR)*. 2019, pp. 148–153.

[MK13] Hubert Mara and Susanne Krömker. "Vectorization of 3D-Characters by Integral Invariant Filtering of High-Resolution Triangular Meshes." In: *2013 12th International Conference on Document Analysis and Recognition*. 2013, pp. 62–66.

[MB99] U.-V. Marti and H. Bunke. "A full English sentence database for off-line handwriting recognition." In: *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*. 1999, pp. 705–708.

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

[Mas+16] J. Massa, B. Bogacz, S. Krömker, and H. Mara. "Cuneiform Detection in Vectorized Raster Images." In: *2016 21st Compuer Vision Winter Workshop (CVWW'16)*. 2016.

[MO16] Bruno Meißner and Karl Oberhuber. *Die Keilschrift*. De Gruyter, 2016.

[Mey+03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. "Discrete Differential-Geometry Operators for Triangulated 2-Manifolds." In: *Visualization and Mathematics III*. Mathematics and Visualization. Springer, 2003, pp. 35–57.

[Mik+13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. Tech. rep. arXiv, 2013.

[Min+22] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. "Image Segmentation Using Deep Learning: A Survey." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (2022), pp. 3523–3542.

[MP69] Marvin Minsky and Seymour Papert. *Perceptrons*. Perceptrons. M.I.T. Press, 1969.

[MPV06] Douglas C. Montgomery, Elizabeth A. Peck, and Geoffrey G. Vining. *Introduction to Linear Regression Analysis (4th ed.)* Wiley & Sons, 2006.

[Mül00] Gerfried G. W. Müller. *Hethitologie Portal Mainz*. 2000.

[Mun57] James Munkres. "Algorithms for the Assignment and Transportation Problems." In: *Journal of the Society for Industrial and Applied Mathematics* 5.1 (1957), pp. 32–38.

[Mur12] Kevin P. Murphy. *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series. The MIT Press, 2012.

[MRR80] C. Myers, L. Rabiner, and A. Rosenberg. "An investigation of the use of dynamic time warping for word spotting and connected speech recognition." In: *ICASSP '80. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5. 1980, pp. 173–177.

[NH10] Vinod Nair and Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines." In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Omnipress, 2010, pp. 807–814.

[NW72] J. A. Nelder and R. W. M. Wedderburn. "Generalized Linear Models." In: *Journal of the Royal Statistical Society. Series A (General)* 135.3 (1972), pp. 370–384.

[Ney+18] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. *Towards Understanding the Role of Over-Parametrization in Generalization of Neural Networks*. Tech. rep. arXiv, 2018.

[Ney+19] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. "The role of over-parametrization in generalization of neural networks." In: *International Conference on Learning Representations*. 2019.

[Nie15] M.A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[NJT06] Eric Nowak, Frédéric Jurie, and Bill Triggs. "Sampling Strategies for Bag-of-Features Image Classification." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer, 2006, pp. 490–503.

[NL11] Sebastian Nowozin and Christoph H. Lampert. "Structured Learning and Prediction in Computer Vision." In: *Foundations and Trends® in Computer Graphics and Vision* 6.3–4 (2011), pp. 185–365.

[OD11] Stephen O'Hara and Bruce A. Draper. *Introduction to the Bag of Features Paradigm for Image Classification and Retrieval*. Tech. rep. 2011.

[OGL19] Umut Ozaydin, Theodoros Georgiou, and Michael Lew. "A Comparison of CNN and Classic Features for Image Retrieval." In: *2019 International Conference on Content-Based Multimedia Indexing (CBMI)* (2019), pp. 1–4.

[PY10] Sinno Jialin Pan and Qiang Yang. "A Survey on Transfer Learning." In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.

[PG11] Devi Parikh and Kristen Grauman. "Relative attributes." In: *2011 International Conference on Computer Vision*. 2011, pp. 503–510.

[Pat+14] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. "The SUN Attribute Database: Beyond Categories for Deeper Scene Understanding." In: *International Journal of Computer Vision* 108.1 (2014), pp. 59–81.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1532–1543.

[PR09] Florent Perronnin and Jose A. Rodriguez-Serrano. "Fisher Kernels for Handwritten Word-spotting." In: *2009 10th International Conference on Document Analysis and Recognition*. 2009, pp. 106–110.

[PSM10] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. "Improving the Fisher Kernel for Large-Scale Image Classification." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer, 2010, pp. 143–156.

[Phi+10] James Philbin, Michael Isard, Josef Sivic, and Andrew Zisserman. "Descriptor Learning for Efficient Retrieval." In: *European Conference on Computer Vision (ECCV)*. Lecture Notes in Computer Science. Springer, 2010, pp. 677–691.

[Pla99] J. C. Platt. "Probabilistic outputs for support vector machines for pattern recognition." In: *Advances in Large margin Classifiers. Kluwer Academic Publishers, Boston* 64 (1999), pp. 975–1005.

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

[Pol64] B. T. Polyak. "Some methods of speeding up the convergence of iteration methods." In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[Pra+24] Rohit Prabhavalkar, Takaaki Hori, Tara N. Sainath, Ralf Schluter, and Shinji Watanabe. "End-to-End Speech Recognition: A Survey." In: *IEEE/ACM Transactions on Audio Speech and Language Processing* 32 (2024), pp. 325–351.

[Pra+16] Ioannis Pratikakis, Konstantinos Zagoris, Basilis Gatos, Joan Puigcerver, Alejandro H. Toselli, and Enrique Vidal. "ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016)." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 613–618.

[Qi+17] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. "PointNet++: deep hierarchical feature learning on point sets in a metric space." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Curran Associates Inc., 2017, pp. 5105–5114.

[Rad+18] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, *et al. Improving language understanding by generative pre-training*. Tech. rep. 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[RKC21] Anuj Rai, Narayanan C. Krishnan, and Sukalpa Chanda. "Pho(SC)Net: An Approach Towards Zero-Shot Word Image Recognition in Historical Documents." In: *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I*. Springer-Verlag, 2021, pp. 19–33.

[RLM03] Toni M. Rath, Victor Lavrenko, and R. Manmatha. "A statistical approach to retrieving historical manuscript images without recognition." In: *Center for Intelligent Information Retrieval, University of Massachusetts* (2003).

[RM03] Toni M. Rath and R. Manmatha. "Word Image Matching Using Dynamic Time Warping." In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2003, pp. 521–527.

[RM07] Toni M. Rath and R. Manmatha. "Word spotting for historical documents." In: *International Journal of Document Analysis and Recognition (IJDAR)* 9.2 (2007), pp. 299–299.

[RL17] Sachin Ravi and Hugo Larochelle. "Optimization as a Model for Few-Shot Learning." In: *International Conference on Learning Representations (ICLR)*. 2017.

[Reb+17] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. "iCaRL: Incremental Classifier and Representation Learning." In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5533–5542.

[Ren+17] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39.6 (2017), pp. 1137–1149.

[Res+21] Christopher Rest, Denis Fisseler, Frank Weichert, Turna Somel, and Gerfrid G.W. Müller. "Illumination-based augmentation for cuneiform deep neural sign classification." In: *Journal on Computing and Cultural Heritage* (2021).

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

[RGP15] Jose A. Rodriguez-Serrano, Albert Gordo, and Florent Perronnin. "Label Embedding: A Frugal Baseline for Text Recognition." In: *International Journal of Computer Vision* 113.3 (2015), pp. 193–207.

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

[Roh+89] J.R. Rohlicek, W. Russell, S. Roukos, and H. Gish. "Continuous hidden Markov modeling for speaker-independent word spotting." In: *International Conference on Acoustics, Speech, and Signal Processing,* 1989, 627–630 vol.1.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation." In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 234–241.

[Ros58] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65 (1958), pp. 386–408.

[Rot+13] L. Rothacker, G. A. Fink, P. Banerjee, U. Bhattacharya, and B. B. Chaudhuri. "Bag-of-features HMMs for segmentation-free Bangla word spotting." In: *Proceedings of the 4th International Workshop on Multilingual OCR*. MOCR '13. Association for Computing Machinery, 2013, pp. 1–5.

[Rot19] Leonard Rothacker. "Segmentation-free word spotting with bag-of-features hidden Markov models." PhD thesis. TU Dortmund University, 2019.

[RF15] Leonard Rothacker and Gernot A. Fink. "Segmentation-free query-by-string word spotting with Bag-of-Features HMMs." In: IEEE, 2015, pp. 661–665.

[Rot+15] Leonard Rothacker, Denis Fisseler, Gerfrid G. W. Müller, Frank Weichert, and Gernot A. Fink. "Retrieving Cuneiform Structures in a Segmentation-free Word Spotting Framework." In: *Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing*. HIP '15. Association for Computing Machinery, 2015, pp. 129–136.

[RMF13] Leonard Rothacker, M, and Gernot A. Fink. "Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents." In: IEEE, 2013, pp. 1305–1309.

[Rot+17] Leonard Rothacker, Sebastian Sudholt, Eugen Rusakov, Matthias Kasperidus, and Gernot A. Fink. "Word Hypotheses for Segmentation-Free Word Spotting in Historic Document Images." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 1174–1179.

[RVF12] Leonard Rothacker, Szilárd Vajda, and Gernot A. Fink. "Bag-of-Features Representations for Offline Handwriting Recognition Applied to Arabic Script." In: *2012 International Conference on Frontiers in Handwriting Recognition*. 2012, pp. 149–154.

[RWF21] Leonard Rothacker, Fabian Wolf, and Gernot 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.

[RFR03] Jamie L. Rothfeder, Shaolei Feng, and Toni M. Rath. "Using Corner Feature Correspondences to Rank Word Images by Similarity." In: *Conference on Computer Vision and Pattern Recognition Workshop*. Vol. 3. 2003, pp. 30–30.

[RF18] Fernando Moya Rueda and Gernot A. Fink. "Learning Attribute Representation for Human Activity Recognition." In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, pp. 523–528.

[RF21] Fernando Moya Rueda and Gernot A. Fink. "From Human Pose to On-Body Devices for Human-Activity Recognition." In: *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, pp. 10066–10073.

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

[Rus+19] Eugen Rusakov, Kai Brandenbusch, Denis Fisseler, Turna Somel, Gernot A. Fink, Frank Weichert, and Gerfrid G. W. Müller. "Generating Cuneiform Signs with Cycle-Consistent Adversarial Networks." In: *Proceedings of the 5th International Workshop on Historical Document Imaging and Processing*. 2019, pp. 19–24.

[Rus+18] Eugen Rusakov, Leonard Rothacker, Hyunho Mo, and Gernot A. Fink. "A Probabilistic Retrieval Model for Word Spotting Based on Direct Attribute Prediction." In: *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2018, pp. 38–43.

[Rus+20] Eugen Rusakov, Turna Somel, Gernot A. Fink, and Gerfrid G.W. Müller. "Towards Query-by-eXpression Retrieval of Cuneiform Signs." In: *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2020, pp. 43–48.

[Rus+21] Eugen Rusakov, Turna Somel, Gerfrid G. W. Müller, and Gernot A. Fink. "Embedded Attributes for Cuneiform Sign Spotting." In: *Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II*. Springer-Verlag, 2021, pp. 291–305.

[Rus+11] Marcal Rusinol, David Aldavert, Ricardo Toledo, and Josep Llados. "Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method." In: *2011 International Conference on Document Analysis and Recognition*. 2011, pp. 63–67.

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

[Rus+15b] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. "Towards query-by-speech handwritten keyword spotting." In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. Aug. 2015, pp. 501–505.

[RL14] Marçal Rusiñol and Josep Lladós. "Boosting the handwritten word spotting experience by including the user in the loop." In: *Pattern Recognition*. Handwriting Recognition and other PR Applications 47.3 (2014), pp. 1063–1072.

[Rus+15c] Olga Russakovsky *et al.* "ImageNet Large Scale Visual Recognition Challenge." In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[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.

[SA96] E Salinas and L F Abbott. "A model of multiplicative neural responses in parietal cortex." In: *Proceedings of the National Academy of Sciences of the United States of America* 93.21 (1996), pp. 11956–11961.

[Sán+13] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. "Image Classification with the Fisher Vector: Theory and Practice." In: *International Journal of Computer Vision* 105 (2013), pp. 222–245.

[San+09] Rodolfo P. dos Santos, Gabriela S. Clemente, Tsang Ing Ren, and George D.C. Cavalcanti. "Text Line Segmentation Based on Morphology and Histogram Projection." In: *2009 10th International Conference on Document Analysis and Recognition*. 2009, pp. 651–655.

[Sch82] Herbert F. Schantz. *The history of OCR, optical character recognition*. [Manchester Center, Vt.] : Recognition Technologies Users Association, 1982.

[Sch+12] Walter J. Scheirer, Neeraj Kumar, Peter N. Belhumeur, and Terrance E. Boult. "Multi-attribute spaces: Calibration for attribute fusion and similarity search." In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 2933–2940.

[Sch+13] Walter J. Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E. Boult. "Toward Open Set Recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7 (2013), pp. 1757–1772.

[SL91] Guy L. Scott and H. Christopher Longuet-Higgins. "An Algorithm for Associating the Features of Two Images." In: *Proceedings: Biological Sciences* 244.1309 (1991), pp. 21–26.

[Ser+14] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. "Overfeat: Integrated recognition, localization and detection using convolutional networks." In: *2nd International Conference on Learning Representations, ICLR 2014*. 2014.

[SRG16] Giorgos Sfikas, George Retsinas, and Basilis Gatos. "Zoning Aggregated Hypercolumns for Keyword Spotting." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 283–288.

[Sha48] C. E. Shannon. "A mathematical theory of communication." In: *The Bell System Technical Journal* 27.4 (1948), pp. 623–656.

[SK15] Arjun Sharma and Pramod Sankar K. "Adapting off-the-shelf CNNs for word spotting & recognition." In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. 2015, pp. 986–990.

[SQL13] Viktoriia Sharmanska, Novi Quadrianto, and Christoph H. Lampert. "Learning to Rank Using Privileged Information." In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 825–832.

[SJ12] Ravi Shekhar and C.V. Jawahar. "Word Image Retrieval Using Bag of Visual Words." In: *2012 10th IAPR International Workshop on Document Analysis Systems*. 2012, pp. 297–301.

[She+11] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. "Weisfeiler-Lehman Graph Kernels." In: *Journal of Machine Learning Research* 12.77 (2011), pp. 2539–2561.

[SHX15] Zhiyuan Shi, Timothy M. Hospedales, and Tao Xiang. "Transferring a semantic representation for person re-identification and search." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4184–4193.

[SK19] Connor Shorten and Taghi M. Khoshgoftaar. "A survey on Image Data Augmentation for Deep Learning." In: *Journal of Big Data* 6.1 (2019), p. 60.

[SVZ14] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps." In: *Workshop at International Conference on Learning Representations*. 2014.

[SZ15] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015.

[SZ03] Sivic and Zisserman. "Video Google: a text retrieval approach to object matching in videos." In: *Proceedings Ninth IEEE International Conference on Computer Vision*. 2003, 1470–1477 vol.2.

[Smi07] R. Smith. "An Overview of the Tesseract OCR Engine." In: *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. Vol. 2. 2007, pp. 629–633.

[SAC07] Mark D. Smucker, James Allan, and Ben Carterette. "A comparison of statistical significance tests for information retrieval evaluation." In: *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. CIKM '07. Association for Computing Machinery, 2007, pp. 623–632.

[Soc+13] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. "Zero-Shot Learning Through Cross-Modal Transfer." In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc., 2013.

[Spr+15] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. "Striving for Simplicity: The All Convolutional Net." In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*. 2015.

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[SGS15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. *Highway Networks*. Tech. rep. arXiv, 2015.

[Ste+22] Matteo Stefanini, Marcella Cornia, Lorenzo Baraldi, Silvia Cascianelli, Giuseppe Fiameni, and Rita Cucchiara. "From Show to Tell: A Survey on Deep Learning-based Image Captioning." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022), pp. 1–1.

[Sud18] Sebastian Sudholt. "Learning attribute representations with deep convolutional neural networks for word spotting." PhD thesis. TU Dortmund University, 2018.

[SF15] Sebastian Sudholt and Gernot A. Fink. "A Modified Isomap Approach to Manifold Learning in Word Spotting." In: *Pattern Recognition*. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 529–539.

[SF16] Sebastian Sudholt and Gernot A. Fink. "PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 277–282.

[SF17] Sebastian Sudholt and Gernot A. Fink. "Evaluating Word String Embeddings and Loss Functions for CNN-Based Word Spotting." In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 493–498.

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

[SRF15] Sebastian Sudholt, Leonard Rothacker, and Gernot A. Fink. "Learning local image descriptors for word spotting." In: *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Aug. 2015, pp. 651–655.

[SRF17] Sebastian Sudholt, Leonard Rothacker, and Gernot A. Fink. "Query-by-Online Word Spotting Revisited: Using CNNs for Cross-Domain Retrieval." In: *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01. 2017, pp. 481–486.

[Sut+13] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. "On the importance of initialization and momentum in deep learning." In: PMLR, 2013, pp. 1139–1147.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks." In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 27. 2014.

[Sze+15] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions." In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2015, pp. 1–9.

[Sze+17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. "Inception-v4, inception-ResNet and the impact of residual connections on learning." In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI'17. AAAI Press, 2017, pp. 4278–4284.

[Sze+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the Inception Architecture for Computer Vision." In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826.

[Sze22] Richard Szeliski. *Computer Vision - Algorithms and Applications, Second Edition*. Texts in Computer Science. Springer, 2022.

[TL19] Mingxing Tan and Quoc Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks." In: PMLR, 2019, pp. 6105–6114.

[Tan+19] Chufeng Tang, Lu Sheng, Zhao-Xiang Zhang, and Xiaolin Hu. "Improving Pedestrian Attribute Recognition With Weakly-Supervised Multi-Scale Attribute-Specific Localization." In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 4996–5005.

[TT09] Kengo Terasawa and Yuzuru Tanaka. "Slit Style HOG Feature for Document Image Word Spotting." In: *2009 10th International Conference on Document Analysis and Recognition*. 2009, pp. 116–120.

[Tho+15] Simon Thomas, Clément Chatelain, Laurent Heutte, Thierry Paquet, and Yousri Kessentini. "A deep HMM model for multiple keywords spotting in handwritten documents." In: *Pattern Analysis and Applications* 18.4 (2015), pp. 1003–1015.

[TH+12] Tijmen Tieleman, Geoffrey Hinton, *et al.* "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[ULH19] Julián Urbano, Harlley Lima, and Alan Hanjalic. "Statistical Significance Testing in Information Retrieval: An Empirical Analysis of Type I, Type II and Type III Errors." In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Association for Computing Machinery, 2019, pp. 505–514.

[Van18] Sylvie Vanséveren. *Unicode cuneiform fonts for macintosh and windows*. 2018.

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is All you Need." In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[Vis+10] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. "Graph Kernels." In: *Journal of Machine Learning Research* 11.40 (2010), pp. 1201–1242.

[Vit67] A. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 260–269.

[Wan+19] Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. "A Survey of Zero-Shot Learning: Settings, Methods, and Applications." In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (2019), 13:1–13:37.

[War+14] David Warde-Farley, Ian J. Goodfellow, Aaron C. Courville, and Yoshua Bengio. "An empirical analysis of dropout in piecewise linear networks." In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. 2014.

[Was54] George Washington. *George Washington Papers, Series 2, Letterbooks 1754-1799: Letterbook 1, Aug. 11, 1754 - Dec. 25, 1755*. 1754.

[Was23] George Washington. *About this Collection | George Washington Papers | Digital Collections | Library of Congress*. 2023.

[WRF16] Christian Wieprecht, Leonard Rothacker, and Gernot A. Fink. "Word Spotting in Historical Document Collections with Online-Handwritten Queries." In: *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. Apr. 2016, pp. 162–167.

[WB16] Tomas Wilkinson and Anders Brun. "Semantic and Verbatim Word Spotting Using Deep Neural Networks." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 307–312.

[WLB17] Tomas Wilkinson, Jonas Lindström, and Anders Brun. "Neural Ctrl-F: Segmentation-Free Query-by-String Word Spotting in Handwritten Manuscript Collections." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 4443–4452.

[WLB20] Tomas Wilkinson, Jonas Lindström, and Anders Brun. *Neural Word Search in Historical Manuscript Collections*. Tech. rep. arXiv, 2020.

[WBF20] Fabian Wolf, Kai Brandenbusch, and Gernot A. Fink. "Improving Handwritten Word Synthesis for Annotation-free Word Spotting." In: IEEE, 2020, pp. 61–66.

[WF20] Fabian Wolf and Gernot A. Fink. "Annotation-Free Learning of Deep Representations for Word Spotting Using Synthetic Data and Self Labeling." In: *Document Analysis Systems*. Lecture Notes in Computer Science. Springer International Publishing, 2020, pp. 293–308.

[WF22a] Fabian Wolf and Gernot A. Fink. "Combining Self-training and Minimal Annotations for Handwritten Word Recognition." In: *Frontiers in Handwriting Recognition: 18th International Conference, ICFHR 2022, Hyderabad, India, December 4–7, 2022, Proceedings*. Springer-Verlag, 2022, pp. 300–315.

[WF22b] Fabian Wolf and Gernot A. Fink. "Self-Training of Handwritten Word Recognition for Synthetic-to-Real Adaptation." In: *2022 26th International Conference on Pattern Recognition (ICPR)*. 2022, pp. 3885–3892.

[Wol96] David H. Wolpert. "The Lack of A Priori Distinctions Between Learning Algorithms." In: *Neural Computation* 8.7 (1996), pp. 1341–1390.

[WSH20] Xiongwei Wu, Doyen Sahoo, and Steven C. H. Hoi. "Recent advances in deep learning for object detection." In: *Neurocomputing* 396 (2020), pp. 39–64.

[Xie+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated Residual Transformations for Deep Neural Networks." In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5987–5995.

[YM12] Ismet Zeki Yalniz and R. Manmatha. "An Efficient Framework for Searching Text in Noisy Document Images." In: *2012 10th IAPR International Workshop on Document Analysis Systems*. 2012, pp. 48–52.

[Yan+15] Shuo Yang, Ping Luo, Chen-Change Loy, and Xiaoou Tang. "From Facial Parts Responses to Face Detection: A Deep Learning Approach." In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 3676–3684.

[YRT89] Steve J. Young, N. H. Russell, and J. H. Simon Thornton. *Token passing: a simple conceptual model for connected speech recognition systems*. Tech. rep. Cambridge University Engineering Department, 1989.

[YK16] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions." In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 2016.

[ZPG17] Konstantinos Zagoris, Ioannis Pratikakis, and Basilis Gatos. "Unsupervised Word Spotting in Historical Handwritten Document Images Using Document-Oriented Local Features." In: *IEEE Transactions on Image Processing* 26.8 (2017), pp. 4032–4041.

[Zei12] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. Tech. rep. arXiv, 2012.

[ZF14] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks." In: *Computer Vision – ECCV 2014*. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 818–833.

[ZSH03] Bin Zhang, Sargur N. Srihari, and Chen Huang. "Word image retrieval using binary features." In: *Document Recognition and Retrieval XI*. Vol. 5296. SPIE, 2003, pp. 45–53.

[Zha+14] Ning Zhang, Manohar Paluri, Marc'Aurelio Ranzato, Trevor Darrell, and Lubomir Bourdev. "PANDA: Pose Aligned Networks for Deep Attribute Modeling." In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1637–1644.

[Zha+16] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Learning Deep Representation for Face Alignment with Auxiliary Attributes." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38.5 (2016), pp. 918–930.

[Zha94] Zhengyou Zhang. "Iterative point matching for registration of free-form curves and surfaces." In: *International Journal of Computer Vision* 13.2 (1994), pp. 119–152.

[ZYT18] Liang Zheng, Yi Yang, and Qi Tian. "SIFT Meets CNN: A Decade Survey of Instance Retrieval." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.5 (2018), pp. 1224–1244.

[Zho+16a] Zhuoyao Zhong, Weishen Pan, Lianwen Jin, Harold Mouchère, and Christian Viard-Gaudin. "SpottingNet: Learning the Similarity of Word Images with Convolutional Neural Network for Word Spotting in Handwritten Historical Documents." In: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, pp. 295–300.

[Zho+16b] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. "Learning Deep Features for Discriminative Localization." In: IEEE, 2016, pp. 2921–2929.

[Zhu+17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2242–2251.

[Zhu+21] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. "A Comprehensive Survey on Transfer Learning." In: *Proceedings of the IEEE* 109.1 (2021), pp. 43–76.

[ZB02] M. Zimmermann and H. Bunke. "Automatic segmentation of the IAM off-line database for handwritten English text." In: *2002 International Conference on Pattern Recognition*. Vol. 4. 2002, 35–39 vol.4.

[ZL17] Barret Zoph and Quoc Le. "Neural Architecture Search with Reinforcement Learning." In: *International Conference on Learning Representations*. 2017.

[Zou+19] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." In: (2019). arXiv: 1905.05055 [cs].

INDEX