technische universität
dortmund

Attribute Representations for Word Spotting in Arabic Handwriting

Bachelor Thesis

Hatem Hamad
May 16, 2020

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Fabian Wolf, M.Sc.

Department of Computer Science
TU Dortmund University
www.cs.tu-dortmund.de

CONTENTS

1

INTRODUCTION

The urge to automate trivial labour has been a compelling motivation for humans. At first, we humans tried to harness mechanical machines to do none intellectual work for us. Then, we tried adding humble intellectual capabilities to our machines; computers were invented. When computers were invented, tasks that used to take humans years to finish could then be completed in seconds. Based on this feature, computer scientists and engineers hoped that computers could tackle more difficult perceptual tasks like recognising objects in images, distinguishing faces, or transcribing speeches or handwritten books. Even though these tasks may seem unrelated, they all boil down to the task of learning recurrent patterns in a medium.

Generally, research in pattern recognition field aims to enable machines to mimic and - eventually - master the human perceptual capabilities. Thus, the task is to find, learn and classify those reappearing patterns in different media. Text recognition is one of many sub-fields in pattern recognition which has been receiving a considerable amount of effort and research.

While it is generally accepted that text recognition is solved for machine printed text, recognition-based approaches often do not yield a comparable level of accuracy for handwritten text. This characteristic is particularly pronounced when dealing with historical handwritten documents. In contrast to text recognition, word spotting deals with finding all instances of a query word that exists in a scanned document image, without fully recognising the word text. Word spotting is used as an efficient alternative to index document images for which a direct classification approach would be infeasible. This also allows for indexing and glancing through extensive document image collections for information, especially historical documents and books.

The general goal of a word spotting system is to map the word image onto a representation that allows ranking the image collection according to their relevance with respect to a query. Both the choice of this representation and how it is derived, play a very vital role in the system efficiency.

Although word spotting systems have been receiving an outstanding amount of attention from researchers in recent years, word spotting on Arabic documents has achieved relatively modest results as compared to other major scripts like Latin and Chinese. Investigating potential refinement to existing word spotting systems on Arabic documents is the motive behind this thesis.

In this thesis, we explore various approaches for adapting the Arabic script for the *Embedded Attributes Framework* proposed by Almazán et al. and further improved by the introduction of *PHOCNet* proposed by Sudholt et al.. The choice of attribute representation, which plays a critical role when performing word spotting in the embedded attributes framework, is one of the areas investigated for possible adaptions. In addition, multiple thoughts on optimising the actual embedding used in the PHOCNet are also investigated.

This thesis is structured as follows. Chapter 2 introduces the fundamental concepts underlying the proposed method. Consequently, a brief overview of the basic concepts and methods in the field of word spotting is given. Another essential overview is the introduction to the Arabic script. We present its characteristics and the challenges that arise from performing word spotting on Arabic documents. Furthermore, the fundamental concepts of Artificial Neural Networks (ANNs) and especially Convolutional Neural Networks (CNNs) are discussed. CNNs have become the state-of-the-art method in word spotting, and they are a crucial tool for the proposed method. Chapter 3 presents the works that lay the groundwork for this thesis. The first part will discuss the attributes framework, which had a tremendous impact on the word spotting field. Moreover, a very influential work using CNNs for word spotting, called PHOCNet, is presented in the second part of this chapter. Chapter 4 discusses the proposed method, which will be evaluated by the experiments presented in Chapter 5. Finally, Chapter 6 reviews the work done in this thesis and present the conclusions drawn from the experimental evaluation as well as discussion and future work.

2 FUNDAMENTALS

2.1 ARABIC WRITING SYSTEM

The Arabic script is the second most widely used alphabetic writing system in the world; originally developed for writing the Arabic language [EB16]. Although it has evolved as a direct descendant of the Aramaic alphabet in the 4th century A. D., its early history is vague. Many scholars believe that the earliest living example of Arabic script is a royal funerary inscription dating from A. D. 328 [EB16].

Quite similar to Arabic numerics, Arabic script is used not only by the Arabic language, but has also been adapted to such diverse languages as Persian, Kurdish, Sindhi, Urdu, and many others [Maj96]. Historically, the Arabic script was used for Turkish[1] and other Turkic languages (Uzbek, Azerbaijani, etc.), Indonesian, Swahili, and Spanish [Maj96].

2.1.1 *Characteristics*

The Arabic script is written from right to left in a cursive style in both printed and handwritten documents. The Arabic alphabet is one of the *Abjads*, alphabets that represent consonants only, consisting of 28 unique letters. However, as the script came to be used as the primary script for other languages besides Arabic, it led to the addition of new letters and other symbols to represent phonemes that do not appear in Arabic phonology. Therefore, in Persian, the alphabet consists of 32 letters, 33 in Kurdish, and 39 in Urdu [Omn20].

Additionally, Arabic script uses diacritics that symbolise additional phonetic information including *I'jam* (إِعجَام), consonant pointing, and *Tashkīl* (تَشكِيل), supplementary diacritics. The latter are vowel marks, whereas *I'jam* diacritics are the points added to same-formed letters to help distinguish them based on the number and position of these points [Gac09]. Originally, *I'jam* diacritics were introduced to help foreigners and early learners of the language, but later emerged to be considered as part of the

1 Following the 1928 Reforms, the use of the Arabic alphabet was abandoned and a new Latin-based alphabet was developed [Lew99].

letter itself [Gac09]. For example, letters like ب, ت, and ث share the same base-form ـب. Thus, a word like بات can mean باب (door) and بات (to spend the night).

2.1.2 *Handwriting*

Although Arabic characters have no upper- and lowercase variations, they have either two or four different position-dependent shapes (see Table 2.1). Each letter, with the exception of six (which have two shapes only), is connected with its next and previous neighbouring letters in the same word [LG06].

Additionally, a *Ligature* is a shape formed by combining two or more characters. Arabic has several standard ligatures, which are exceptions to the above rules for connecting letters. *e.g.*, combining the letter *laam* ل and *alif* ا would look like لا and not لا which is clearly not the "Final" shape of *alif* in Table 2.1.

Considering that *I'jam* points are part of the letter itself, the Arabic alphabet distinguishes 15 out of 28 letters based on these diacritical points [Gac09]. On the other hand, *Tashkīl* are normally omitted from handwriting except for *hamza*, *shadda*, and *madda* showed in Figure 2.1.

(a) madda (b) shadda (c) hamza

Figure 2.1: Handwritten Arabic words containing the obligatory diacritics. Images are extracted from IfN/ENIT Dataset [PMM+02].

After examining these different characteristics of the Arabic script, we can find a notable contrast to Latin scripts in terms of the quantity of distinct patterns. Attempting to address each unique position-dependent shape as a distinct entity will eventually result in a rather large set of patterns. A superficial comparison between Arabic and Latin scripts shows that Latin has 26×2 possible patterns[2] whereas Arabic could have $22 \times 4 + 6 \times 2$ unique patterns[3]. Additionally, diacritics like *shadda*, for example, can be placed on most letters thereby forming a slightly modified pattern. Thus, the Arabic script has a very large set of patterns making the task harder for automatic systems to recognise and distinguish this wide range of patterns.

2 Upper- and lowercase have different shapes in Latin.
3 Considering the position-dependent shapes characteristic of Arabic letters.

Name	Isolated	Initial	Medial	Final
alif	ا	-	-	ـا
baa	ب	بـ	ـبـ	ـب
jeem	ج	جـ	ـجـ	ـج
dal	د	-	-	ـد
haa	ه	هـ	ـهـ	ـه

Table 2.1: Example for position-dependent shapes of some Arabic letters.

2.2 WORD SPOTTING

Historical books and documents have a great amount of information relevant to all fields, primarily science and history. Still, such documents are rare and have a poor accessibility. Digitization allows preserving these types of documents from degrading and protects them from frequent usage. Digitized documents can then be indexed and searched with help of pattern recognition methods.

Although traditional OCR[4] approaches excel in recognition tasks within modern printed documents, they do not yield satisfactory results when handling highly degraded historical documents [GSGN17]. Nonetheless, these systems are confronted with challenging barriers when dealing with historical documents. Such barriers extend from the difficulties in segmenting characters or words, the variability of the handwriting, and the degeneration of the original documents [GSGN17].

On one hand, *recognition-based* methods rely on the full recognition of documents, either at a character or word level, to obtain its correct transcription. On the other hand, word spotting methods *i.e. recognition-free* are tasked to retrieve all relevant elements from a collection of document images in a retrieval list without the need for correct character recognition, but by directly characterising image features at a character or word level [GSGN17].

Word spotting methods may be categorized based on multiple factors. Based on the query type we can distinguish Query-by-Example (QbE) from Query-by-String (QbS) methods. In the QbE scenario, the user selects an image of the word to be searched in the document collection, whereas if the query can be provided as an arbitrary text

4 Optical Character Recognition

string, a method is able to perform Query-by-String. Finally, word spotting methods that are directly applied to whole document pages are considered *segmentation-free* methods, whereas in *segmentation-based* methods, a previous segmentation of the word images from the page has to be applied during preprocessing [GSGN17].

In recent years, the topic of word spotting has been the subject of extensive research. As a result, various methods have emerged that usually rely on Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), Artificial Neural Networks (ANNs), or they might follow a hybrid approach by combining different classifiers such as Support Vector Machines (SVMs) with HMMs or HMMs with ANNs [GSGN17].

2.3 ARTIFICIAL NEURAL NETWORKS

Artificial Neural Networks (ANNs) play a fundamental role in pattern recognition and computer vision in general. Word spotting task has been greatly benefiting from ANNs especially in Zhong's SpottingNet [ZPJ+16] and Sudholt's PHOCNet [SF16], both have obtained state-of-the-art results. PHOCNet will be discussed in detail in section 3.2. The following subsection gives an overview of neural networks as these machine learning models also have a significant part in the methodology used later in this thesis.

ANNs have been originally inspired by the way our brains process information using biological neural systems that rely on networks of neurons. However, its development has since diverged and has become a matter of engineering and achieving good results in Machine Learning tasks [KL20]. The general idea behind an ANN is that the weights w (the analogon to the biological synaptic strength) control the strength of influence of one neuron on another, and these weights are then parameters to be adapted in the learning phase (described in detail in subsection 2.3.2). When activating a sequence of neurons it ends by activating a single end-unit that indicates which pattern or class has been recognised.

2.3.1 *Feedforward Neural Networks*

One of the early influential works on neural networks was the Perceptron model introduced by Rosenblatt [Ros58]. The Perceptron classifies input samples $x \in \mathbb{R}^D$ into either of two classes $y \in \{-1, 1\}$. A weighted linear combination of the vector elements is forwarded to a threshold activation function which effectively extracts the sign of

the linear combination. This is the class y predicted by the Perceptron. The output of the Perceptron is therefore given by

$$y(x) = \begin{cases} 1 & \text{when } \mathrm{w}^{\mathrm{T}} x + w_0 > 0, \\ -1 & \text{otherwise} \end{cases} \tag{2.1}$$

where $\mathbf{w}$ is the vector of weights for the input and w_0 is a bias.

The main drawback of a Perceptron is its inability to handle non-linearly separable data [Bis95]. However, Minsky and Papert [MP72] argued this inability could be resolved by stacking multiple layers of Perceptrons. This approach gave rise to a neural network model which is known as Feedforward Neural Networks, or Multilayer Perceptrons (MLP). The term feedforward network describes a multilayer network where the output of any layer i does only depend on the output of the $(i-1)$-th layer. Furthermore, in feedforward networks there are no *feedback* connections in which outputs of the model are fed back into itself. However, when networks include feedback connections, they are called Recurrent Neural Networks [GBC16].

The goal of these networks is to approximate some function f^*. For example, a classifier $y = f^*(x)$ maps an input x to a category y. A feedforward network defines a mapping $\hat{y} = f(x;\theta)$ and the parameters θ may be optimised such that they approximate the function [GBC16].

An MLP contains multiple Perceptrons in sequentially stacked layers that have connections or edges between them. When each neuron in a layer is connected to all neurons in the preceding layer, we call the layer a fully connected layer [KL20]. All these connections have weights associated with them and form a directed acyclic graph that represents how the functions are composed [Sch15]. For example, in Figure 2.2 we have $L-1$ functions $f^1, f^2, ..., f^L$ all connected in a chain, to form

$$\hat{y} = f^L(f^{L-1}(...(f^1(x))...)). \tag{2.2}$$

As each layer s in an MLP is a Perceptron, the layer-wise output is computed as such: first the weighted sum of the layer's input vector is calculated (for simplicity, bias w_0^s is encoded into the weights matrix W^s)

$$i^s(x, W^s) = W^s x. \tag{2.3}$$

The result of the weighted sum of inputs is then forwarded to a function called the *activation function*

$$f^s(x, W^s) = \Theta\left(i^s(x, W^s)\right). \tag{2.4}$$

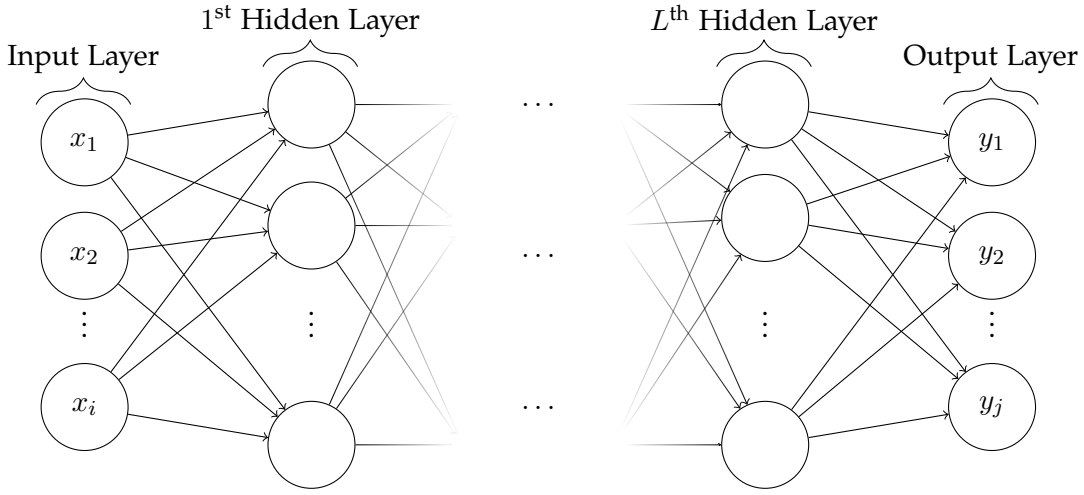


Figure 2.2: Visualization graph of a Multilayer Perceptron. Each layer except for the input layer is a fully connected layer.

The simplest activation function is referred to as linear activation, where no transformation is applied at all. Non-linear activation functions, like the sigmoid function, were introduced to address this problem. Hence, non-linear functions are differentiable and therefore more effective while optimising the network. This is an important aspect for optimising MLPs which will be discussed in subsection 2.3.2.

In contrast to the activation function used in the Perceptron, MLPs traditionally made use of the sigmoid activation function which can be defined as

$$\Theta_\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.5}$$

A different activation function was proposed in [GBB11] called the Rectified Linear Unit (ReLU). ReLU is a unit using the rectified activation function $g(x)$ that simply outputs the input directly if it is positive, otherwise, it will output zero. The ReLU is defined as

$$\Theta_{ReLU}(x) = \max\{0, x\}. \tag{2.6}$$

Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimise with gradient-based methods [GBC16]. With these activation functions, neural networks with a large number of layers can be trained efficiently as they eliminate the problem of vanishing gradients. Additionally, computations are cheaper when using ReLU since we abandoned the exponential

function. Furthermore, negative input values can output true zero values (compared to sigmoid where there are values very close to zero, but not a true zero) allowing the activation of hidden layers to contain one or more true zero values. This is called a *sparse representation* and is a desirable property in representational learning as it can accelerate learning [GBC16].

2.3.2 *Multilayer Perceptrons Optimisation*

During network optimisation, we optimise $f(x;\theta,W)$ to match $f^*(x)$. This optimisation process is referred to as network training or simply training. The training data provides examples of $f^*(x)$ evaluated at different training points. Each example x is accompanied by a label $y \approx f^*(x)$. The output layer then must produce a value that is as close to y as possible *i.e.* as accurate as possible. The behaviour of the hidden layers is to help produce the desired output by activating different paths in the network resulting in a better output. This is done by minimising the difference between the output obtained for a specific input and the desired output with respect to the weights of the MLP [GBC16]. The network updates the weights and biases iteratively with a procedure known as *gradient descent*.

The loss function l lets us quantify the quality of any particular set of weights W. This function computes a numerical value that represents a measure of deviation of the network's prediction $\hat{y} = f(x;\theta,W)$ from y. Hence, our goal is to find the optimal W that minimises the loss function [KL20]. One of the simplest functions is the Euclidean loss

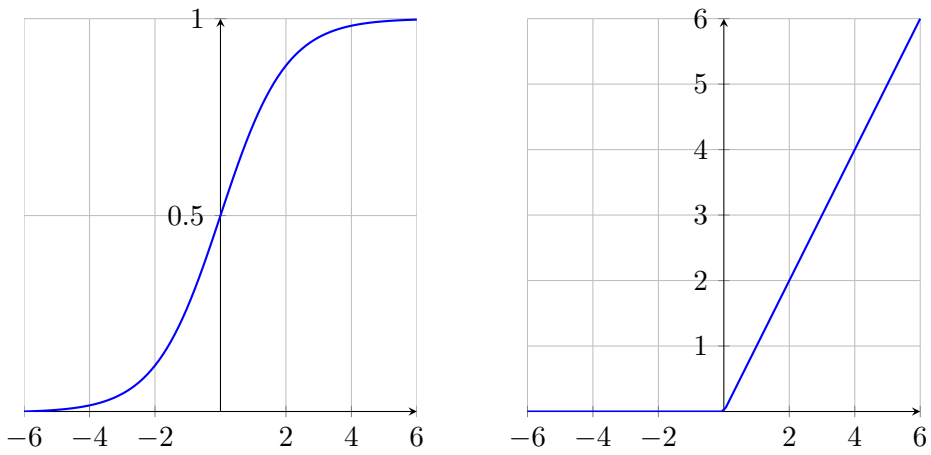
$$l_{euc}(\hat{y},y) = \sum_{i=1}^{n} \frac{1}{2}(\hat{y}_i - y_i)^2. \tag{2.7}$$

Gradient descent uses a loss function to iteratively find a local minimum for l as follows: the weight $w^s_{i,j}$ in layer s is updated by adding a fraction of the negative gradient of the loss with respect to $w^s_{i,j}$

$$w^s_{i,j} \leftarrow w^s_{i,j} - \eta \frac{\delta l}{\delta w^s_{i,j}}. \tag{2.8}$$

η, representing what is called the *learning rate*, controls how much to change the model in response to the estimated error each time the model weights are updated. If chosen too large, the optimization might oscillate or even diverge. On the other hand, a small learning rate might require a large amount of iterations for the algorithm to converge to a local minimum [GBC16].

The above formulation suffices for optimising a network's single layer, but when dealing with the remaining hidden layers a more complex approach called *Backpropa-*



(a) Sigmoid function

(b) Rectified linear function

Figure 2.3: Example of known activation function used in different types of ANN.

gation is used [KL20]. The function f that resembles a MLP consists of compositions of functions (Eq. 2.2), each of which is modelled by one layer of the network. If we now want to determine the gradients of all layers, we have to derive the partial derivative considering the *chain rule*.

A weight has an influence on the results along different paths through the following layers. This implies that for computing the gradient of any weight $w_{i,j}^{s}$, we first need to find the gradient for the previous layer. Hence, for updating all the weights in the network the gradients are computed starting at the last layer and ending at the first layer. The process of updating the weights starts by the output layer computing the gradient of its weights directly from the loss function. The output layer also computes the gradient with respect to its input and sends this gradient "back" to its preceding layer. The preceding layer is handed the gradient for its output. All hidden layers then compute their gradients the same way. This process is known as the *Backpropagation* algorithm. *Backpropagation* is proven to be an efficient algorithm for computing gradients on neural networks using the chain rule [Bis95].

In the preceding subsection, we talked about how using linear activation functions is not a feasible approach when optimising a network with gradient descent. The gradient of a linear step function is 0 everywhere except for when $x = 0$ – for which the gradient is not defined. Meaning that the overall gradient would be the zero vector, and the weights would never receive any updates [Bis95].

2.3.3 *Convolutional Neural Networks*

Intelligent systems that deal with images expect input with high-dimensional data. When neural networks were introduced to the field of image recognition, they were faced with a massive number of parameters making the optimisation task a highly computationally intensive problem. Convolutional Neural Networks (CNNs) represent a type of feedforward networks specialised for image processing. CNNs offer a significant reduction of the number of parameters in a network while preserving much of the model quality. These specialised neural networks were proposed initially by Fukushima [Fuk80] under the name of *Neocognitron* but did not receive significant attention until made famous by LeCun et al. [LBD+90] as CNNs.

While CNNs and MLPs share many similarities, they have several differences that make CNNs specialised in detecting pattern in images. In comparison to MLPs, CNNs use layers called the convolutional layers, which consist of small filters of trainable weights to slide (more precisely, *convolve*) over the input [GBC16]. These filters are responsible for the core functionality of a CNN, which is to let these filter learn what patterns to detect in the data. This learning process is conducted by facilitating filter weights. These filters can be considered as feature detectors that get activated by particular structures or objects in the image, *e.g.*, edges or strokes. However, CNNs can also – and usually do – have non-convolutional layers side by side with convolutional ones.

Furthermore, Convolutional layers, unlike regular MLP layers, are made up of n different filters (also referred to as kernels) and a bias b for each filter. In particular, the convolutional layers of a CNN have neurons arranged in 3 dimensions: width w, height h and channel c (also called depth) [KL20]. During the forward pass, the input image $I_{h,w,c}$ is discretely convolved with the different filters to produce n distinct outputs which in this context are known as *feature maps*. Each convolution is only applied in a small spatial region but spans across all channels of the input [GBC16].

For example, the input image might have thousands or millions of pixels, but we can detect small, meaningful features with filters that occupy only tens or hundreds of pixels. Hence, fewer parameters have to be trained and stored [GBC16]. This reduction benefits not only the optimisation process but also reduces the memory requirements of the model.

A typical layer of a convolutional network consists of three stages. In the first stage, the layer performs several convolutions in parallel to produce a set of linear activations. In the second stage, each linear activation is ran through a non-linear activation function. Traditionally, CNNs use the Rectified Linear Unit (ReLU) mentioned previously (cf. Eq. 2.6). In the third stage, a *pooling function* is used to further

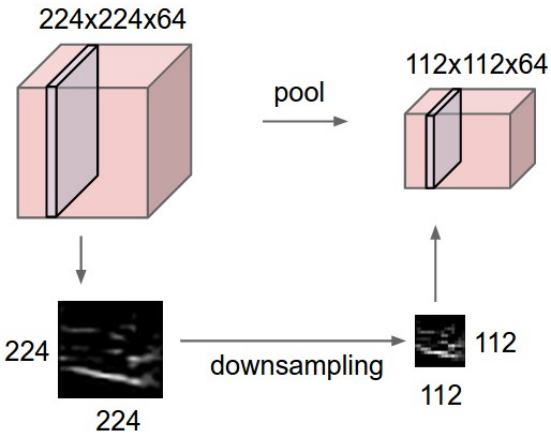


Figure 2.4: Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. Figure is taken from Almazán et al. work [AGFV14].

modify the output of the layer [GBC16]. A pooling function replaces the feature maps at a particular location with an aggregation of the nearby outputs. For example, the *max-pooling* (Zhou and Chellappa [ZC88]) operation extracts the maximum output within a rectangular neighbourhood. Whereas in average pooling, the output feature map values are computed from averaging the values in the respective pooling regions. Figure 2.4 visualises the pooling operation.

As we discussed in the previous section, optimising the weights requires selecting a suitable loss function l. CNNs make use of the categorical cross-entropy loss between the CNN's output and the label vector. CNNs are expected to work with images and predict a single class to each image. In order to do so, a CNN computes the *softmax*

$$\hat{y}_i(o) = \frac{e^{o_i}}{\sum_{j=1}^{N_c} e^{o_j}} \tag{2.9}$$

for each element o_i of the last layer's output o in order to obtain the posterior probability $\hat{y}_i$ for the i-th of N_c classes. The predicted class is the one with the highest probability $\hat{y}$. The gradient of the computed loss is then backpropagated through the neural network.

However, convolutional layers can also accept the output of other convolutional layers, meaning, a convolutional layer can be applied on previous feature maps extracted from an image. Hence, CNNs can learn more complex and hierarchical features. Natural images contain objects composed of object parts which in turn can be composed of edges or strokes. This fact explains why convolutional layers close

to the input image typically learn to detect colour blobs and edges, whereas deeper layers have filters that are activated by objects parts or entire objects [GBC16].

In conclusion, CNNs can learn hierarchical representations of an input. These hierarchical structures are typically given for images of handwritten text. Word images can be decomposed into characters which in turn can be decomposed into strokes. This explains the state-of-the-art performance of the PHOCNet, a CNN developed by Sudholt et al. specifically for the application to word images [SF16].

3 RELATED WORK

3.1 EMBEDDED ATTRIBUTES FRAMEWORK

Initially proposed by Almazán et al. [AGFV14], the embedded attributes framework has inspired the work on word spotting considerably [GSGN17]. The framework has inspired many recent works, further extending or adapting the base model [KDJ16, WB16, SF16]. This framework introduces a learning-based method which can deal with both types of query formulation (QbE and QbS). It can learn projections from an image space and a text-string space to a common latent subspace using Kernel Common Subspace Regression (CSR).

As seen in Figure 3.1, the primary concept of the attributes framework is to encode both word strings and word images, as common fixed-length representation using attributes. Attributes are entities that are then shared between different classes. Hence, two mappings are needed to encode inputs into the common attribute space: textual and visual. The *textual model* serves as a mapping for the word strings, *i.e.* word classes, whereas a *visual model* maps word images into the attribute space. Fixed-length representations present a clear advantage over sequential representations, as the fixed-size feature vectors can be compared using standard distances such as the Euclidean distance. This way, image matching is reduced to a much faster nearest neighbour search problem [GSGN17]. While Almazán et al. require the mapping for the word classes to be directly obtainable from the respective string representation, the mapping from word images to attribute space is advised to be obtained by a trainable model.

By learning character attributes independently, training data is better used (since the same training words are used to train several attributes). Additionally, by making the visual model learn attributes instead of class labels, the framework allows for predicting attribute representations from classes which were not known during training [AGFV14]. However, this characteristic benefits QbS word spotting in particular: In QbS, the textual model is responsible for predicting the query representations. However, since it can compute this representation directly from a string unknown at training time, the queries are not constrained to come from a predefined or closed lexicon.

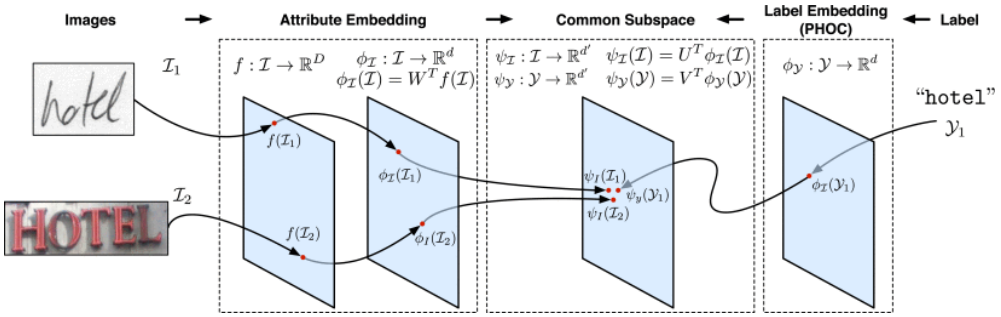


Figure 3.1: In this common subspace representations are comparable and labels and images that are relevant to each other are brought together [AGFV14].

The selection of the attribute representation for the attributes, computed from word strings or images, plays a vital part in the embedded attributes framework. Almazán et al. propose an embedding called the Pyramidal Histogram of Characters (PHOC). This PHOC encodes, pyramidally, if a particular character appears in a specific spatial region of the string [AGFV14].

To construct the representation, an alphabet of characters or *unigrams* needs to be defined first. As unigrams, Almazán et al. initially used all lower-case characters from the Latin alphabet plus the ten digits. They assumed that a word image should be considered relevant for a given query if the transcriptions match regardless of the letter case. Next, an attribute is created for each unigram, indicating its presence or absence in a particular split of the string. However, splits of a string are grouped into levels. Each level represents further splits of the prior level split. Thus, forming a pyramidal structure. At the first level of the PHOC, all unigrams in the string are considered, and the respective attribute is set to 1 if the corresponding unigram appears at least once in the word. In the following levels, the string is split into *regions* of same sizes. For each unigram and region, it is then evaluated whether the region contains at least one of the respective unigrams.

To determine whether a specific unigram appears in the given string (within a specific region), certain criteria are set. Foremost, all unigrams in a string are treated with equal width of 1. In the example in Figure 3.2, the total width of the word would thus be 5. Almazán et al. then define a unigram to be present in a given region if it overlaps at least 50% with the particular region. For example, the character *a* in Figure 3.2 has an overlap of 50% with both left and right regions in the second level. Hence, it is marked as present for both regions. In the ensuing levels, the string is split into increasingly smaller regions. This pattern continues for as many levels as are desired. Almazán et al. propose to not use the first level in the PHOC as the resulting

vector for this level can not distinguish between anagrams, e.g., *asleep* and *please*. In their PHOC definition, they use the levels 2, 3, 4 and 5.

After defining a global feature representation, the goal is to predict the PHOC representation corresponding to the query word image. Almazán et al. make use of an ensemble of SVMs as the visual model, which they term AttributeSVM. The AttributeSVM requires that the image is encoded into some form of holistic feature representation. For this, Almazán et al. choose the Fisher Vector on top of enriched SIFT descriptors which are extracted in a grid. However, to predict the PHOC representation corresponding to the transcription of the word image, an SVM for predicting each attribute is created. Each SVM learns its own model and does not benefit from the shared parameters. The combination of all SVMs then predicts the entire PHOC.

Apart from that, there are other approaches based on this framework reported in the literature. Although these approaches are not related to word spotting, they offer a visual model facilitating neural networks [SGC15, RASC14]. However, since we are interested in the application of this framework on word spotting task, this study makes use of a CNN-based visual model called the PHOCNet.

3.2 PHOCNet

PHOCNet is a deep Convolutional Neural Network developed by Sudholt et al. [SF16]. PHOCNet has achieved state-of-the-art result in both QbS and QbE word spotting while maintaining short training and test times [GSGN17].

However, this CNN can predict the attribute representation for all word images in the desired corpus prior to running retrieval. QbE word spotting can then be performed by predicting the attribute representation for a given query word image and ranking

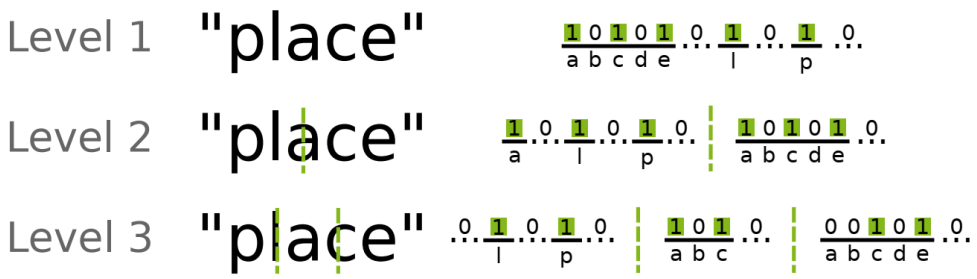


Figure 3.2: Visualization of a PHOC vector for the word string "place" with levels 1,2 and 3. The figure is courtesy of Sudholt work [SF16].

all images in the database according to the distance to the query representation. As the attribute representation is a function of word strings, QbS word spotting can be performed by computing the attribute representation for the given query string and ranking all attribute representations of word images in the corpus as was done for QbE. Figure 3.3 visualises the PHOCNet method.

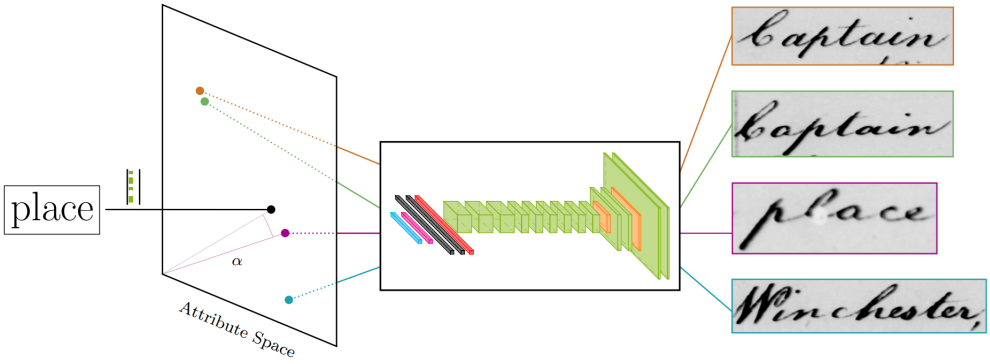


Figure 3.3: For Query-by-String, the attribute representation is extracted directly from the string (left side). For word images, a CNN predicts the attribute representation. This way, annotations and word images can be projected into a joint attribute space. Similarity in the attribute space can be determined by applying the cosine distance. Not shown in the figure is Query-by-Example, which is done by simply ranking the attribute representation obtained from the word images in a nearest neighbour approach. Figure and caption were extracted from [SF18].

Inspired by the successful VGG16 architecture [SZ15], Sudholt et al. proposed a CNN architecture to directly embed image features to PHOC attributes. Usually, CNNs are used to predict one out of k classes for a given image, *i.e.* multi-class classification. However, the last layer in the network represents the probability of each class being recognised in the input. By computing the softmax (c.f Eq. 2.9) of each element in the last layer's output, the predicted class is determined by the highest probability. In contrast to the multi-class, PHOCNet goal is to predict attribute representations, *i.e.* multi-label. Here, the softmax output becomes infeasible since *at most one* element in the output vector can become 1. To overcome this, Sudholt et al. proposed using sigmoid activation in the final layer of the PHOCNet. Furthermore, Sudholt et al. investigated in [SF17], how different loss functions affect the network performance. They found that both Cosine Loss, and Binary Cross Entropy (BCE) Loss are well fit to be used with binary vectors like PHOC. PHOCNet was reported to preform well using

either, Cosine or BCE loss, since no choice was superior to the other over different configuration with different datasets.

Another aspect to consider regarding segmentation-based word spotting is that the input images witness broad diversity in their size. Besides, traditional strategies to generate fixed-size images, such as resizing and cropping, possibly produce semantic distortion. While in multi-class classification problems, the outputs for different crops can simply be averaged. A similar approach is not feasible in the case of PHOCNets since PHOCNets contain a certain level of positional encoding of the attributes [SF16]. In order to fix this, Sudholt et al. employ a Spatial Pyramid Pooling (SPP) layer after the last convolutional layer. This way, the first fully connected layer following the convolutional part is always presented with a fixed size image representation, independent of the input image size [SF18].

As we described in the previous chapter, the first step of defining a PHOC structure is choosing the set of unigrams which the PHOC will be built upon. This step plays a very vital role in building an efficient word spotting system. This role takes its importance from the fact that the same unigram appearing in multiple word strings should have the same pattern or shape in the corresponding word images. Hence, optimizing the selection of unigrams will facilitate and improve efficiency in the task of learning a unigram shape through a visual model.

4

METHOD

In this work, various attribute representations for the embedded attributes framework are investigated. We discuss how these different representations perform for the task of word spotting on Arabic scripts. A comparison of efficiency is also reported later in the experiments section. The PHOCNet is used to carry out theses comparisons by evaluating a model created using each representation. The choice of PHOCNet is because of its significant impact on word spotting and its state-of-the-art results [GSGN17].

The current investigation involves integrating and experimenting with multiple sets of unigrams for the PHOC representation. Then, other representation approaches are introduced and implemented with the goal to raise model efficiency and performance. However, all of the previously mentioned approaches are constructed with careful consideration of the particularities of the Arabic script (c.f. subsection 2.1.2).

The first two unigrams sets are chosen to establish a baseline for the rest. Then, the Sub-Character set, supplied by I. Ahmad the author of [AFM14], is tested. Also, a new unigrams set is introduced. Furthermore, we implement and experiment two approaches to reduce the PHOC size regardless of which unigrams set is selected.

The following section presents each set of unigrams in detail and gives related background information. Section 4.2 deals then with the proposed representation approaches.

4.1 UNIGRAMS SETS

Previously in the fundamentals section, we explored the important characteristics of the Arabic script and brought challenging aspects into the light. However, we noted how the Arabic characters are context dependant and how this context affects character pattern. As a result of which, Arabic scripts have up to four position-dependent shapes for each letter. For simplicity, we define the *extended Arabic alphabet*. This alphabet considers each position-dependent shape of Arabic letters an individual character. Hence, the *extended alphabet* contains 100 characters while the regular alphabet has only 28. Table A.2 shows the full extended alphabet.

For each set of unigrams, a lookup table is created to map each character to a responding unigram. This lookup is then used to transform the labels of word images. A Label represents the transcription of the word image stating what characters are present in the image.

4.1.1 *PHOCNet unigrams (Baseline)*

To create a baseline for further sets, we reconstruct the unigrams set used in the original work of Sudholt et al. in [SF16]. They use a reduced character set which is generated in the following way: First, all character shapes are mapped to their representative Arabic characters regardless of their position within a word. Shadda diacritic is ignored in the sense that, characters with a Shadda diacritic are handled as characters without it. Additionally, ligatures are simply mapped to their constructing characters without the shape contexts.

4.1.2 *Direct unigrams*

This set represents a very simple set created using the same characters used to build the dataset labels. Note that, while raw shape contexts are present in the labels and represented as unique letters, ligatures are considered individual unique characters without stating its relation to the constructing characters. Additionally, since Shadda diacritic could be placed on almost all characters, characters with Shadda are handled as different from the originals. This makes this set the largest set of unigrams considered in this study.

4.1.3 *Sub-Character unigrams*

The Sub-Character unigrams set is derived from the work of Ahmad et al. which proposed and improved [ARFM13, AFM14] a different approach for handling the Arabic script. This approach is originally developed for building HMM models with a considerably reduced number of HMMs while still capturing the variations in shape patterns. However, Ahmad's approach suggests modelling Arabic characters at the sub-character level, *i.e.* strokes that build a character. Modelling at this level allows the sharing of common patterns between different position-dependent shapes of an Arabic character as well as between different characters. Figure 4.1 illustrates the idea with some example characters. Moreover, just like the previous set, ligatures here are handled as stand-alone patterns.

In addition to the above patterns, two special patterns are required. A *connector* pattern which is the small horizontal stroke between two connected letters. Whereas a *space* pattern is required to represent the gap between letters that do not connect with the following letter. A *space* pattern is also needed to indicate the end of a word.

For this set, a lookup is not created since the labels were pre-transformed and supplied by the author.

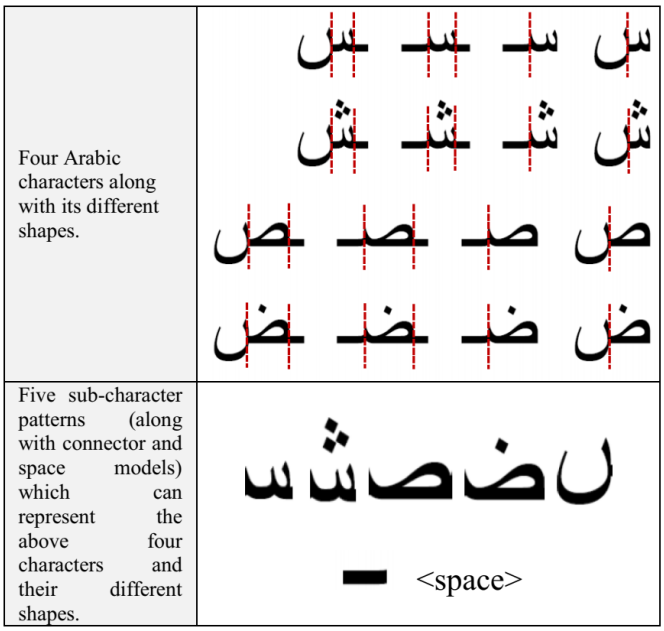


Figure 4.1: An example of sub-character modelling presenting common patterns shared between different Arabic letters. Figure is extracted from [ARFM13].

4.1.4 *Ground Form Representation unigrams*

The approach we use intuitively is to split each character into three groups of patterns and focus on the *ground form* pattern of the character. Hence, this approach is dubbed the Ground Form Representation (GFR). This discrimination into separate patterns takes a very similar approach used by Kaplony for reading Arabic words [Kapo8]. He uses three layers, each representing a group of different phonetic information. The first layer is the *rasm*, undotted characters representing the actual most basic form. To this *rasm*, we add *I'jam* dots, to have a second layer with unambiguous characters. Lastly, we add *Tashkīl* strokes to make vocalized characters. Table 4.1 visualises these layers.

Name	Rasm	I'jam	Tashkīl
Letter *baa*	ب	ب	بَ
kitab (a book)	كتاب	كتاب	كِتابٌ

Table 4.1: Visualisation of the three layers used to separate pattern groups.

Early written Arabic used only *rasm* letters and did not have *I'jam* dots. Later on, they are added to remove ambiguity in words without context. This, however, explains the current presence of shared shapes between different Arabic letters; namely a shared *rasm*. Moreover, since *I'jam* dots are also shared across the alphabet, it is only fair to represent these shapes as individual entities. These shapes can then be combined, theoretically, with any *rasm* to form a full character. Table 4.2 shows how a single *rasm* can supersede three characters with the help of three add-ons.

In this set, characters glyphs are divided into two pattern groups. On one hand, there is a *ground form* pattern which represents the *rasm*. On the other hand, we defined the *add-on* pattern, which represents any additional pattern added on top of the ground form. Figure 4.2 illustrates how ground form and add-on patterns are combined to form a full character. Furthermore, considering that *Tashkīl* diacritics are very rare in handwritings, the last layer is not represented within the resulting unigrams set. Yet a few diacritics are obligatory and could be placed on various characters (c.f. Figure 2.1). Therefore, *hamza*, *shadda*, and *madda* diacritics are handled as add-on patterns.

The resulting unigrams set consists of all unique patterns from both groups. However, although ligatures are not handled as separate patterns, they are mapped to the unigrams used to construct that ligature; preserving the shape contexts. A full list of the resulting unigrams is provided in Appendix Table A.3.

4.2 ATTRIBUTES REPRESENTATIONS

After we implemented the previous sets and integrated them in the PHOC representation. We next investigate further modifications on the original representation.

Name	Isolated	Initial	Medial	Final
baa	ب	بـ	ـبـ	ـب
taa	ت	تـ	ـتـ	ـت
thaa	ث	ثـ	ـثـ	ـث
shared	ـب	ـ	ـ	ـب

Table 4.2: This table contain three different Arabic letters and their shared patterns.

4.2.1 *PHOC-d*

As discussed in section 3.1, PHOC represents a pyramidal view for characters within a word. These characters are grouped into splits in each level. However, the process of creating splits is conducted based on spatial region overlaps. Which is why the calculation of the overlap steers the virtual segmentation of a word.

The original criteria to calculate this overlap suggests that each unigram gets a width of 1. After that, a word with n characters gets the width of n. Furthermore, at each level, the word width is split into same-sized horizontal regions.

However, in GFR, the approach suggests dividing a character into two groups of patterns. These patterns are separated vertically and not horizontally like the rest of the sets. This in turns transforms each character with these patterns into two or more unigrams. Hence, transforming into the GFR set may mislead the width of the word and cause wrong spatial information of the regions of a PHOC.

In this work, a modification of the PHOC representation is suggested, to avoid the spatial inaccuracy. This modification is considered for vertical diacritics that should not affect the width of the word. Thus it is called Pyramidal Histogram of Characters with suffixing Diacritics (PHOC-*d*). Figure 4.3 illustrates how the suggested representation works.

In this approach, diacritics would not take a space of a unigram, rather be assigned the same spatial region of its carrier unigram. This is done by giving all diacritics the width of 0 while calculating the width of word n. Then, in the calculation of overlap factor, diacritical unigrams are handled exclusively– returning the overlap value of the previous unigram.

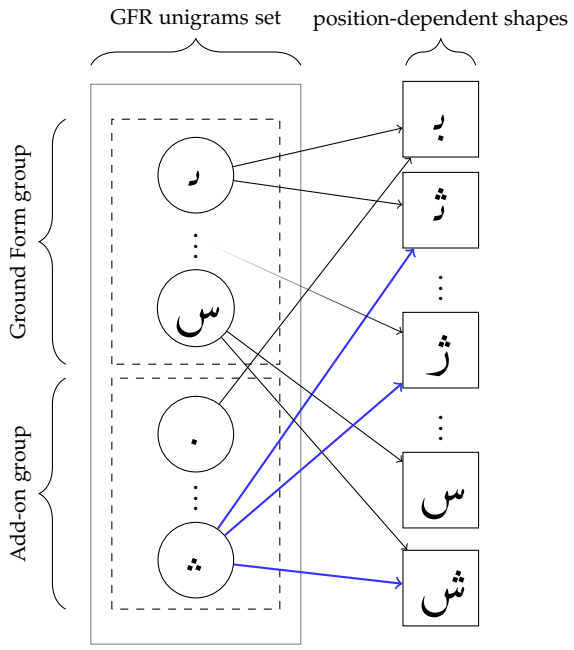


Figure 4.2: This figure illustrates how a ground form unigram can be combined with different add-on unigrams to form other characters. An add-on unigram can also be used in different combination to form new characters.

This modification has the disadvantage of requiring a predefined group of diacritical unigrams to be provided. Additionally, no vertical spatial information is stored in the architecture. This can be bypassed, for example, by dividing each PHOC unit into multiple vertical regions.

4.2.2 *Dynamic PHOC*

Dynamic PHOC is the last approach proposed in this study. This approach suggests reducing the number of trainable parameters in the PHOCNet by reducing the corresponding PHOC size.

The motivation behind this approach comes from the observation, that some attributes have a very low occurrence rate in the lower levels of a PHOC – sometimes none at all. This rate is observed locally on the PHOCs Vector of the entire training set. Locally here means that these occurrences are collected in each region –in each level– separately and then the rate is calculated relatively to the attribute region. However,

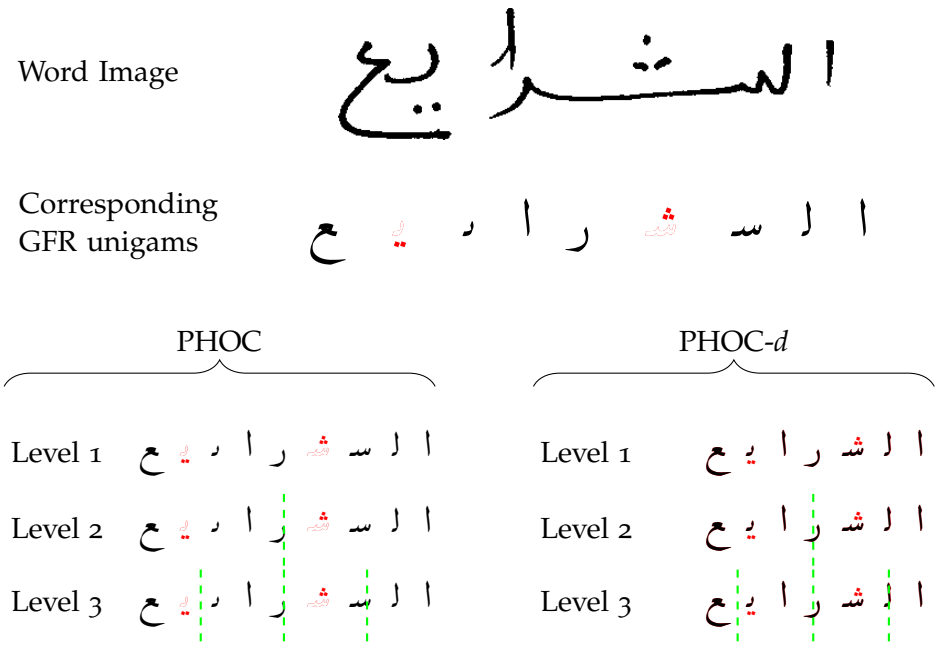


Figure 4.3: Illustration of how the PHOC-*d* (on the right) handles unigrams with diacritics side-by-side with the normal PHOC. Diacritics (in red) are handled as individual unigrams by PHOC while PHOC-*d* handles them combined with the carrier. Note how the width of the word $n = 9$ in PHOC changes to $n = 7$ when using the corresponding PHOC-*d*.

lower occurrence rate indicates that these attributes are hardly learned by the network and their representation remains a burden.

In this representation, a threshold θ is selected. This θ value represents the lower limit for the relative occurrence rate of an attribute. So for an attribute α in a region S, we define the relative occurrence rate

$$r_\alpha^S = \frac{\sum_{i=0}^{n} x_{i\alpha}}{\sum_{i=S_0}^{S_n} \sum_{j=0}^{h} x_{ij}}, \tag{4.1}$$

where h represents the PHOC size while S_n and S_0 represent the start and end indices of the region S respectively.

Based on the above, the process to extract dynamic PHOC is simple. We first generate the PHOCs Vector of the entire training set. Then, all regions S have to be iterated over to exclude all attributes α in this region, which has a $r_\alpha^S \leq \theta$. This process offers to reduce a huge amount of parameters and thus minimizing overhead during network training.

However, there is a clear disadvantage of this approach: removing attributes will result in inhomogeneous regions in the PHOC while also causing a slight loss of information. Moreover, since the attributes elimination process is based solely on the training data, attributes that were not present during network training will not be predicted correctly.

5 EXPERIMENTAL EVALUATION

This chapter covers the experiments conducted to evaluate the method described in the previous chapter. All experiments are performed for the IfN/ENIT dataset presented in 5.1. Then the choice of the performance measure applied in this evaluation is described in 5.2. All experiments follow the same evaluation protocol described in 5.3. Finally, in 5.5, the results of the different experiments are analysed and compared with the state-of-the-art results reported in the literature.

5.1 DATASET

Representative datasets are needed for the evaluation and comparison of different approaches. These provide training and test data with the necessary annotations for training models in a supervised manner and for evaluating performances.

The IfN/ENIT Dataset [PMM+02] is the benchmark dataset for word spotting in Arabic in the form of pre-segmented annotated word images. The dataset in version 2.0 patch level le (v2.0p1e) consists of more than 32,000 handwritten word images by more than 1000 writers. The words represent 937 Tunisian town/village names written in Arabic. It holds about 138,000 pieces of Arabic words (PAWs), and about 257,000 characters. The IfN/ENIT is divided into subsets from a to e ranging by difficulty based on deformation intensity. Two new subsets f and s were also added for testing. These subsets witness the most deformation and different writing styles as compared with the other sets. Set f was collected in Tunisia, while set s was collected in the United Arab Emirates (UAE) at the University of Sharjah [MA09]. Thus, set s is regarded as the most challenging set. Table 5.1 shows sample images from the subsets.

IfN/ENIT dataset has been used in various text recognition competitions, and the results are published in the top conferences like ICDAR [MA09] and ICFHR [MA10]. The dataset is also publicly available [1].

Furthermore, the existence of subsets simplifies dividing the dataset into training and testing sets. Hence, we use the most common train test configuration, as reported in the literature, including the various competitions held using this dataset.

1 http://www.ifnenit.com/

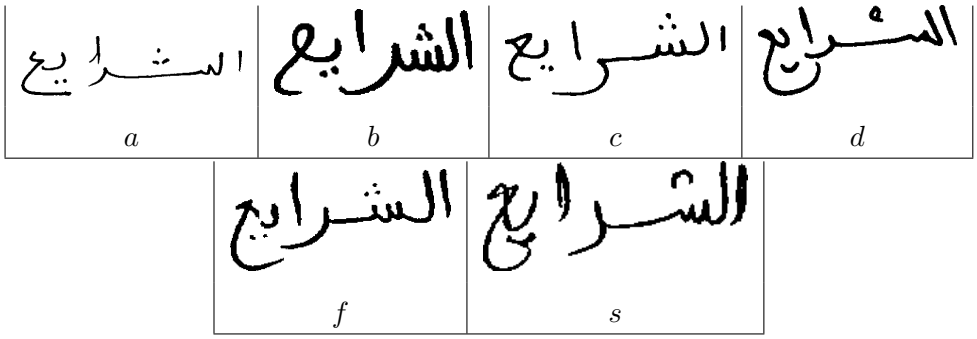


Table 5.1: Comparing training sets to the test subsets f and s. Notice how the f image has some distortion and how the writing style in s is different from the above. Images represent the same word and are selected from the IfN/ENIT dataset.

5.2 PERFORMANCE MEASURES

As word spotting is primarily an information retrieval problem, performance measures for specific word spotting methods are typically obtained from this field of research as well. A retrieval list for Word spotting consists of a number of word images from the dataset, which are relevant with respect to the query. A very well known measure for assessing the *precision*, *recall* as well as sorting of a retrieval list simultaneously is the Average Precision (AP) [GSGN17]. Mathematically, Average Precision for query q can be defined as

$$AP_q = \frac{\sum_{i=1}^{n} p_q(i) \cdot r_q(i)}{t_q} \tag{5.1}$$

where n is the length of the retrieval list, and $p(i)$ denotes the precision, with respect to the first i elements, $r(i)$ indicate if the i-th element is relative to the query and t is the total number of relevant elements in the list.

However, we need to test the retrieval system to not only perform well for a single query but rather a number of different ones. Thus, the use of the mean Average Precision (mAP) which corresponds to the mean over all Average Precisions computed for each query.

5.3 EVALUATION PROTOCOL

Evaluation is based on a commonly used protocol, which is also applied in [AGFV14] and [SF16]. The evaluation process starts by training a network which maps a word image to the respective attribute representation.

Prior to training, we transform the labels of all images in the dataset using the regarded unigrams set. Additionally, since that Arabic script is written from right to left opposed to Latin script, the word images from the IfN/ENIT had to be mirrored along the vertical axis. Then we train the PHOCNet with the corresponding train-test configurations. The training setup is defined in section 5.4.

Once the network was trained, the mAP value on the test subset is calculated. To do so, first, the attribute representation of the query is derived. In QbS, the attribute vector can be directly derived from a string. However, to exclude redundant queries, we only use the unique strings in the test set. For QbE, we use the previously trained PHOCNet to derive the attribute representation from a query image. Word spotting is then performed by ranking all images in the test set with respect to the query, represented by a PHOC vector. Each word image in the test set is used once as a query. The ranking succeeds by sorting the test set based on their distance to the query vector. The Cosine distance is used as a distance metric [SF17].

For each set of unigrams[2], we train and evaluate the network on the following train-test combinations: abc$-d$, abcd$-e$, abcde$-f$, abcde$-s$. Where the letters before the "$-$" represent the training subsets, and the letter after that, is the subset we test on.

5.4 TRAINING SETUP

PHOCNet is trained in an end-to-end fashion given word images as input and their corresponding attribute representation as labels. The chosen attribute representation is the PHOC where the levels 2, 3, 4 and 5 are used; representing 14 spatial regions. The network training for all experiments made use of the same set of configurations. The training is performed using an *Adam* optimiser [KB15], and its parameters are derived from [Sud18]. In the following experiments, we use the Binary Cross Entropy (BCE) loss function for training. All networks are trained with a mini-batch size of 10, momentum of 0.9, weight decay of $5 \cdot 10^{-5}$ and an initial learning rate of 10^{-4}. We let the training run for a maximum of 80000 iterations with the learning rate being divided by ten after 70000 iterations.

2 Except the Sub-Character set where the labels were not supplied for the subsets f and s.

Weight initialisation follows the same approach in [Sud18]. Layer biases are initialised with zeros, and all other initial weights are randomly sampled from a zero-mean uniform distribution. The corresponding variance is set to $\frac{n}{2}$, where n corresponds to the number of parameters in the respective layer. Implementation is accomplished in Python using the PyTorch framework [PGC+17].

5.5 RESULTS AND DISCUSSION

In this section, we present and discuss the results of all the conducted experiments. We also share some observations regarding the experiments. Finally, we compare our results with the findings reported in the literature.

5.5.1 *Baseline*

In this experiment, we adopt the set of unigrams from [SF16], which we set as a baseline for the other sets. We are able to reproduce the results reported in their work for the same configurations. Table 5.2 shows the results for both QbE and QbS on different train-test combinations. Additionally, we evaluate this set of unigrams on the new subsets f and s, which were not reported in Sudholt's work. The performance on set s is not quite good as on other test subsets. This can be traced back to the variation in writing style used in training and test subsets (cf. section 5.1).

The resulting unigrams set has a size of 50, and the corresponding PHOC size is 700. However, this set does not store any shape context for the characters nor distinguish diacritics or ligatures. Thus, this set is the smallest compared with the rest.

Table 5.2: Baseline results for the QbE and QbS experiments in mAP [%]

Unigrams Set	abc−*d*		abcd−*e*		abcde−*f*		abcde−*s*	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
Original [SF16]	96.11	92.14	-	-	-	-	-	-
Baseline	96.30	94.56	94.67	96.87	96.21	97.17	88.90	93.08

5.5.2 *Direct Unigrams*

This approach has no mapping for the characters since it is directly extracted from the unique characters used in the dataset labels. Thus, no actual transformation was applied to the labels before training. The resulting unigrams number is 178 while the PHOC is of size 2492. This set is the largest of the four because complete shape context information remain intact with addition to the various combinations of characters with diacritics, *e.g.*, Shadda.

Using the Direct set, the PHOCNet performance is significantly underdeveloped examined alongside the other unigrams sets. Table 5.3 lists the mAP values for QbE and QbS using the direct unigrams set.

Table 5.3: Results for the QbE and QbS experiments in mAP [%] using a Direct set

Unigrams Set	abc−*d*		abcd−*e*		abcde−*f*		abcde−*s*	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
Direct	90.17	89.52	93.82	96.96	95.40	96.99	87.52	92.26

5.5.3 *Sub-Character Unigrams*

After transforming the labels using the Sub-Character set, we get 97 unique unigrams. The resulting PHOC size is 1358. While the size of this set is noticeably larger than the baseline, one should remark that this set stores both shape context and diacritics.

We observed that the extra two unigrams used in this set – namely the *separator* (sp) and the *connector* (cnn) – were forming two highly redundant attributes. Nonetheless, based on what we previously presented in subsection 4.1.3, we investigate taking these two unigrams out of the set commutatively. We assume that the separator is superfluous since it represents an *empty* space between characters which led to no useful pattern learned to this attribute.

On the other hand, the connector represents a length-varied stroke that is almost absent not only in all ligatures but also in many writing styles. However, the variations without these unigrams[3] achieve slightly better results than the original while also reducing the PHOC size.

3 This approach is referred to with the (no-sp-no-cnn) flag.

Compared to the direct set, the Sub-Character set witnesses a major reduction in the unique unigrams number used to represent the dataset labels. Table 5.4 reports the results for different variations of this set.

Table 5.4: Results for the QbE and QbS experiments in mAP [%] using Sub-Character unigrams

Unigrams Set	Variation	abc−*d*		abcd−*e*	
		QbE	QbS	QbE	QbS
Sub-Character	-	91.66	90.49	94.49	96.70
Sub-Character	no-cnn	91.70	90.22	**94.68**	**96.99**
Sub-Character	no-sp	91.82	90.39	94.25	96.46
Sub-Character	no-sp-no-cnn	**91.91**	**90.96**	94.16	96.62

5.5.4 *GFR Unigrams*

In this experiment, we apply the unigrams set proposed in subsection 4.1.4 and evaluate its efficiency. By employing the GFR approach, 53 unique unigrams are obtained, and the resulting PHOC size is 742. Even though the GFR set is slightly larger than the baseline in term of unigrams count, GFR proved to be a rather compact set since it simultaneously preserves the shape context while using a small number of unigrams. The same argument we used in comparing the previous Sub-Character set with the baseline does also apply here. However, opposed to the two sets that store context information, this set is the smallest set of unigrams among them.

Regarding performance aspects, the results listed in Table 5.5 shows that GFR set outperforms all other unigrams sets presented in this work, with the exception for the abc−*d*. The fallback in performance when evaluating this train-test configuration motivated us to examine the distribution of unigrams in the corresponding training subsets. As a result of which, we deduced that distinguishing characters with a Shadda diacritic could be the reason since the Shadda unigram is under-represented in these subsets. We then took Shadda unigram out of the set, thereby ignoring Shaddas in words. This means that characters containing a Shadda in their labels are now mapped to their analogous characters without a Shadda. This removal[4] led to the PHOCNet performing better for all the train-test configurations.

4 This approach has the (no-shd) flag.

Table 5.5: Results for the QbE and QbS experiments using GFR set

Unigrams Set	abc−*d*		abcd−*e*		abcde−*f*		abcde−*s*	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
GFR	93.30	90.55	96.14	98.36	96.87	97.79	90.42	93.56
GFR (no-shd)	97.66	94.32	96.46	98.35	96.96	98.02	90.12	93.57

5.5.5 *PHOC-d*

While the previous experiments are testing different unigrams set for the same attribute representation, in this experiment, we test a new representation formulated in subsection 4.2.1. We implement a slightly different representation of the PHOC with special handling for diacritics. The experiment is conducted using the GFR set because it is the only set among the four presented that depicts diacritics in individual unigrams. Nonetheless, a predefined list of diacritics is supplied to the representation. A full list of the used GFR diacritics (add-on patterns) is provided in Table A.3.

Furthermore, Table 5.6 shows the results of using PHOC-*d* for both variations of GFR. Although there is just a slight difference between the results here and the original, there is an efficiency advantage when using PHOC-*d* representation. Figure 5.1 shows how the performance in early training iterations is better than the regular PHOC. The performance is measured for the QbE task evaluated on the subset *d*.

Table 5.6: Results for the QbE and QbS experiments in mAP [%]

Representation	abc−*d*		abcd−*e*		abcde−*f*		abcde−*s*	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
PHOC-*d* (GFR)	93.02	90.12	96.24	98.35	96.69	97.78	90.41	93.90
PHOC-*d* (GFR (no-shd))	97.24	94.20	96.34	98.37	96.92	97.85	90.27	93.49

5.5.6 *Dynamic PHOC*

In subsection 4.2.2, we proposed an approach to reduce the PHOC size. This experiment then shows how this reduction has affected network performance and efficiency.

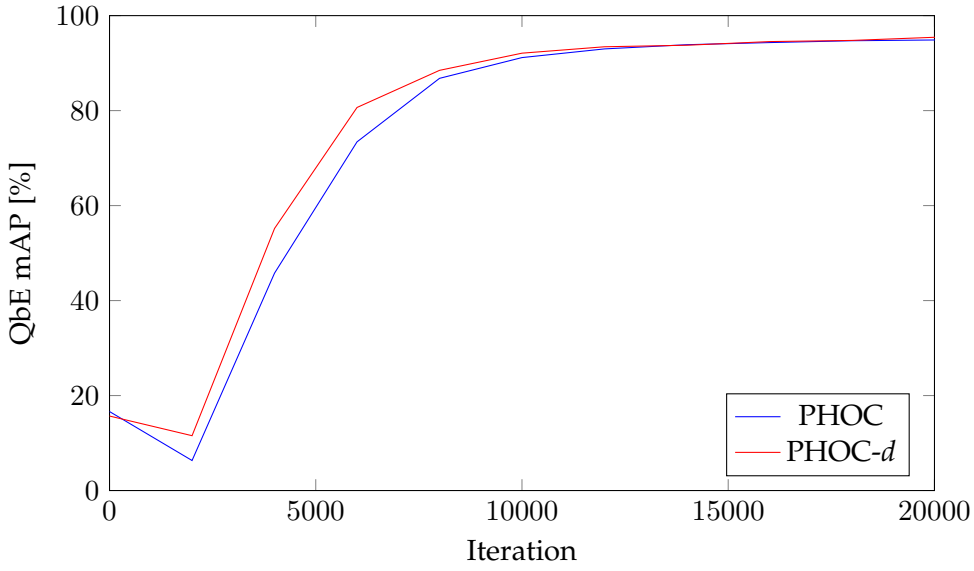


Figure 5.1: The figure compares the mAP between PHOC and PHOC-*d* representations over the training iterations. It shows that the network can learn the PHOC-*d* representation faster while performing relatively the same compared to learning PHOC.

We start by selecting different values for θ and then evaluate the network using the new representation. The Table 5.7 lists the changes in PHOC size corresponding to different θ values using different unigrams sets.

Table 5.7: PHOC sizes for different sets when using different values of θ. Here, the modified versions of Sub-Character and GFR are used: Sub-Character (no-sp-no-cnn) and GFR (no-shd).

Unigrams Set	θ					
	None	0	0.20	0.50	0.80	1
Baseline	700	589	481	410	348	317
Direct	2492	1815	1115	639	419	347
Sub-Character	1330	1069	773	518	381	325
GFR	728	650	539	434	360	334

In addition, Figure 5.2 visualises how a reduction in the represented attributes number can affect both efficiency and performance of the PHOCNet. In particular, we found how applying a $\theta = 0.2$ has raised the performance of the Direct representation while also keeping the performance of the others semi-stable.

Based on the value of θ, the number of reduced attributes differ. Consequently, the number of attributes that need to be learned by the network is reduced and thus, the learning is faster in the newly trained networks. However, for high values for θ, the network did not yield comparable results.

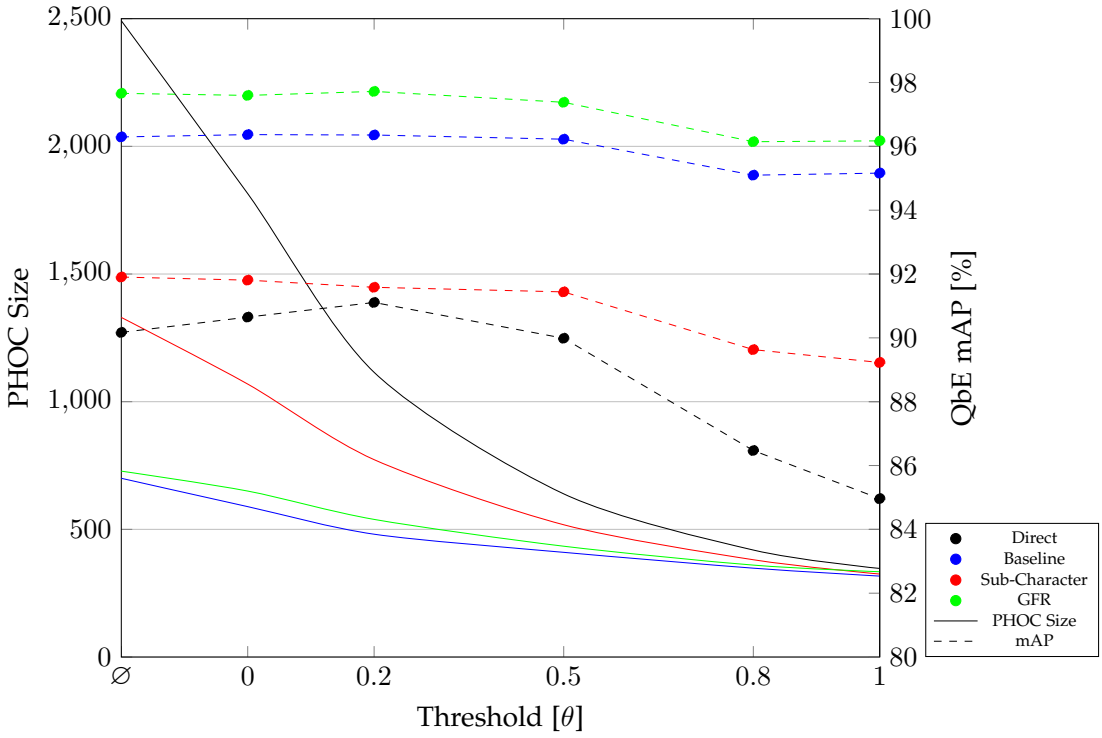
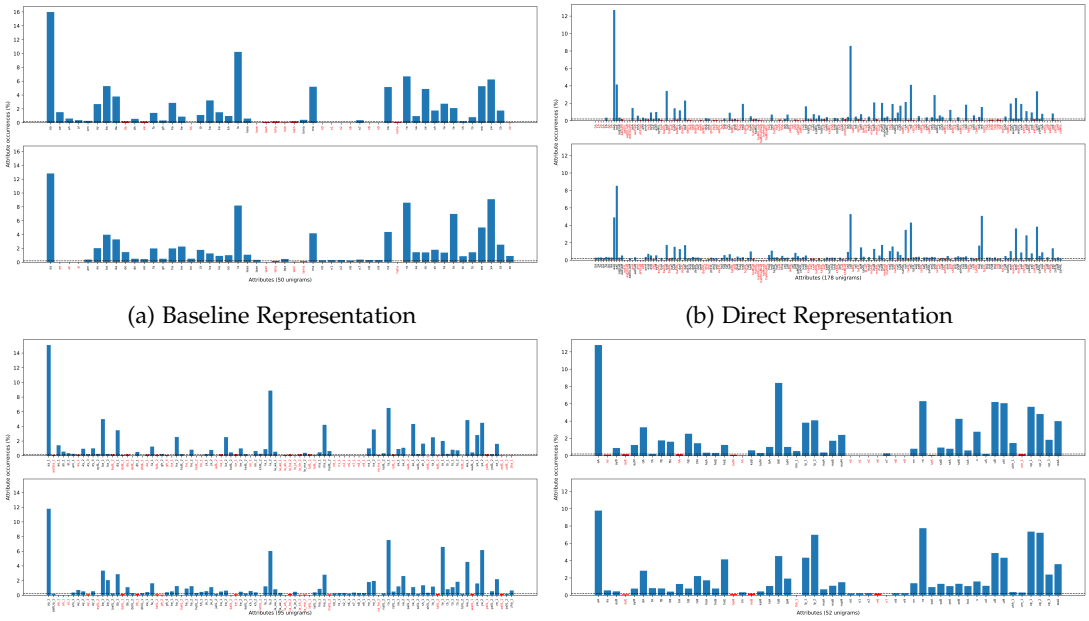


Figure 5.2: Relation between changes in PHOC Size and QbE Performance. It shows that for $\theta \leq 0.2$ the QbE performs the same or even better as in the case of Direct Representation. However, less resources and time are required to reach this performance.

Figure 5.3 illustrates how choosing a $\theta = 0.2$ affects the PHOC built using different unigrams sets. In 5.3b (larger figures are available in the appendix Figure A.2b), we see how the Direct PHOC witnesses the most significant number of attributes being removed. This number can be argued in the view of two aspects. On the one hand, there are certain letters or unigrams that appear only rarely in certain parts of the word,

e.g., numbers appears mostly at the end of the word. Thus, attributes representing these unigrams may have a very limited occurrence rate in one PHOC region, and have a higher rate in the others. This aspect represents a rather general property in the language that is not only specific to this set of unigrams.

On the other hand, many attributes have a rare occurrence in the whole dataset, *e.g.*, ligatures combined with a Shadda occur only once or twice. However, these two aspects define an abstract rule for reduction. When a represented attribute has a minimal projection into actual patterns in the images, attempting to learn such attribute will either slow down the whole learning process or cause a high error rate in prediction.



(a) Baseline Representation

(b) Direct Representation

(c) Sub-Character Representation

(d) Ground Form Representation

Figure 5.3: Distribution of attribute occurrences in Level 2 of their PHOC in different Representations. The top distribution is for the left split of the word, whereas the bottom represents the right split. Below threshold $\theta = 0.2$ attributes are marked in red.

5.5.7 *Comparison*

After evaluating all the experiments, we now present our evaluation results compared to the state-of-the-art results reported in the literature. First of all, we begin with

summarising results of all the previous evaluations, Table 5.8 contains the best results from each approach. Also, a more comprehensive list of evaluated approaches is available at the appendix in Table A.1.

Table 5.8: Results for all the conducted QbE and QbS experiments by their mAP [%][5].

Unigrams	Representation	abc−*d*		abcd−*e*		abcde−*f*		abcde−*s*	
		QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
Baseline	PHOC	96.30	**94.56**	94.67	96.87	96.21	97.17	88.90	93.08
Direct	PHOC	90.17	89.52	93.82	96.96	95.40	96.99	87.52	92.26
Sub-Character	PHOC	91.91	90.96	94.16	96.62	-	-	-	-
GFR	PHOC	**97.66**	94.32	**96.46**	98.35	**96.96**	**98.02**	90.12	**93.57**
GFR	PHOC-*d*	97.24	94.20	96.34	98.37	96.92	97.85	**90.27**	93.49
Baseline	Dynamic PHOC	96.37	94.82	94.93	96.86	96.27	97.35	88.96	92.82
Direct	Dynamic PHOC	90.65	89.10	93.88	96.83	95.36	96.97	88.28	92.84
Sub-Character	Dynamic PHOC	91.81	90.27	94.17	96.54	-	-	-	-
GFR	Dynamic PHOC	97.60	94.00	96.43	**98.56**	96.94	**98.02**	89.85	**93.57**

In Table 5.9, we compare the evaluation results of the investigated approaches with results reported using the same performance measure, mAP. Also, all these results are evaluated only on the default abc−*d* train-test configuration. Since the model proposed and used in [SF16] is the same model used by the evaluated approaches above, the results are directly comparable. Furthermore, both the evaluation protocol and the training data, are shared with [SF16]. However, the Sub-Character approach shows a lower mAP when compared to the original PHOCNet, whereas the proposed GFR clearly outperforms both. Nonetheless, using the proposed Dynamic PHOC with the GFR slightly enhances the mAP performance while enabling more efficient network training.

Compared to the results reported by Uni-script in [ARVK19], all the used approaches in this work show a drawback in performance. This drawback can be attributed to the disparity in training data. All the other mentioned approaches uses IfN/ENIT as the only training data, whereas Uni-script uses IfN/ENIT with additional training datasets

5 For the Sub-Character and GFR, the modified versions (no-sp-on-cnn) and (no-shd) are used respectively. For the Dynamic PHOC evaluations, a threshold $\theta = 0$ is chosen.

Table 5.9: Comparison of the results obtained in this work with state-of-the-art results reported in the literature. Results represent values of mAP [%] on subset *d*. Marked in italic is the best result among our experiments.

Method	IfN/ENIT	
	QbE	QbS
PHOCNet [SF16]	96.11	92.14
Uni-script [ARVK19]	**99.00**	**99.00**
PHOCNet + Sub-Character	91.91	90.96
PHOCNet + Dynamic PHOC†+ Sub-Character	91.81	90.27
PHOCNet + GFR	97.66	94.32
PHOCNet + PHOC-*d* + GFR	97.24	94.20
PHOCNet + Dynamic PHOC*+ GFR	*97.72*	*94.55*

† Threshold $\theta = 0$
* Threshold $\theta = 0.2$

like the George Washington database [LRM04], IAM [MB02], and MLT [NYB+17] [6]. This additional amount of training data enables the network to have more examples for each attribute and thus better learn this attribute.

Although the main focus of this work is to report various approaches for word spotting tasks, we evaluated these approaches simultaneously for the task of text recognition. In order to perform recognition, the PHOC vector is computed for each entry of a lexicon. For a word image, the CNN predicts a PHOC representation and the recognition is then performed by projecting all PHOCs of the lexicon into a subspace and computing the nearest neighbour of the predicted PHOC of the word image. The results for recognition are reported using the Word Error Rate (WER) which measure the fraction of falsely predicted word from total predictions.

Table 5.10 shows how the proposed GFR approach also outperforms both the Baseline and the Sub-Character approaches. It also shows how the reduced size of the Dynamic PHOC assisted the performances in the recognition scenario as well. The

6 Both George Washington and IAM have only Latin scripts, MLT on the other hand have Latin, Arabic and Bengali scripts.

table shows how the performances reported by Poznanski's are superior to all the other methods.

When compared to the HMM models [RF16, AF15], the GFR achieves on par results on the *d* and *e* subsets, but achieves even more significant results on the harder subsets *f* and *s*. These models have sequence structures that are well suited for recognition, whereas our model (PHOCNet) was not designed with the task of text recognition in mind but rather for the recognition-free task of word spotting.

Table 5.10: Comparison of the WER [%] of different methods on the IfN/ENIT dataset. The best result among our experiments is marked in italic.

Method	IfN/ENIT			
	abc−*d*	abcd−*e*	abcde−*f*	abcde−*s*
Bag-of-Features HMM [RF16]	3.0	5.8	8.7	18.2
Multi-Stage HMM [AF15]	1.9	5.1	7.7	15.4
CNN-N-Gram [PW16]	**0.7**	**2.9**	**3.2**	**5.9**
PHOCNet + Baseline	5.3	12.4	9.5	13.8
PHOCNet + Dynamic PHOC†+ Baseline	4.8	12.4	9.5	13.6
PHOCNet + Sub-Character	10.3	19.9	-	-
PHOCNet + Dynamic PHOC†+ Sub-Character	9.8	19.6	-	-
PHOCNet + GFR	3.8	*5.4*	*5.5*	*8.9*
PHOCNet + Dynamic PHOC*+ GFR	*3.5*	5.4	5.6	9.1

† Threshold $\theta = 0$
* Threshold $\theta = 0.2$

6

SUMMARY AND CONCLUSION

By developing the PHOCNet, Sudholt et al. proved that convolutional neural networks are a powerful tool for building excellent word spotting systems. The PHOCNet has set the state-of-the-art performances for word spotting on most benchmark datasets for different scripts including IfN/ENIT for Arabic. This CNN learns a mapping between word images and an attribute representation, such as the PHOC. This mapping is highly dependant on both the choice of the representation and the set of unigrams used. So the two aspects were investigated in this work.

In the first series of experiments, various sets of unigrams were explored, and it is shown how the presented GFR improved the mAP of the used system significantly. Experiments have also shown that the use of GFR not only keeps the shape context of characters intact but also maintains a compact number of unigrams. In addition, we tested using the Sub-Character proposed unigrams, and it shows no improvements to the original unigrams presented in PHOCNet.

Furthermore, two modified versions of the PHOC representation were also evaluated in the second series of experiments. PHOC-*d* is inspired by the virtual vertical segmentation of word characters used by GFR. While this representation does not always improve the performance of the original PHOC, it improves the efficiency during the network's learning when using the GFR set. Apart from that, we presented the Dynamic PHOC, which shows almost the same performance as the original while significantly reducing the number of attributes. A higher number of attributes corresponds to a higher number of parameters in the MLP part of the CNN. A reduction of the number of attributes is therefore an improvement as it results in a network that is easier to optimise and is less prone to overfitting. Finally, the results of the adapted Sub-Character set as well as the proposed GFR were compared to results reported in the literature. GFR achieved competitive accuracies compared to state-of-the-art results.

Nonetheless, in this work, we found that reducing the dimensionality of the attribute representation via the Dynamic PHOC can improve the efficiency of a network. This could also be utilised in future work to other vectorised representations, such as the SPOC. Apart from that, in addition to selection of representation, we found how the amount of training data has a significant influence on the resulting performance since

the proposed approach is only exceeded by methods that use additional data sources (Out of Domain data) (cf. Table 5.9).

Additionally, GFR offers an open set of characters that can be extended and used to represent all other languages of the Arabic scripts like Kurdish, Persian, Urdu, and others. In order to adapt the GFR to Persian for example, only two additional add-on unigrams are required. Since these approaches were only evaluated on IfN/ENIT for Arabic, future work might focus on evaluating these approaches on datasets of other languages that use the Arabic script.

APPENDIX

A

Table A.1: A list of all experiments conducted during the thesis. QbE and QbS results are presented in mAP [%]. Word Error Rate (WER) [%] values are also included. The size refer to the corresponding PHOC size.

Unigrams	Representation	Test Set	Variation	θ	Size	QbE	QbS	WER
Baseline	PHOC	d			700	96.30	94.56	5.27
Baseline	PHOC	d		0	589	96.37	94.82	4.77
Baseline	PHOC	d		0,2	481	96.36	94.44	5.42
Baseline	PHOC	d		0,5	410	96.22	94.49	5.36
Baseline	PHOC	d		0,8	348	95.10	93.81	6.73
Baseline	PHOC	d		1	317	95.16	94.34	7.80
Direct	PHOC	d			2492	90.17	89.52	12.62
Direct	PHOC	d		0	1815	90.65	89.10	11.49
Direct	PHOC	d		0,2	1115	91.11	89.92	11.54
Direct	PHOC	d		0,5	639	89.99	87.92	14.94
Direct	PHOC	d		0,8	419	86.47	85.40	26.15
Direct	PHOC	d		1	347	84.96	84.62	34.33
GFR	PHOC	d			742	93.30	90.55	10.45
GFR	PHOC	d	no-shd		728	97.66	94.32	3.83
GFR	PHOC-*d*	d	no-shd		728	97.24	94.20	4.14
GFR	PHOC-*d*	d			742	93.02	90.12	10.76
GFR	PHOC	d	no-shd	0	650	97.60	94.00	3.91
GFR	PHOC	d		0	666	93.34	90.36	10.66

Unigrams	Representation	Set	Variation	θ	Size	QbE	QbS	WER
GFR	PHOC	d	no-shd	0,2	539	**97.72**	94.55	**3.53**
GFR	PHOC	d	no-shd	1	334	96.17	93.92	8.55
GFR	PHOC	d	no-shd	0,5	434	97.38	93.91	4.10
GFR	PHOC	d	no-shd	0,8	360	96.15	92.91	6.76
Sub-Char	PHOC	d			1358	91.66	90.49	10.04
Sub-Char	PHOC	d	no-sp-no-cnn		1330	91.91	90.96	10.32
Sub-Char	PHOC	d	no-sp		1344	91.82	90.39	10.48
Sub-Char	PHOC	d	no-cnn		1344	91.70	90.22	10.84
Sub-Char	PHOC	d	no-sp-no-cnn	0	1069	91.81	90.27	9.81
Sub-Char	PHOC	d	no-sp-no-cnn	0,2	773	91.58	90.28	10.45
Sub-Char	PHOC	d	no-sp-no-cnn	0,5	518	91.44	90.38	11.28
Sub-Char	PHOC	d	no-sp-no-cnn	0,8	381	89.63	89.84	15.74
Sub-Char	PHOC	d	no-sp-no-cnn	1	325	89.23	89.59	17.67
Baseline	PHOC	e			700	94.67	96.87	12.40
Baseline	PHOC	e		0	595	94.93	96.86	12.46
Direct	PHOC	e		0	1824	93.88	96.83	20.12
Direct	PHOC	e			2492	93.82	96.96	20.22
GFR	PHOC	e	no-shd		728	96.46	98.35	5.35
GFR	PHOC-*d*	e			742	96.24	98.35	12.76
GFR	PHOC-*d*	e	no-shd		728	96.34	98.37	5.35
GFR	PHOC	e	no-shd	0	655	96.43	**98.56**	5.35
GFR	PHOC	e	no-shd	0,2	540	96.30	98.45	5.47
GFR	PHOC	e			742	96.14	98.36	12.86
GFR	PHOC	e		0	669	96.32	98.47	12.86
Sub-Char	PHOC	e			1358	94.49	96.70	19.41
Sub-Char	PHOC	e	no-cnn		1344	94.68	96.99	19.39

Unigrams	Representation	Set	Variation	θ	Size	QbE	QbS	WER
Sub-Char	PHOC	e	no-sp		1344	94.25	96.46	19.59
Sub-Char	PHOC	e	no-sp-no-cnn		1330	94.16	96.62	19.92
Sub-Char	PHOC	e	no-sp-no-cnn	0	1076	94.17	96.54	19.61
Baseline	PHOC	f			700	96.21	97.17	9.48
Baseline	PHOC	f		0	595	96.27	97.35	9.49
Direct	PHOC	f		0	1824	95.36	96.97	15.77
Direct	PHOC	f			2492	95.40	96.99	15.78
GFR	PHOC	f	no-shd		728	96.96	98.02	5.52
GFR	PHOC-*d*	f	no-shd		728	96.92	97.85	5.89
GFR	PHOC-*d*	f			742	96.69	97.78	11.89
GFR	PHOC	f	no-shd	0	655	96.94	98.02	5.55
GFR	PHOC	f	no-shd	0,2	542	96.97	97.97	5.70
GFR	PHOC	f			742	96.87	97.79	10.96
GFR	PHOC	f		0	669	96.87	97.84	10.67
Baseline	PHOC	s			700	88.90	93.08	13.76
Baseline	PHOC	s		0	595	88.96	92.82	13.57
Direct	PHOC	s		0	1824	88.28	92.84	22.42
Direct	PHOC	s			2492	87.52	92.26	22.61
GFR	PHOC	s	no-shd		728	90.12	93.57	8.92
GFR	PHOC-*d*	s	no-shd		728	90.27	93.49	8.60
GFR	PHOC-*d*	s			742	90.41	93.90	17.71
GFR	PHOC	s	no-shd	0	655	89.85	93.57	8.79
GFR	PHOC	s	no-shd	0,2	542	89.99	93.56	9.11
GFR	PHOC	s		0	669	90.80	94.26	17.71
GFR	PHOC	s			742	90.42	93.56	17.67

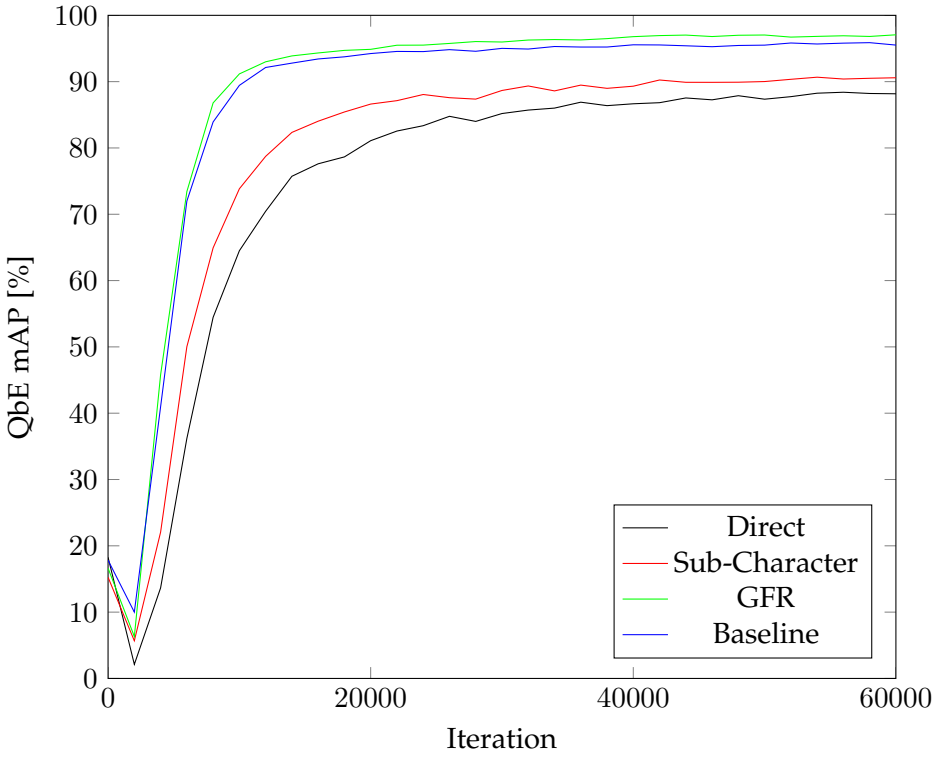
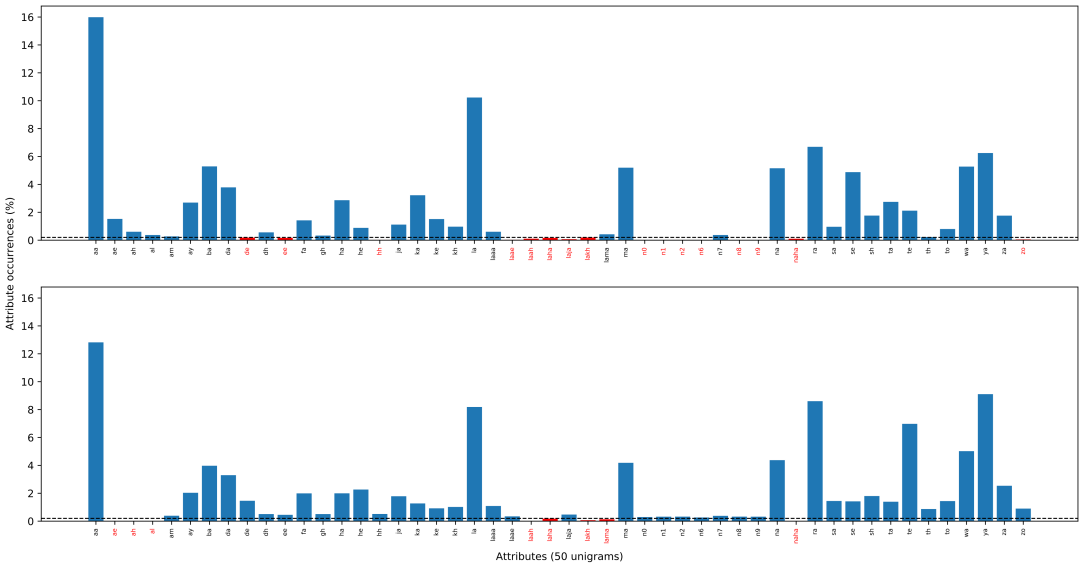
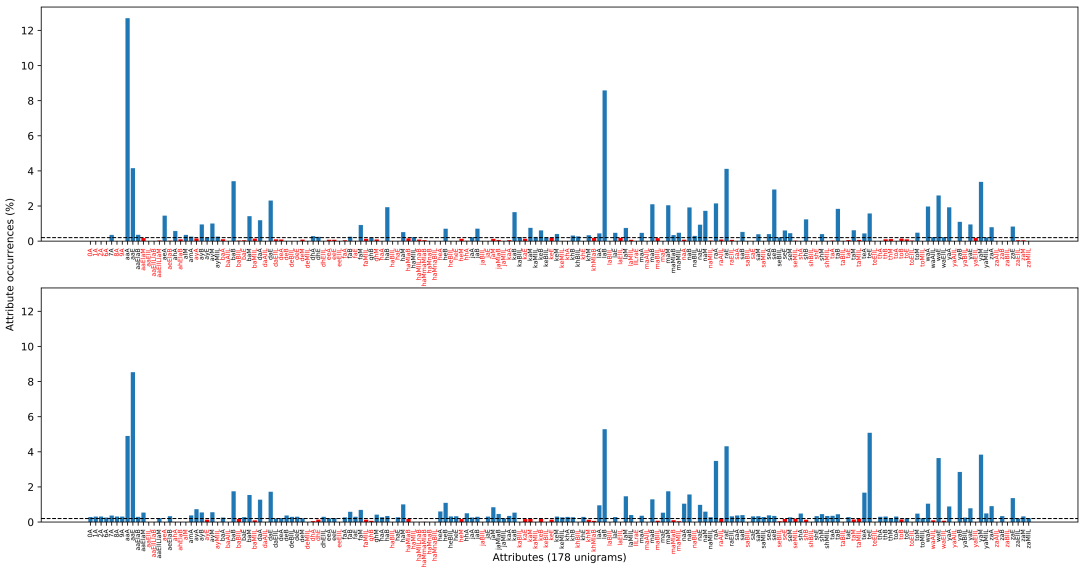


Figure A.1: The figure displays the mAP over the training iterations for the four QbE Relation between changes in PHOC Size and QbE Performance. It shows that for $\theta \leq 0.2$ the QbE has performed the same or even - in case of Direct Representation - better. However, less resources and time were required to reach this Performance.

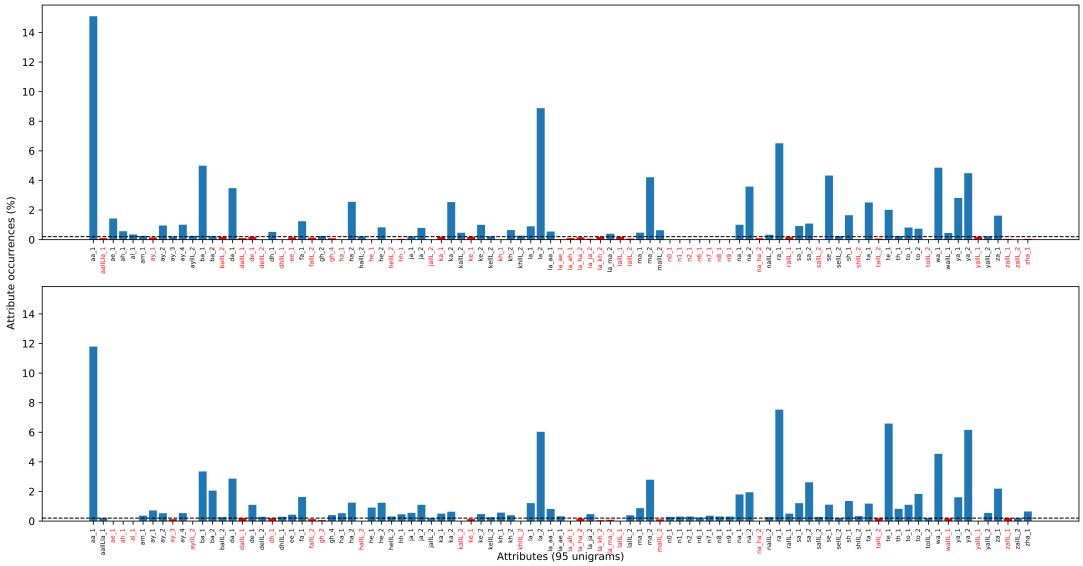


(a) Baseline Representation

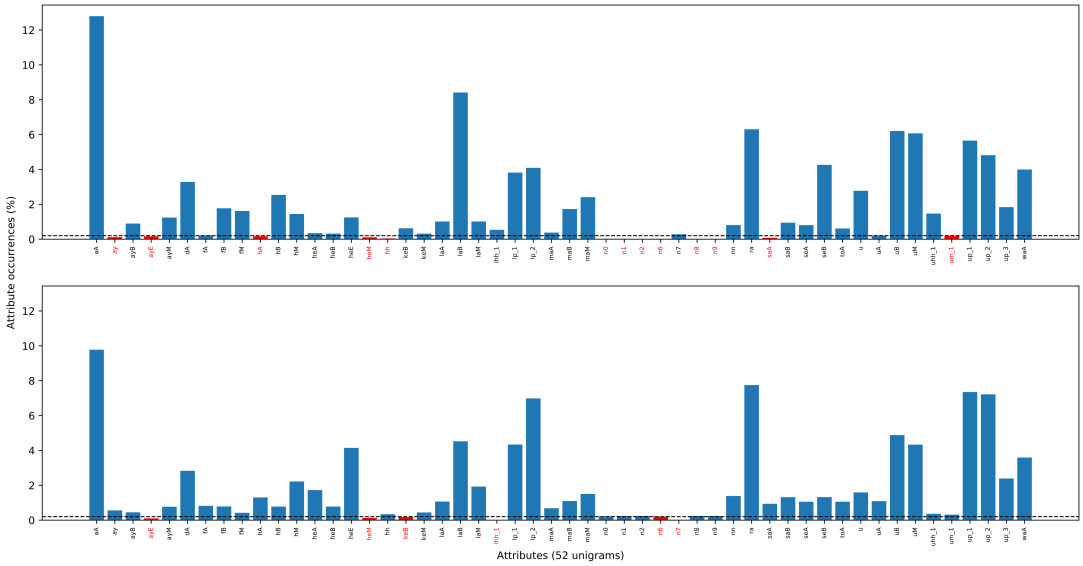


(b) Direct Representation

Figure A.2: Distribution of attribute occurrences in Level 2 of their PHOC in different Representations. The top distribution is for the left split of the word, whereas the bottom represents the right split. Below threshold $\theta = 0.2$ attributes are marked in red.



(c) Sub-Character Representation



(d) Ground Form Representation

Figure A.2: (cont.) Distribution of attribute occurrences in Level 2 of their PHOC in different Representations. The top distribution is for the left split of the word, whereas the bottom represents the right split. Below threshold $\theta = 0.2$ attributes are marked in red.

Name	Extended Alphabet				Rasm Alphabet			
	Final	Medial	Initial	Isolated	Final	Medial	Initial	Isolated
alef	ا	-	-	ا	ا	-	-	ا
beh	ب	بـ	بـ	ب	ب	بـ	بـ	ب
teh	ت	تـ	تـ	ت				
theh	ث	ثـ	ثـ	ث				
noon	ن	نـ	نـ	ن	ن			ن
yeh	ي	يـ	يـ	ي	ى			ى
alef maqsurah	ى	ـى	ـى	ى				
jeem	ج	جـ	جـ	ج	ح	حـ	حـ	ح
hah	ح	حـ	حـ	ح				
khah	خ	خـ	خـ	خ				
dal	د	-	-	د	د	-	-	د
thal	ذ	-	-	ذ				
reh	ر	-	-	ر	ر	-	-	ر
zain	ز	-	-	ز				
seen	س	سـ	سـ	س	س	سـ	سـ	س
sheen	ش	شـ	شـ	ش				
sad	ص	صـ	صـ	ص	ص	صـ	صـ	ص
dad	ض	ضـ	ضـ	ض				
tah	ط	طـ	طـ	ط	ط	طـ	طـ	ط
zah	ظ	ظـ	ظـ	ظ				
ain	ع	عـ	عـ	ع	ع	عـ	عـ	ع
ghain	غ	غـ	غـ	غ				
feh	ف	فـ	فـ	ف	ف	فـ	فـ	ف
qaf	ق	قـ	قـ	ق				
kaf	ك	كـ	كـ	ك	ك	كـ	كـ	ك
lam	ل	لـ	لـ	ل	ل	لـ	لـ	ل
meem	م	مـ	مـ	م	م	مـ	مـ	م
heh	ه	هـ	هـ	ه	ه	هـ	هـ	ه
waw	و	-	-	و	و	-	-	و

Table A.2: This table contains the extended Arabic alphabet side-to-side with the original *rasm* characters used to build it. Note how most rasm glyphs are shared between normal characters.

Isolated	Initial	Medial	Final
ا	-	-	-
ب	بـ	ـبـ	-
ح	حـ	ـحـ	-
د	-	-	-
ر	-	-	-
س	سـ	ـسـ	-
ص	صـ	-	-
ط	-	-	-
ع	عـ	ـعـ	ـع
ف	فـ	ـفـ	-
-	كـ	ـكـ	-
ل	لـ	ـلـ	-
م	مـ	ـمـ	-
ن	-	-	-
و	-	-	-
ه	هـ	ـهـ	ـه
ى	-	-	-
ء	-	-	-

(a) List of all unique patterns for the ground form group

Name	Pattern
Shadda	ّ
One Upper point	٠
Two Upper points	٠٠
Three Upper points	٠٠٠
Upper Hamza	ٔ
One Lower point	٠
Two Lower points	٠٠
Madda	~
Lower Hamza	ٕ

(b) List of all unique patterns for the add-on group

Table A.3: List of unigrams used in the GFR set.

BIBLIOGRAPHY

[AF15] Ahmad, Irfan ; Fink, Gernot A.: Multi-stage HMM based Arabic Text Recognition with Rescoring. In: *Int. Conf. on Document Analysis and Recognition (ICDAR)*. Nancy, France, 2015, S. 751–755

[AFM14] Ahmad, Irfan ; Fink, Gernot A. ; Mahmoud, Sabri A.: Improvements in sub-character HMM model based Arabic Text Recognition. In: *Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Crete, Greece, 2014, S. 537–542

[AGFV14] Almazán, Jon ; Gordo, Albert ; Fornés, Alicia ; Valveny, Ernest: Word Spotting and Recognition with Embedded Attributes. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 36 (2014), Nr. 12, S. 2552–2566

[ARFM13] Ahmad, Irfan ; Rothacker, Leonard ; Fink, Gernot A. ; Mahmoud, Sabri A.: Novel sub-character HMM models for Arabic text recognition. In: *Int. Conf. on Document Analysis and Recognition (ICDAR)*. Washington, DC, USA, 2013, S. 658–662

[ARVK19] Al-Rawi, Mohammed ; Valveny, Ernest ; Karatzas, Dimosthenis: Can One Deep Learning Model Learn Script-Independent Multilingual Word-Spotting? In: *Int. Conf. on Document Analysis and Recognition (ICDAR)*. Sydney, Australia, 2019, S. 260–267

[Bis95] Bishop, Christopher M.: *Neural Networks for Pattern Recognition*. Oxford university press, 1995

[EB16] Encyclopaedia Britannica, The E.: *Arabic alphabet*. https://www.britannica.com/topic/Arabic-alphabet. Version: Aug 2016

[Fuk80] Fukushima, Kunihiko: Neocognitron: A self-organizing Neural Network Model for a Mechanism of Pattern Recognition unaffected by shift in Position. In: *Biological cybernetics* 36 (1980), Nr. 4, S. 193–202

[Gac09] Gacek, Adam: *Arabic manuscripts: a vademecum for readers*. Bd. 98. Brill, 2009

[GBB11] Glorot, Xavier ; Bordes, Antoine ; Bengio, Yoshua: Deep Sparse Rectifier Neural Networks. In: *Proc. Int. Conf. on Artificial Intelligence and Statistics*. Fort Lauderdale, USA, 2011, S. 315–323

[GBC16] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[GSGN17] Giotis, Angelos P. ; Sfikas, Giorgos ; Gatos, Basilis ; Nikou, Christophoros: A Survey of Document Image Word Spotting Techniques. In: *Pattern Recognition* 68 (2017), S. 310–332

[Kap08] Kaplony, Andreas: What are those few dots for? Thoughts on the orthography of the Qurra Papyri (709-710), the Khurasan Parchments (755-777) and the inscription of the Jerusalem Dome of the Rock (692). In: *Arabica* 55 (2008), Nr. Fasc. 1, S. 91–112

[KB15] Kingma, Diederik P. ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *Proc. Int. Conf. on Learning Representations (ICLR)* (2015)

[KDJ16] Krishnan, Praveen ; Dutta, Kartik ; Jawahar, C. V.: Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text. In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016, S. 289–294

[KL20] Karpathy, Andrej ; Li, Fei-Fei: *CS231n Convolutional Neural Networks for Visual Recognition*. http://cs231n.github.io/. Version: Mar. 2020. – Online

[LBD+90] LeCun, Yann ; Boser, Bernhard E. ; Denker, John S. ; Henderson, Donnie ; Howard, Richard E. ; Hubbard, Wayne E. ; Jackel, Lawrence D.: Handwritten Digit Recognition with a back-propagation Network. In: *Advances in Neural Information Processing Systems*, 1990, S. 396–404

[Lew99] Lewis, Geoffrey: *The Turkish Language Reform: A Catastrophic Success*. OUP Oxford, 1999

[LG06] Lorigo, L. M. ; Govindaraju, V.: Offline Arabic Handwriting Recognition: a Survey. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 28 (2006), Nr. 5, S. 712–724

[LRM04] Lavrenko, Victor ; Rath, Toni M. ; Manmatha, Raghavan: Holistic Word Recognition for Handwritten Historical Documents. In: *Proc. of the First Int. Workshop on Document Image Analysis for Libraries*, 2004, S. 278–287

[MA09] Märgner, V. ; Abed, H. E.: ICDAR 2009 - Arabic Handwriting Recognition Competition. In: *Int. Conf. on Document Analysis and Recognition (ICDAR)*. Barcelona, Spain, 2009, S. 1383–1387

[MA10] Märgner, V. ; Abed, H. E.: ICFHR 2010 - Arabic Handwriting Recognition Competition. In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Kolkata, India, 2010, S. 709–714

[Maj96] Majidi, Mohammad R.: *Einführung in die arabisch-persische Schrift*. Buske Helmut Verlag GmbH, 1996

[MB02] Marti, Urs-Viktor ; Bunke, Horst: The IAM-database: an English Sentence Database for Offline Handwriting Recognition. In: *Int. Journal on Document Analysis and Recognition (IJDAR)* 5 (2002), Nr. 1, S. 39–46

[MP72] Minsky, Marvin ; Papert, Seymour: Perceptrons: An Introduction to Computational Geometry. In: *Bulletin of the American Mathematical Society* 78 (1972), Nr. 1, S. 12–15

[NYB+17] Nayef, Nibal ; Yin, Fei ; Bizid, Imen ; Choi, Hyunsoo ; Feng, Yuan ; Karatzas, Dimosthenis ; Luo, Zhenbo ; Pal, Umapada ; Rigaud, Christophe ; Chazalon, Joseph u. a.: ICDAR2017 Competition on Robust Reading: Multi-lingual Scene Text Detection and Script Identification Challenge–RRC-MLT. In: *Int. Conf. on Document Analysis and Recognition (ICDAR)* Bd. 1. Kyoto, Japan, 2017, S. 1454–1459

[Omn20] Omniglot: *"Alphabets and Writing Systems"*. https://www.omniglot.com/writing/. Version: Mar. 2020. – Online

[PGC+17] Paszke, Adam ; Gross, Sam ; Chintala, Soumith ; Chanan, Gregory ; Yang, Edward ; DeVito, Zachary ; Lin, Zeming ; Desmaison, Alban ; Antiga, Luca ; Lerer, Adam: Automatic differentiation in PyTorch. (2017)

[PMM+02] Pechwitz, Mario ; Maddouri, S S. ; Märgner, Volker ; Ellouze, Noureddine ; Amiri, Hamid u. a.: IfN/ENIT-database of Handwritten Arabic Words. In: *Proc. of CIFED* Bd. 2 Citeseer, 2002, S. 127–136

[PW16] Poznanski, Arik ; Wolf, Lior: CNN-N-Gram for Handwriting Word Recognition. In: *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA, 2016, S. 2305–2314

[RASC14] Razavian, Ali S. ; Azizpour, Hossein ; Sullivan, Josephine ; Carlsson, Stefan: CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In: *Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*. Columbus, OH, USA, 2014, S. 512–519

[RF16] Rothacker, Leonard ; Fink, Gernot A.: Robust Output Modeling in Bag-of-Features HMMs for Handwriting Recognition. In: *Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016, S. 199–204

[Ros58] Rosenblatt, Frank: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. In: *Psychological Review* 65 (1958), Nr. 6, S. 386–408

[Sch15] Schmidhuber, Jürgen: Deep Learning in Neural Networks: An Overview. In: *Neural Networks* 61 (2015), S. 85 – 117

[SF16] Sudholt, Sebastian ; Fink, Gernot A.: PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents. In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016, S. 277–282

[SF17] Sudholt, Sebastian ; Fink, Gernot A.: Evaluating Word String Embeddings and Loss Functions for CNN-based Word Spotting. In: *Int. Conf. on Document Analysis and Recognition (ICDAR)* Bd. 1. Kyoto, Japan, 2017, S. 493–498

[SF18] Sudholt, Sebastian ; Fink, Gernot A.: Attribute CNNs for Word Spotting in Handwritten Documents. In: *Int. Journal on Document Analysis and Recognition (IJDAR)* 21 (2018), Nr. 3, S. 199–218

[SGC15] Shankar, Sukrit ; Garg, Vikas K. ; Cipolla, Roberto: Deep-carving: Discovering visual attributes by carving deep neural nets. In: *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. Boston, MA, USA, 2015, S. 3403–3412

[Sud18] Sudholt, Sebastian: *Learning Attribute Representations with Deep Convolutional Neural Networks for Word Spotting*. Dortmund, Germany, TU Dortmund University, Dissertation, 2018

[SZ15] SIMONYAN, Karen ; ZISSERMAN, Andrew: Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *Int. Conf. on Learning Representations (ICLR)*. San Diego, CA, USA, 2015

[WB16] WILKINSON, Tomas ; BRUN, Anders: Semantic and Verbatim Word Spotting Using Deep Neural Networks. In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016, S. 307–312

[ZC88] ZHOU, Yi-Tong ; CHELLAPPA, Rama: Computation of Optical Flow using a Neural Network. In: *Proc. Int. Conf. on Neural Networks (ICNN)*. San Diego, CA, USA, 1988, S. 71–78

[ZPJ+16] ZHONG, Z. ; PAN, W. ; JIN, L. ; MOUCHÈRE, H. ; VIARD-GAUDIN, C.: SpottingNet: Learning the Similarity of Word Images with Convolutional Neural Network for Word Spotting in Handwritten Historical Documents. In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition (ICFHR)*. Shenzhen, China, 2016, S. 295–300