technische universität
dortmund

Densely Connected Convolutional Networks for Word Spotting in Handwritten Documents

Master Thesis

Fabian Wolf
September 25, 2018

Supervisors:
Prof. Dr.-Ing. Gernot A. Fink
Wilmar Fernando Moya Rueda, M.Sc.

Fakultät für Informatik
Technische Universität Dortmund
http://www.cs.uni-dortmund.de

Contents

1 INTRODUCTION

Reading and understanding written text constitutes a complex task, which humans are able to perform almost perfectly after years of training. Written document collections provide a huge amount of information, which is manually accessible, but the shier extent makes an automated process desirable. Powered by the progress in fields as computer vision and machine learning, document image analysis has become an active field of research.

Different methods tackled the task of processing and transcribing written text and they have been quite successful with regards to machine-written documents. When it comes to handwritten documents, accuracies decrease and the performances of classic approaches are limited. This observation is mainly due to the structural differences between hand- and machine-written text. Machine-written text generally has a uniform visual appearance independent of the writer. In case of handwriting, no two words share the exact same visual appearance even when written by the same person. Considering documents written by different writers, their handwritings show huge variation in style and visual appearance. A system to identify a specific word has to cope with this high intra-class variance. Furthermore, even a single misidentified character results in misclassification. Recognition methods that aim at directly transcribing a handwritten text often struggle when confronted with these requirements.

Another approach for understanding handwritten text follows the concept of word spotting. Instead of considering the problem as a recognition problem, it is interpreted as a form of retrieval task. The entire document is treated as a collection of individual word images. Word spotting is then defined as the task of retrieving all images that are relevant with respect to a given query. The query can be either a word image itself or a string. Different methods have achieved good results, making word spotting a suitable method for indexing and retrieving information from handwritten text automatically.

The general requirement of a word spotting system is to map the word image onto a representation that allows to rank the image collection according to their relevance with respect to a query. Here, attribute representations, which are based on character occurrences and location, found popular use. Still, approximating the mapping between visual appearance and attribute representation is highly complex. Most popular methods rely on machine learning techniques. Considering the applied

methods, similar developments took place in the fields of computer vision and word spotting. Feature based approaches, which have been popular at first, were recently replaced by methods based on convolutional neural networks (CNN). CNNs have been shown to be an extremely powerful tool and strongly influenced the field of word spotting.

The performance of these methods is highly determined by network architectures. Driven by the increasing possibilities in terms of computational power and memory, CNNs have become increasingly complex, and especially they employ high numbers of layers. A lot of work on different network architectures continuously increased performances in various applications. Recently, different networks incorporating identity paths in their network architecture have become popular. One of these highly powerful architectures applies the concept of densely connected neural networks. This dense network structure is based on connecting each layer of the network to all subsequent layers. Following this design pattern allows each layer to access all previously generated feature maps. This pattern is combined with small numbers of filters. Experiments show that the network highly benefits from this idea of feature reuse.

This thesis investigates the capabilities of the dense connectivity pattern in the context of word spotting and is inspired by a method using the so called PHOCNet. Different dense network architecture are proposed as a model to approximate the mapping between word image and attribute representation. While the dense connectivity pattern only provides a general network structure its actual architecture is determined by several hyperparameters. Therefore, another focus of this work is to get an insight on how different hyperparameters affect the network's properties and performance.

This thesis is structured as follows. Chapter 2 introduces the fundamental concepts the proposed method is based on. Therefore, a brief overview on the basic concepts and methods in the field of word spotting is given. Since neural networks and especially CNNs have become the state-of-the-art method and they are an essential tool for the proposed method, their fundamental concepts are discussed. Chapter 3 presents the works that inspired this thesis. The first part will discuss recent developments in the field of deep network architecture, which are focused on the utilization of identity paths. Furthermore, two influential works using CNNs for word spotting are presented in the second part of chapter 3. Chapter 4 discusses the proposed method, which is evaluated by the experiments presented in chapter 5. Finally, the conclusions drawn from the experimental evaluation are summarized and discussed in chapter 6.

2 FUNDAMENTALS

This chapter covers the fundamental tools and concepts which motivated the proposed method. Section 2.1 introduces the problem of word spotting and discusses how it has been addressed in research. Systems based on CNNs have become increasingly popular and outperformed traditional approach. Taking this as a motivation, the proposed method also relies on the use of a CNN. Therefore, section 2.2 introduces the fundamental concepts of artificial neural networks. Due to their relevance for this thesis, convolutional neural networks are discussed in more detail in section 2.3.

2.1 WORD SPOTTING

Handwritten text is a major source of information. While a vast amount of data is provided in form of handwritten documents, extracting relevant parts is often tedious and time-consuming. Especially the exploration of historic document collections, which are not annotated and often poorly indexed, is troublesome. Therefore, transcribing, indexing and analyzing handwritten documents in an automated fashion, is of special interest in many fields of research.

Optical character recognition (OCR) methods aim at directly transcribing a document image. Even though these approaches render relative good results for machine printed text, they struggle when facing handwritten documents. In contrast to machine printed text, the appearance of handwriting shows a huge variability. The performance of OCR methods is further limited by document degradation, especially present in historic documents [GSGN17].

One of the most popular approaches to overcome these limitation is the concept of word spotting, first proposed by [MHRC96]. Instead of transcribing a document entirely, the authors proposed to first segment the images of document pages into word images. Transforming the document into a collection of words, allows the creation of word clusters. The images in one cluster are visually similar and therefore probable to share the same annotation. As discussed in [RM07], this is particularly suitable for indexing historic documents. A label can be assigned manually to a cluster of interest, leading to a partial transcription of the document. Thereby, access to the

textual information is granted without requiring the transcription of the document's entire content.

In a more general form, word spotting can be considered as a retrieval task. A specific query word is provided to the word spotting system, which returns a subset of the collection of word images. The relevance of the retrieved word images with respect to the query determines the system's performance. A perfectly accurate system would return all occurrences of the word in the considered document, with no additional irrelevant words.

In the literature different word spotting systems mainly address the two scenarios of segmentation-based and segmentation-free word spotting. In case of segmentation-free word spotting, an entire document page is considered. No further information on the position or extent of individual words and lines is provided. Extracting the collection of word images is considered to be part of the word spotting task. This work addresses the second scenario. It is assumed that the problem of segmenting the document into words is solved independently. Hence, the word spotting task is only concerned with finding those word images relevant with respect to a given query.

Another factor determining the design and performance of a word spotting system is the representation of the query word. The two most prominent paradigms are *query-by-example* (QbE) and *query-by-string* (QbS). In case of *query-by-example*, the query word is provided in form of an example image. Therefore, the retrieval task is reduced to finding those images, which are visually similar to the query image. The biggest drawback is the problem of manually identifying the query word. This can be tedious if the relevant word rarely occures in the text. Furthermore, the system is useless for checking whether a specific word is present in a document. This special case is already solved by the manual identification of the query image. The *query-by-string* paradigm overcomes these limitations. Instead of using a word image, the query is represented by a string. Even though this solves the problem of identifying a query image, a new requirement is imposed to the system. Retrieving a subset of word images from a query string, cannot be based solely on visual similarity. A *query-by-string* system needs a mapping between word image and a corresponding textual representation.

One of the first methods in the field of segmentation-based word spotting was proposed by [MHRC96]. In a first step, the considered word images are binarized by thresholding. A designated query word is then matched against all other words of the document. Matching is performed by simply XOR'ing two images and the use of Euclidean Distance Mapping [Dan80]. Other approaches investigated the use of sequential models for matching word images. In [RM07] the distribution of ink along one axis of the word image is used to generate a characteristic word profile. Two profiles of the same word are unlikely to line up due to variations in handwriting.

Therefore, Dynamic Time Warping (DTW) [SK99] is used to align two profiles to improve the significance of the distance measure. [RSP09] and [FKFB10] investigated Hidden Markov Models (HMMs) in the context of word spotting and showed that the trained subword models outperform the matching approaches based on DTW.

Holistic representations became increasingly popular, especially when combined with local descriptors. In [RATL11] the authors use SIFT features to represent image patches. Based on local descriptors, a bag-of-visual-words model is used to perform segmentation-free word spotting. The method was further refined in [RATL15] by latent semantic analysis and product quantization. In general, the predominant local descriptors in the field of word spotting are SIFT descriptors [RATL11], [ARTL13], [RATL15], [RF15], [SF15], geometric features [RM07], [FFMB12] and HOG-based descriptors [AFV13], [GMAJ13]. Many approaches based on matching word images do not incorporate any information on the transcription of an image during training. While this relieves the requirement of annotated training data, it comes with the drawback that *query-by-string* word spotting cannot be performed directly.

One of the most influential methods in the field of segmentation-based word spotting was presented in [AGFV14]. The authors proposed to use an attribute representation called Pyramidal Histogram of Characters (PHOC) (see section 3.2.1). This binary word string embedding encodes the presence of a character in a specific split of the word. While generating a PHOC vector from a string is trivial, finding the attribute representation for a word image is not. In a first step, the word images are projected into an attribute space using Fisher Vectors [PSM10]. Based on the corresponding labels, represented by PHOC vectors, a set of Support Vector Machines (SVM) is trained. A common subspace between Fisher Vectors and PHOC embeddings is learned, making word image and string embedding comparable. Therefore, the matching step of word spotting can be performed by projecting the word images into a common subspace and comparing them by a suitable distant measure. By incorporating word labels in the supervised training procedure, which requires annotated training data, most unsupervised approaches were outperformed. Furthermore, the existence of a common subspace between string embedding and word images allows *query-by-string* word spotting. As discussed in [AGFV14], the method is not restricted to Fisher Vectors and SVMs. In general, every method which is able to predict a PHOC representation from a word image is suitable.

Inspired by the success in the field of computer vision, Convolutional Neural Networks (CNNs) found their way into word spotting. [KDJ16] improved the method of [AGFV14] by using a CNN as a feature extractor. Training the SVMs is then based on the features extracted by the CNN, effectively replacing Fisher Vectors. The method presented in [SF16] further exploits the use of a CNN in combination with an attribute

representation. Learning the PHOC vector of a word image is considered a multi-label classification task. By training a designated CNN to learn the mapping between image and PHOC representation, other methods were outperformed, making it the state-of-the-art approach up to today.

A slightly different method was proposed by [WB16]. Instead of training the mapping between word string embedding and word image in an end-to-end fashion, a triplet network is used. Three identical CNNs, sharing the same weights, are used to learn a feature vector. This is done by providing triplets of images, with two images of the same and one of a different class. The loss function used in [WB16] aims at minimizing the distance between two feature vectors of the same class, while maximizing the minimal distance between two feature vectors, corresponding to word images of different classes. After the CNN is trained to map a word image on a discriminative feature vector, a multi layer perceptron (MLP) is trained. The MLP projects the feature vector on a word string embedding such as PHOC. The results presented in [WB16] did not show any significant improvement compared to [SF16], but still outperformed methods not based on CNNs.

For a detailed and extensive survey of word spotting methods and challenges see [GSGN17].

2.2 ARTIFICIAL NEURAL NETWORKS

Word spotting methods are highly influenced by the field of machine learning, which relies on computational models that are used to approximate arbitrary complex functions. One of these models took inspiration from how human beings and animals process information. A biological neural network (BNN) realizes essential functions of computation such as storing, processing and transmitting information. The fundamental building blocks of a BNN are small structures called neurons. A neuron essentially receives signals and generates a composite signal which is then transmitted to another neuron [Kono5]. Everything the human brain is capable of is based on the interconnection and communication of a large number of such neurons. This insight on the basic functionalities of a biological nervous system inspired the idea of artificial neural networks. A powerful computational model is realized by a network of numerous simple basic functions. Nowadays, in many state-of-the-art applications, neural networks are the predominant tool due to their performance and flexibility.

2.2.1 *Perceptron*

An artificial neural network (ANN) can be considered as a network of primitive functions [Roj96]. An artificial neuron is a computing unit, first proposed by Warren McCulloch and Walter Pitts in 1943 [MP43]. This simple computation unit takes n binary signals as an input and processes them in two parts. First, the sum of the neuron's inputs is computed, which is followed by a non-linear function, mapping the output on a finite range. In the case of the McCulloch-Pitts neuron, the non-linear step function with a threshold of θ is used. Therefore, the output of the neuron, given an arbitrary number of binary inputs $x_1, x_2, ..., x_n$, is defined by:

$$f(x_1, x_2, ..., x_n) = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} x_i \geqslant \theta \\ 0, & \text{otherwise} \end{cases} \tag{2.2.1}$$

Despite its simple structure, a single McCulloch-Pitts neuron is capable of approximating different monotonic logical functions such as **AND**, **NOR** or **NOT**. The capabilities of the artificial neuron were further extended by Frank Rosenblatt [Ros58], who introduced weights corresponding to each input. This so called perceptron was generalized by Minsky and Papert [MP69]. In contrast to the classic perceptron, input signals with a real value in the interval $x \in [0, 1]$ are used. Furthermore, the weights are not set to a fixed value. By adapting the corresponding weights the Minsky-Papert perceptron can be used to approximate different functions and solve different problems, such as linear classification. It was shown that if a Minsky-Papert perceptron is capable of representing a specific function, a corresponding set of weights can be learned [MP69].

2.2.2 *Feedforward Networks*

An artificial neural network is based on the composition of perceptrons as basic computing units as introduced in section 2.2.1. The simplest form of a neural network is the single layer perceptron as shown in figure 2.2.1. It consists of a single perceptron and defines a mapping $y = f(\mathbf{x})$ depending on its weight vector $\mathbf{w}$ and a bias b. The output of the single layer perceptron simply results from the composition of the dot product of inputs and weights and the non-linear activation function φ:

$$f(\mathbf{x}) = \varphi(\mathbf{w}^T\mathbf{x} + b) \tag{2.2.2}$$

The activation function corresponds to the thresholding part of the classic perceptron. Popular activation functions replace the simple thresholding behavior by more

complex functions, as discussed in section 2.2.3. Since information flows through the network without any cycles or feedback connections, such a network is called feedforward network [Kono5]. Even though a single perceptron already implements a variety of functions, its capabilities are still quite limited. Following the analogy with biological neural networks, more complex functions are realized by composing multiple perceptrons.

The information flow and composition of functions in a feedforward network can be described by an acyclic graph [GBC17]. A simple and commonly used structure is based on a consecutive evaluation of basic functions. Consider such a chain structure with three basic functions $f^{(1)}$,$f^{(2)}$ and $f^{(3)}$ [GBC17]. The overall output of the network is given by $f(\mathbf{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{x})))$. Each function of the chain is called a layer of the network, while the length of the chain determines the depth [GBC17]. The arbitrary complex combination of network structures, activation and basic functions makes artificial neural networks a powerful and flexible model of computation suitable for a variety of tasks in machine learning and computational intelligence.

2.2.3 *Activation Functions*

As discussed in section 2.2.2, a neuron usually composes two functions. In a first step, the input data is processed based on the corresponding weights and biases. Second, the neuron's output is generated by a non-linear activation function $\varphi(\cdot)$. The following section discusses the predominant types of activation functions and their derivatives.

The sigmoid non-linearity is similar to a step function. It takes a single real number and maps it to the finite range of $[0, 1]$. As shown in figure 2.2.2a, the sigmoid function is close to zero for high negative input values. For high positive inputs, the

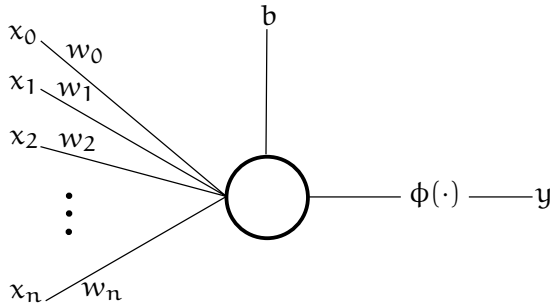


Figure 2.2.1: The single layer perceptron (SLP) as a simple feedforward network [Roj96].

sigmoid function approaches one. Again, the behavior can be interpreted analogue to the behavior of a biological neuron. The neuron is either not activated, in case of high negative values, or completely activated, in case of high positive inputs. Mathematically, this behavior is realized by the following continuous and differentiable function [Roj96]:

$$\varphi(x) = sg(x) = \frac{1}{1+e^{-x}} \qquad \varphi'(x) = sg'(x) = \frac{e^{-x}}{(1+e^{-x})^2} \tag{2.2.3}$$

Even though the sigmoid activation function was historically popular, it is rarely used nowadays. A major drawback of the sigmoid activation are its not zero-centered outputs. This is unfavorable for the training procedure with stochastic gradient descent. The undesirable dynamics introduced by the sigmoid function can be elevated by using a scaled and translated sigmoid function. The resulting hyperbolic tangent function (see figure 2.2.2b) is defined as [Roj96]:

$$\varphi(x) = \tanh(x) = 2(sg(x) - 1) = \frac{1-e^{-x}}{1+e^{-x}} \tag{2.2.4}$$

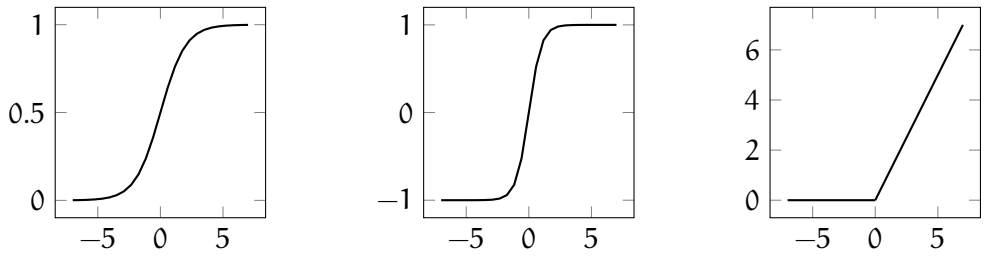
$$\varphi'(x) = \tanh'(x) = 1 - \tanh^2(x)) \tag{2.2.5}$$

Although the hyperbolic tangent is zero-centered, it shares another undesirable property with the sigmoid function. As discussed in section 2.2.4, training a network relies on calculating gradients. It can be easily seen that the gradient of the sigmoid and hyperbolic tangent is close to zero for values close to saturation at zero or one. Especially in the case of deep networks, the gradient vanishes, while information flows through the network. This hinders the network to efficiently learn a set of weights [PMB12].

[GBB11] proposed a different activation function. The rectified linear unit (ReLU) thresholds the input value at zero and performs a linear mapping for positive inputs (see figure 2.2.2c). The resulting function can be summarized as:

$$\varphi(x) = f(x) = \max(0, x) \qquad \varphi'(x) = \begin{cases} 1, & \text{if } x \leqslant 0 \\ 0, & \text{if} x > 0 \end{cases} \tag{2.2.6}$$

As shown by [KSH12], using a ReLU instead of sigmoid or hyperbolic tangent activation strongly improves performance. Due to its linear, not saturating form, stochastic gradient descent converges faster to an optimal set of weights. Furthermore, a ReLU is computationally simple. It can be implemented by simply thresholding a value, opposed to the complex calculation of exponential functions, in case of sigmoid or hyperbolic tangent activation. These properties make the ReLU one of the most widespread used activation functions in state-of-the art networks.



(a) Sigmoid (b) Hyperbolic Tangent (c) ReLU

Figure 2.2.2: Common activation functions [FFJK17].

2.2.4 *Gradient Descent*

In order to implement a desired function with an artificial neural network, a set of weights and biases is required. With increasing network complexity, finding a corresponding set of parameters is not trivial and might not be possible in a deterministic way. A popular approach to solve this problem relies on learning a set of parameters. Therefore, the network is trained on a given training data set and the gradient descent algorithm is used to minimize a cost function [GBC17][Roj96].
The cost function quantifies how well a given set of parameters approximates the desired function. As a simple example for a cost function, the mean squared error is considered [Nie15]. For a given set of weights $\mathbf{w}$ and biases $\mathbf{b}$, the output of the network $f(\mathbf{x})$ is determined for n training examples $\mathbf{x}$. With y as the desired output of the network, the mean square error function C is defined as follows.

$$C(\mathbf{w}, \mathbf{b}) = \frac{1}{2n} \sum_{\mathbf{x}} \|f(\mathbf{x}) - y\|^2 \tag{2.2.7}$$

By minimizing a suitable cost function, an optimal set of parameters can be learned by approximating an arbitrary function. This optimization problem is commonly solved by the algorithm of gradient descent, well known from the field of machine learning. The general idea behind gradient descent is to iteratively update a set parameter in order to minimize an objective function. The direction of the update is given by the gradient of the objective function ∇C. Furthermore, a small positive parameter η called learning rate is introduced. The learning rate scales the changes made at each iteration. A small value of η results in small steps and a high training time, while a big value could result in an overshoot increasing the value of the cost function.

Equation (2.2.8) defines the update rule for an arbitrary set of parameters $\mathbf{v}$ at each iteration [Nie15].

$$\mathbf{v} \rightarrow \mathbf{v}' = \mathbf{v} - \eta \nabla C \tag{2.2.8}$$

In case of an artificial neural network the parameter vector $\mathbf{v}$ consist of the weights w and biases b of the network. Equation (2.2.8) can be used as an update rule to find an optimal set of weights and biases. Following this basic formulation of the gradient descent algorithm, the gradient is calculated and averaged over the whole training set. Computing the gradient for a large training set might require big computational resources for a complex network. Considering only a smaller subset of training images can speed up the training procedure while still finding a good set of parameters. In the case of stochastic gradient descent only a single training example is considered at each iteration. Batch gradient descent compromises between taking the whole training set and a single example, by averaging over a small batch of training examples [Nie15]. All these versions of gradient descent require a fast and efficient computation of the gradient for a given parameter set. As proposed by David Rumelhart, Geoffrey Hinton, and Ronald Williams in 1986 [RHW86], the backpropagation algorithm offers an efficient computation strategy for neural networks. The underlying principle of the algorithm is to find an expression for each partial derivative $\partial C/\partial w_k$ and $\partial C/\partial b_l$. In a first step, the output of the network is computed for a given input sample following the forward path of the network. Based on the output, the corresponding error is determined, which is then backpropagated. Following this procedure, the partial derivatives can be calculated layer-wise and allow for an efficient training by gradient descent.

2.3 CONVOLUTIONAL NEURAL NETWORKS

As discussed in section 2.2, an ANN consists of multiple layers, where each layer can be considered the combination of multiple neurons. The combination of different types of layers determines the networks' architecture and properties. Especially Convolutional Neural Networks (CNN), first proposed by [FM82], have been proven to be a powerful computational model. Due to good performances and flexibility, CNNs have become the predominant model in pattern recognition and computer vision. The following section gives a brief overview of the layers typically used in state-of-the-art CNNs.

2.3.1 *Convolutional Layer*

Convolutional layers were designed for networks which work with images as input data. An image can be represented by a volume of data points, with its width and height corresponding to the image's dimensions. The depth of the volume depends on the pixel representation and is typically one, in case of gray scaled images, or three for RGB images.

Consider the task of classifying an image. Image classification is typically based on finding specific features in the input image. A neuron designed to identify such features only requires the pixel data from an enclosing patch of the image. Therefore, finding a feature can be realized by evaluating a set of subvolumes of the input data. For each subvolume the neuron can use the same weights, since the characteristics of a feature are independent of its spatial position.

A layer that implements the previously discussed characteristics can be derived from the convolution operation [GBC17]. Consider two arbitrary continuous functions x and w. For an integer valued time index t, the discrete convolution is defined as:

$$f(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{2.3.1}$$

In the context of CNNs, x is often called the input and w is referred to as the kernel. The output $f(t)$ is typically named feature map. To implement a convolution operation with an input image $\mathbf{I}$, Equation (2.3.1) has to be generalized to two dimensions. Therefore, a two-dimensional kernel $\mathbf{K}$ is used:

$$F(i,j) = (\mathbf{K} * \mathbf{I})(i,j) = \sum_{m} \sum_{n} \mathbf{I}(i,j)\mathbf{K}(i-m, j-n) \tag{2.3.2}$$

Figure 2.3.1 shows an example of a convolution of an image $\mathbf{I}$ (size 3x3) and a 2x2 kernel. The output is calculated by the dot product between a patch of the image and the kernel. The image patch, used as an input of the neuron, is called receptive field and is continuously shifted. In the example, the receptive field is shifted by one pixel at each step. This parameter is named stride s. The values of the kernel define the weights of the neuron and can be learned according to the respective task [FFJK17].

A convolutional layer defines a number of k filters with d kernels, where d is equal to the depth of the input volume. The input data is often padded by zeros enclosing the original data. Overall, a convolutional layer is entirely defined by the four hyperparameters of the number of filters k, their spatial extent f, the stride s and the amount of zero padding p [FFJK17]. Consider an input volume of size $[W_1, H_1, D_1]$.

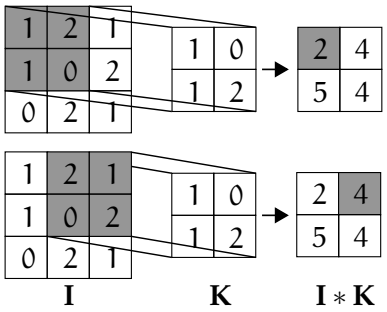


Figure 2.3.1: A convolutional layer with a single kernel **K** performs the discrete convolution operation (Equation (2.3.2)) on image **I**. The receptive field is slid over the input image with a stride of one.

A convolutional layer with corresponding hyperparameters produces the following output volume $[W_2, H_2, D_2]$:

$$W_2 = \frac{W_1 - f - 2p}{s+1} \qquad H_2 = \frac{H_1 - f - 2p}{s+1} \qquad D_2 = k \tag{2.3.3}$$

This type of layer implements the previously motivated properties. Each neuron only considers a subregion of the input data at once. Its activation depends on a total number of weights given by $f \cdot f \cdot D_1 \cdot k$. Learning these weights can be interpreted as identifying a specific feature in a region of its input data. A common choice of hyperparameters, which is used in various state-of-the-art networks like [SZ14],[HZRS16a] or [HLMW17], are $f = 3, s = 1, p = 1$. In this case, the convolutional layer does not change width and height of the data volume. The depth of the output volume is determined by the chosen number of filters. In addition to the filter weights, a convolutional layer often utilizes a learnable set of biases.

2.3.2 *Pooling Layer*

The structure of a pooling layer is similar to a convolutional layer. Each neuron processes the information of a receptive field, with a spatial extent of f. The receptive field is moved over the input data with a stride s. In contrast to a convolutional layer, a pooling layer is not connected with a set of learnable weights. Instead, a predefined operation is performed. A pooling layer essentially downsamples the input data and therefore reduces its spatial extent. For each distinctive receptive field, a single value is computed, following a predefined pooling strategy. L2-norm pooling is based on the square roots of the sum of the squared values in the receptive field. An average

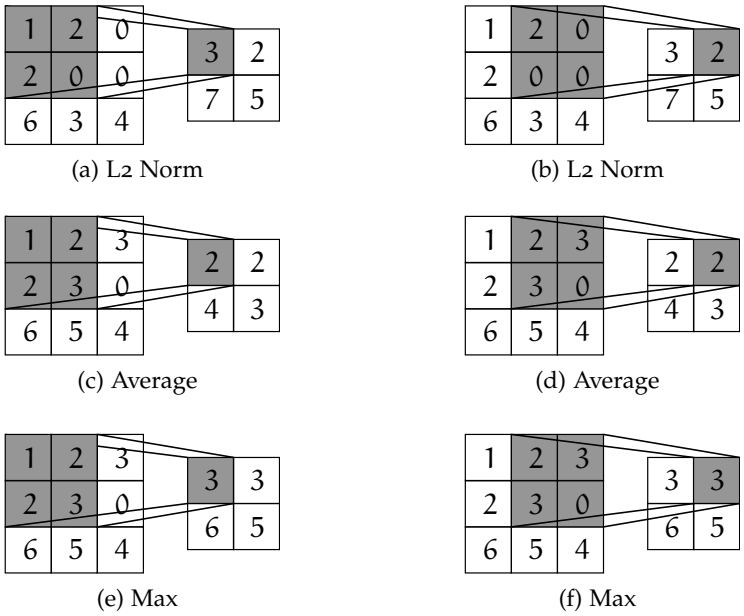


Figure 2.3.2: Example of an pooling layer with stride $s = 1$ and spatial extent $f = 2$ operating on an input with dimension 3x3. Each correspond to a different pooling strategy. After pooling the input is downsampled to a size of 2x2.

pooling layer computes the averages over each receptive field. The most commonly used pooling strategy is called max pooling. Its output is the maximum value, which has been proven to perform well in many applications and is computationally simple. Figure 2.3.2 shows an example of the different pooling strategies, for a pooling layer with stride $s = 1$ and spatial extent $f = 2$.

In general, pooling layers are inserted in between convolutional layers to reduce the spatial extent of the feature maps. This also corresponds to a reduction of parameters, while the essential information is carried on. A lower amount of parameters benefits the training procedure and makes the network less prone to overfitting. Consider an input data volume of size $[W_1, H_1, D_1]$. The dimensions of the output volume $[W_2, H_2, D_2]$, generated by a pooling layer with stride s and size f, are given by [FFJK17]:

$$W_2 = \frac{W_1 - f}{s+1} \qquad H_2 = \frac{H_1 - f}{s+1} \qquad D_2 = D_1 \tag{2.3.4}$$

2.3.3 *Fully Connected Layer*

The simplest form of a layer in a CNN is the fully connected layer. It closely follows the idea of a perceptron as introduced in section 2.2.1. Each neuron of the layer is connected to all data points of the input volume [FFJK17]. Consider an input volume of shape $[W_1, H_1, D_1]$. The input data can also be represented by a one dimensional vector of shape $[(W_1 \cdot H_1), 1]$. The neuron's output results from the dot product of the input and weight vector, with an added bias. A fully connected layer is entirely defined by the number of neurons, which can be considered a hyperparameter. The output volume, produced by a fully connected layer, is a one dimensional vector with a length equal to the number of neurons. Analogue to the concept of a single layer perceptron (see section 2.2.2), an activation function (see section 2.2.3) usually introduces a non-linearity to the layer.

2.3.4 *Softmax Activation for Classification*

CNNs are highly suitable to perform classification tasks. The use of multiple convolutional layers allows to learn a set of filters, which generate a feature representation of the input data. Classification is then usually performed by standard classifiers for example such as a Multilayer Perceptron (MLP), which consists of multiple fully-connected layers. In case of single label classification, the aim of the network is to assign the input data to its corresponding class. Therefore, a special type of activation function is typically used. In contrast to the activation functions discussed in section 2.2.3, the softmax activation does not only consider a single neuron [Bis11]. Instead, it generates a set of pseudo-probabilities based on the activation of all input neurons.

Assume that a given network shall assign some input data to one out of k classes. The last layer is fully connected and produces an output vector. To perform classification, this vector is supposed to be mapped to another vector where each value represents the probability of the input data belonging to class k. Such a mapping is realized by the softmax activation function [Bis11]. The probability p_i of the input data belonging to class i results from input vector $\mathbf{x}$ as follows:

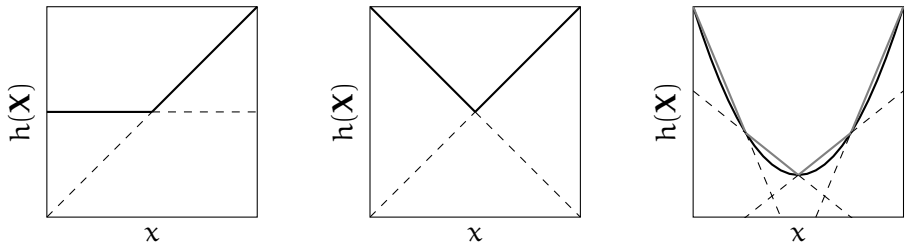
$$\varphi(x)_i = \mathrm{sm}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2.3.5}$$

2.3.5 *Maxout Layer*

The idea of maxout layers stems from the approach of improving the model averaging capabilities of dropout [GWFM+13]. The combination of dropout and maxout aim to improve network optimization and prevent overfitting [SHK+14]. As an additional advantage, maxout layers are further used for other application such as network compression [RGF17]. In general, a maxout layer constitutes a special kind of activation function and can also be considered a cross channel pooling operation. For a given input layer $\mathbf{X} = [x_0, x_1, x_2, \ldots, x_N]$ with N neurons, the following output is computed:

$$h(\mathbf{X}) = \max[x_{jk+0}, x_{jk+1}, x_{jk+2}, \ldots, x_{jk+(k-1)}] \quad \forall j \in [0, N/(k-1)] \tag{2.3.6}$$

A single maxout unit calculates the maximum across a number of k neurons. As shown by [GWFM+13], this corresponds to the implementation of an universal approximator. The combination of several neurons approximates an arbitrary complex function in a piecewise linear fashion. Figure 2.3.3 shows a two-dimensional example, where a single maxout unit approximates different activation functions. Even though the example only visualizes the behavior for one dimensional inputs, it can be generalized to multiple dimensions and arbitrary complex convex functions. Based on this approximation capability, multiple neurons are combined to a single, complex neuron.



(a) ReLU (b) Absolute Value (c) Quadratic

Figure 2.3.3: Example of a maxout unit as an universal approximator. By taking the maximum of two inputs, a rectified linear (subfigure a) and an absolute value rectifier activation function (subfigure b) are implemented. Subfigure (c) shows the approximation of a quadratic function by combining four one dimensional inputs [GWFM+13].

3 RELATED WORK

This chapter presents an overview on the established architectures and methods, which inspired the proposed method. Since the field of word spotting is strongly influenced by the application of CNNs, this work will investigate the use of densely connected networks for word spotting. This special design paradigm is motivated by and closely related to residual networks. Both network architectures achieve highly competetive results and are introduced in detail in section 3.1. Section 3.2 covers two established methods which successfully use CNNs to perform the task of word spotting.

3.1 DEEP NETWORK ARCHITECTURES

Driven by the success of neural networks in machine learning and computer vision, exploring different network architectures has been of special interest in research. Improving the performance of convolutional neural networks traditionally corresponded to an increase in the number of layers in a network. LeNet5 [LBBH98], which was one of the first networks used for document recognition, employed only 5 convolutional layers. The popular VGGNet [SZ14] used 19 convolutional layers and improved several benchmarks in image classification. This trend shows that increasing the depth of a network often benefits the performance. However, simply adding more layers can complicate optimization and thus hinders efficient application. This has been shown by investigations of deeper networks which was allowed by the increase in computational power. Nonetheless, several networks as Highway Networks [SGS15], Residual Networks (ResNet) [HZRS16a] or Densely Connected Convolutional Networks (DenseNet) [HLMW17] efficiently employed more than 100 layers and improved the state-of-the-art. In order to optimize and still to benefit from the extremely deep architecture, all these network share a common property. Additional paths are added to the traditional feedforward architecture to bypass layers. The following sections discuss how ResNets and DenseNets utilize these skip-connection in a systematic pattern and thereby achieves cutting-edge performances. Even though both connectivity patterns are quite similar, their approach and interpretation differ.

3.1.1 *Residual Networks*

As discussed in [SGS15] and [HZRS16a], a new problem emerges when optimizing increasingly deep networks. Adding layers to a traditional feedforward network leads to an increase in performance that saturates at some point. Increasing the network's depth beyond this point results in a significant drop in accuracy. [HZRS16a] argues that this degradation is not explained by overfitting, since accuracies also degrade for training. It is argued that the deeper model is still more powerful, but harder to optimize. Therefore, the shallower model outperforms the deeper network.

Residual Networks address this problem by the approach of deep residual learning. Consider a traditional feedforward network. One or multiple layers can be summarized as a non-linear mapping $H(\mathbf{x})$ of some input data $\mathbf{x}$. Optimizing the network corresponds to finding a set of weights, such that the function $H(\mathbf{x})$ approximates the underlying mapping $F(\mathbf{x})$ determined by the task. Even though it is still an open question in research [MPCB14], it is assumed that multiple layers can asymptotically approximate arbitrary complex functions. Following this assumption, [HZRS16a] proposed to not approximate the underlying mapping directly. Instead, the layers shall approximate the residual function $F(\mathbf{x}) - \mathbf{x}$.

Consider a feedforward network with ℓ indexing each layer. The output of a layer $\mathbf{x}_\ell$ is given by $\mathbf{x}_\ell = H(\mathbf{x}_{\ell-1})$. Residual learning is then introduced by adding identity connections to the network by bypassing one or multiple layers. Assume that only a single layer is skipped by the identity path. The output of layer ℓ is given by:

$$\mathbf{x}_\ell = H(\mathbf{x}_{\ell-1}) + \mathbf{x}_{\ell-1} \approx F(\mathbf{x}_{\ell-1}) \tag{3.1.1}$$

with

$$H(\mathbf{x}_{\ell-1}) \approx F(\mathbf{x}_{\ell-1}) - \mathbf{x}_{\ell-1} \tag{3.1.2}$$

The added paths do not increase the number of parameters nor the computational complexity. Nonetheless, the experiments presented in [HZRS16a] show that deep networks benefit form the deep residual learning approach. The authors argue that this is due to the better preconditioning of the optimization problem. A layer initialized with weights equal to zero constitutes an identity mapping due to the additional, bypassing path. Hence, the identity mapping is learned more easily. Further experiments showed that the residual functions of a deep network usually have small responses. Therefore, one can conclude that the target functions of the weight layers are closer to an identity- than to a zero-mapping, which is an explanation for the improved optimization behavior.

The proposed architecture achieved highly competitive results and outperforms other more shallow networks. Introducing identity paths enables an efficient optimization of increasingly deep architectures. It has been shown that the network's performance often benefits from deeper structures. [HZRS16a] also investigated extremely deep residual networks with more than a thousand layers. The experiments indicate that such a deep network may be optimized efficiently, but still might be too large for a given task, resulting in problems such as overfitting.

3.1.2 *Densely Connected Networks*

Inspired by networks like Highway Networks [SGS15] or ResNets [HZRS16a], the concept of densely connected networks aims to further exploit the application of identity paths. [HLMW17] proposes a simple connectivity pattern, based on the insight that the heavy use of skip-connections benefits optimization and performance of a convolutional network. Densely connected networks employ an identity path to each layer. The probably most influential difference between the concepts of residual learning and dense connectivity, lies in how each architecture combines the layer's output and identity. Residual learning is implemented by the summation of output and identity function (see Equation (3.1.1)). In a densely connected network the summation is replaced by the concatenation operation.

Analogue to section 3.1.1, consider a feedforward network with a layer index ℓ. Let $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_\ell$ denote the feature maps produced by each layer. By concatenating identity and output of each layer, the input of the following layer includes the input of the preceding layer. Following this design pattern, the input of layer ℓ consist of the concatenation of feature maps, generated by layers 0 up to $\ell - 1$. This tensor is denoted by $[\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]$. Therefore, the output of a layer computing the non-linear function $H(\cdot)$ is given by:

$$\mathbf{x}_\ell = H([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}]) \tag{3.1.3}$$

A direct consequence of this design pattern is that each layer has access to the feature maps of all preceding layers. The DenseNets proposed in [HLMW17] employ the composition of three operations as weight layers. First batch normalization is used as a regularizer. Hence, mean and variance of the mini-batch are used for scaling and shifting the layer's inputs. Using this normalization improves the training and optimization behavior of the network [IS15]. The batch normalization operation is then followed by a ReLU and a convolutional layer with kernel size of 3×3. This composition stems from [HZRS16b], where it is employed efficiently for residual learning.

See figure 3.1.1 for an example of the described design pattern, with batch normalization (BN), ReLU and the convolution operation as layers. Even though the dense connectivity pattern is implemented by introducing skip-connections to each individual layer, the structure can be interpreted as follows. By concatenation, each layer has access to all previously generated feature maps. This corresponds to a direct connection from each layer to all preceding layers. The network in figure 3.1.1 consists of 4 densely connected layers. The input data $\mathbf{x}_0$ has 6 channels and passes through the network. Due to dense connectivity, the input is also connected to all other layers. Layer H_1 computes 4 feature maps from the input data x_0, which are again connected to all succeeding layers. At the end of the example network, the so called transition layer receives the concatenation of the original data and all generated feature maps, $[\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4]$. Since each weight layer adds a total of 4 feature maps to the overall concatenation tensor, the output has a depth of 22.

The number of feature maps added by each layer is a hyperparameter which significantly determines the network's properties. Following the terminology of [HLMW17], we refer to this parameter as growth rate k. The number of feature maps used as an input to a layer increases towards the end of the network. Layer ℓ has an input with a number of $k_0 + k(\ell - 1)$ channels, where k_0 denotes the channels of the original input data. Experiments have shown that even small growth rates achieve competitive performances in image classification. Considering the convolutional layers of the network, a small growth rate corresponds to very narrow layers, generating only a small set of feature maps.

The use of convolutional layers with kernel sizes 3×3 does not change the size of the input. Therefore, the concatenation operation in Equation (3.1.3) is well defined. Neural networks for typical tasks like image classification usually employ pooling layers to reduce the size of feature maps towards the end of a network. This also results in a reduction of parameters, since succeeding fully-connected layers require less connections. DenseNets are organized in blocks to employ pooling and still avoid the concatenation of differently sized feature maps. Each block of the network follows the dense connectivity pattern with constant feature map sizes. Two blocks are connected via transition layers. By using pooling layers only in the transitions between dense blocks, the dense connectivity pattern can be implemented straightforwardly.

[HLMW17] presented a set of experiments that investigate the significance of individual feature maps. The results indicated that each layer spreads its weights over numerous preceding feature maps. Regarding transition and classification layers, it was observed that these layers tend to make stronger use of feature maps created late in the network. This insight motivates the introduction of a compression stage. At the transition between two blocks, the number of feature maps is reduced by a

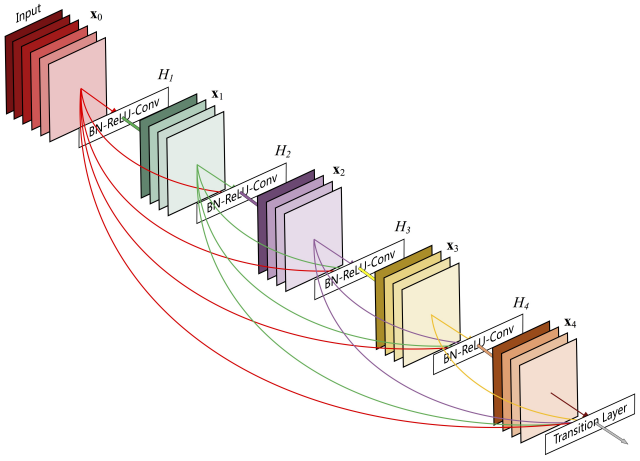


Figure 3.1.1: Densely connected convolutional network with 4 weight layers and a growth rate of $k = 4$. Image taken from [HLMW17].

compression factor θ. Since the following block is highly focused on the complex feature maps generated at the end of the preceding block, the loss of information is considerably low. Therefore, reducing the number of feature maps hardly influences the performance of the network, while model compactness is significantly improved. DenseNets implement the compression approach by employing a convolutional layer with kernel size 1×1 as an additional transition layer. Assume that the dense block preceding the compression layer produces a number of m feature maps. The number of output channels of the convolutional layer is then set to θm with $0 < \theta \leqslant 1$. Compressing a network corresponds to a compression factor of $\theta < 1$.

Inspired by its successful application in convolutional [SVI+16] and especially residual networks [HZRS16a], the use of bottleneck layers were also investigated for densely connected networks. Before each weight layer (BN, ReLU, 3×3 convolutional layer) a bottleneck layer can be introduced. This bottleneck layer also uses the composition of BN, ReLU and convolutional layer, but only employs a kernel size of 1×1. A single skip-connection, implemented as an identity path, bypasses weight and bottleneck layer. The number of output feature maps of the bottleneck layer is set to $4k$. Thereby, the number of input feature maps of the weight layer is reduced. Especially towards the end of the network, where a high number of feature maps have been produced, this design significantly reduces the number of parameters and improves computational efficiency.

[HLMW17] investigated the proposed design pattern on different benchmark tasks in the field of image classification. Depending on the datasets and especially the image's dimension, a reasonable number of pooling layers and densely connected blocks needs to be defined. The authors evaluated networks with three and four dense blocks, different growth rates and introduced the bottleneck and compression layers. For all considered datasets, the DenseNet architecture outperforms former state-of-the art networks. Furthermore, the presented DenseNets have been shown to be highly parameter efficient. Especially, networks using bottleneck and compression layers were able to achieve competitive results with a significantly lower number of parameters.

As argued by [HLMW17], the improved model compactness compared to other architectures may mainly result from the reuse of features. The dense connectivity pattern is comparable to building a "collective knowledge". By having a connection to all preceding feature maps, each layer has access to all features learned in the earlier stages of the network. Based on this knowledge a small number of higher-level and more complex features are learned and added to the "collective knowledge". While a layer in a traditional network has to either drop or relearn low-level features, this information is preserved in a densely connected network. Similar to ResNets, the use of identity connections improves the gradient and information flow, which benefits the optimization behavior. In general, the experiments in [HLMW17] show that the simple dense connectivity pattern results in a highly compact and powerful model.

3.2 WORD SPOTTING WITH CONVOLUTIONAL NETWORKS

3.2.1 *PHOCNet*

Word spotting systems based on CNNs got increasingly popular in recent years. This section discusses the method proposed by [SF16], where a specifically designed CNN named PHOCNet is used for word spotting. The CNN learns a mapping between a word image and its attribute representation. By applying an attribute representation which can be easily derived from a string, the method is suitable for *query-by-example* and *query-by-string* word spotting. The PHOCNet outperforms other methods not based on CNNs and yields state-of-the-art performances on different datasets for segmentation-based word-spotting [PZG+16].

Attribute representation

The idea of using an attribute representation in the context of word spotting was first proposed by [AGFV14]. As argued by the authors, traditional word spotting methods based on feature representations and sequential models do not share any information between similar words. For example, two words which only differ in a single character, require individual discriminative models, even though their images are visually similar. This shared information is exploited by attribute based approaches, which have been successfully applied for image classification and retrieval [FEHF09]. Two objects or word images might share several attributes without belonging to the same class. By sharing information, a more discriminative representation can be learned. Furthermore, this approach is extremely suitable for word spotting, since an attribute representation can also be generated for a word not included in the training vocabulary. The authors propose a binary word embedding called pyramidal histogram of characters (PHOC), also used by the PHOCNet.

A binary PHOC vector encodes a string by using multiple histograms of its characters with respect to a considered alphabet. Character occurrences can be represented by a vector, corresponding to the considered alphabet. In case of the English alphabet plus digits, the respective histogram consists of 36 dimensions. If a character is present in the word, the corresponding histogram value is set to one. In terms of attributes, this represents if a word contains a specific character or not. Since a single histogram is not word-discriminative, multiple histograms are generated in a pyramidal fashion. These so called levels consider different splits of a string. On level two, the string is split into two parts and for each of them a binary histogram is generated. This procedure is continued for higher numbers of splits and histograms. A common choice of levels, used in [AGFV14] and [SF16], is the use of levels 2, 3, 4 and 5. This results in a PHOC vector with 504 dimensions, considering digits and the English alphabet. Furthermore, the vector is extended by the 50 most common English bigrams at level 2. Figure 3.2.1 shows an example, where a PHOC vector for the word "place" is generated, using levels 1, 2, 3 without any bigrams. The resulting attribute representation encodes whether a character is present in a specific split of the word. This leads to a discriminative representation, which allows to share information on attributes between images from different word classes.

Additionally, [SF17] investigated other attribute representations. The authors conclude that other word string embeddings, also based on character occurrence and position, perform equally well, but not superior to PHOC vectors.

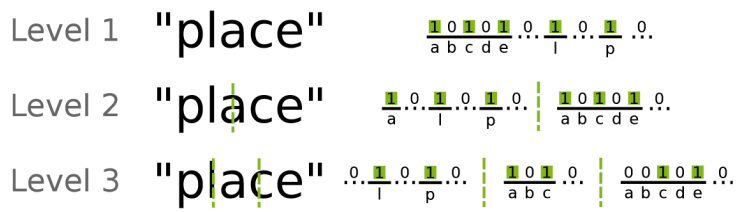


Figure 3.2.1: PHOC vector for the word "place". Image taken from [SF16].

Architecture

The general PHOCNet architecture is heavily inspired by the convolutional network VGGNet, proposed in [SZ14]. As shown by the experiments with VGGNet, convolutional layers with small receptive fields achieve highly competitive results. The design in [SZ14] only uses convolutional layers with kernel sizes of 3×3. This paradigm is adopted for the PHOCNet architecture. All convolutional layers use kernels of size 3×3 in combination with a ReLU activation. This corresponds to a regularization of the filter kernels, which prevents overfitting. In general, the number of filters increases towards the end of the network. As argued in [SZ14] and [SF16], this results in a small number of low level features for smaller receptive fields. By an increased number of filters in the higher layers of the network, a bigger number of abstract, more complex features are learned. The PHOCNet uses two max pooling layers with kernel sizes of 2×2 and strides of 2. Therefore, the size of the feature maps is cut in half after two and after four convolutional layers.

While many networks designed for image classification assume input images of equal dimensions, the PHOCNet follows a different strategy. Since the aspect ratios of word images are highly dissimilar, cropping or anisotropical rescaling might severely influence the network's performance. To cope with different image sizes, the PHOCNet uses a Spatial Pyramid Pooling Layer (SPP) [HZRS15]. In contrast to convolutional and pooling layers, fully-connected layers cannot process feature maps of varying sizes. Therefore, a SPP layer generates an output of constant size independent of the dimensions of the input feature map. This is done in a pyramidal manner. For a SPP layer with three levels as used in the PHOCNet, each feature map is divided into multiple spatial bins. On level one the input feature map is considered to be a single bin and the max pooling operation is performed. On the second level the feature map is split along each dimension, resulting in 4 separate bins. For each of the 4 bins a single output is generated by max pooling. This procedure continues for an increasing number of levels. Each individual bin is divided along its spatial

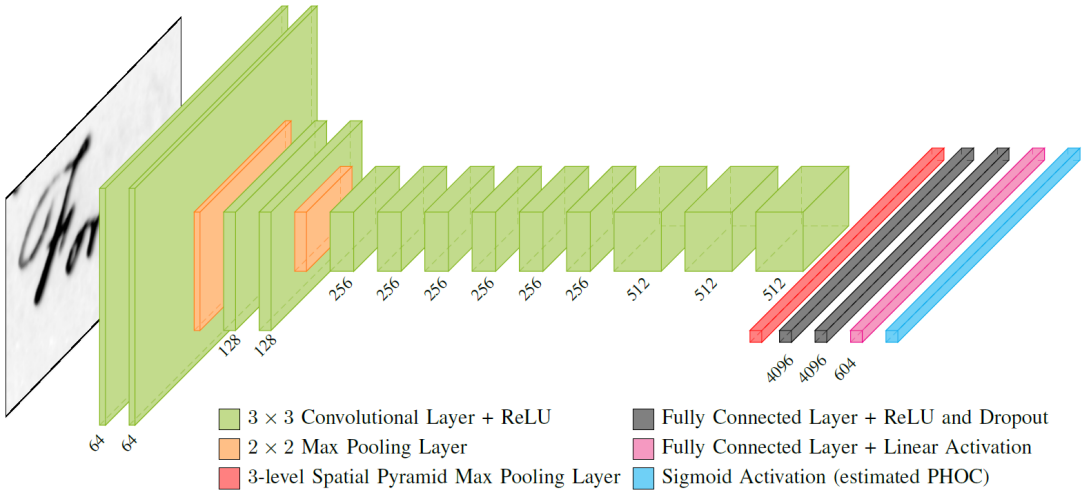


Figure 3.2.2: The figure shows the architecture of the SPP-PHOCNet. The number under each layer corresponds to the number of filters for convolutional layers or number of neurons for fully-connected layers. Image taken from [SF16].

dimensions, resulting in 16 bins on level three. Following this procedure for multiple levels, a fixed number of outputs is generated. The SPP layer allows to use input images of varying sizes by mapping the resulting feature maps to a feature vector of constant size. Furthermore, this allows the application of a standard classification strategy, which usually assumes inputs of fixed sizes. In another contribution, the PHOCNet was enhanced by replacing the SPP layer with a new layer call Temporal Pyramid Pooling layer (TPP) [SF17]. As argued by the authors, the subdivision along the horizontal axis is more important for word images. Therefore, the TPP layer only divides the feature maps along the horizontal axis. Here, a level determines the number of horizontal bins. In [SF17], a five level TPP layer is used. It has been shown that this newly introduced layer either achieves comparable accuracies or beats the state-of-the-art.

The pyramid pooling layer connects the convolutional part of the network to a multi layer perceptron (MLP) with two hidden layers of size 4096. The process of learning a PHOC vector corresponds to a multi-label classification task. Instead of using a softmax activation function to calculate the pseudo-probability that the input belongs to a specific class, the sigmoid activation function is applied to each output. In terms of learning an attribute representation, the output of the sigmoid activations can be considered as the pseudo probability of the attribute being present in the input image.

The overall architecture is depicted in figure 3.2.2. By training the network in an end-to-end fashion, it is able to learn a mapping between word image and PHOC vector. As discussed in [SF17] and [PZG+16], the PHOCNet achieves state-of-the-art accuracies on several benchmark experiments for segmentation-based word spotting using standard evaluation protocols.

3.2.2 *Triplet CNN*

The method proposed in [WB16] also considers the task of segmentation-based QbE and QbS word spotting. The approach is based on three identical CNNs that share their weights. Training of this so called Triplet CNN is conducted simultaneously by means of a specific loss function. Each string of the network processes a different word image and it is supposed to generate a distinctive feature vector. The aim of the training procedure is to map two images of the same class on two feature vectors which are close to each other. Therefore, the network considers triplets of word images during training. Each triplet T consists of two images from the same class (positive samples p_1, p_2) and another word image from a different class (negative sample n). The desired mapping is then learned by minimizing the SoftPN loss function [BJTM16]:

$$\begin{aligned} L(T) = &\left(\frac{e^{d(p_1,p_2)}}{e^{\min(d(p_1,n),d(p_2,n))} + e^{d(p_1,p_2)}}\right)^2 + \\ &\left(\frac{e^{\min(d(p_1,n),d(p_2,n))}}{e^{\min(d(p_1,n),d(p_2,n))} + e^{d(p_1,p_2)}} - 1\right)^2 \end{aligned} \tag{3.2.1}$$

By minimizing the loss, the distance between the two positive samples $d(p_1, p_2)$ is also minimized while the smallest distance between positive and negative sample $\min(d(p_1, n), d(p_1, n))$ is maximized. A triplet of three residual networks is trained on resized word images of size 60×160 pixels using a similar augmentation strategy as [SF16] and [AGFV14]. After training, a single string of the network can be used to generate a distinctive feature vector from a word image.

Word spotting is then be performed by mapping the space of feature vectors onto a suitable word embedding space. The work of [WB16] investigates different classic string based embeddings such as PHOC vectors, but also proposes a semantic embedding space. A two layer fully connected network is used to learn the mapping between feature vector and word embedding. First, one string of the triplet CNN is used to generate the feature vectors of the training images. In a second step, this set of feature vectors serves as training samples to learn a suitable mapping.

The Triplet CNN method achieves competitive results on various datasets, but it is generally outperformed by the TPP-PHOCNet [SF17]. Even though both approaches share many similarities, it seems favorable to directly learn the mapping between word images and word embedding compared to the triplet based approach. Despite their architectural differences, both CNN based approaches seem clearly superior to non-CNN based methods.

4

METHOD

As discussed in section 2.1, convolutional neural networks has become the state-of-the-art approach in word spotting. Considering recent developments in research on network architectures, deep models and especially architectures incorporating identity paths show promising results. Inspired by the cutting edge performances in the field of image classification, this work investigates the use of densely connected networks for word spotting. The general approach is similar to [SF15]. A CNN is used to learn a mapping between word image and the respective attribute representation. This work investigates how different architectures, which at least partly follow the dense connectivity pattern, perform on different benchmark datasets for word spotting. Section 4.1 discusses the proposed architectures and the general CNN based approach. Since most proposed networks have a large number of parameters, they are prone to overfitting. Therefore, several regularization measures are used for training the networks which will be presented in section 4.2.

4.1 WORD SPOTTING WITH DENSE CONVOLUTIONAL NETWORKS

The proposed approach uses a densely connected network to learn an attribute representation of a word image. Based on this representation, word images can be ranked with respect to their similarity to a given query string or word image. Given a dataset with already segmented word images and annotated training data, word spotting is then performed by returning those images most similar to the query with respect to their attribute representation.

As argued by [SF17], attribute embeddings only based on character positions and occurrence give similar results, despite their different characteristics. Inspired by [SF16], this work uses the PHOC representation as an attribute vector. The PHOC vector consists of histograms for splits 2, 3, 4 and 5 following the approach discussed in section 3.2.1. Besides individual character occurrences, the PHOC vector is extended by the 50 most common bigrams on split 2. Given an annotated set of training data, this representation can be straightforwardly derived from a word image's label. The resulting training data consists of word images and their corresponding binary PHOC representation.

Learning the mapping between image and PHOC representation follows the same approach as [SF16]. Despite their architectural differences, all networks employ a multi layer perceptron (MLP) and sigmoid activations. The MLP uses two fully-connected hidden layers of size 4096 to connect the preceding layer to a fully-connected layer corresponding to the size of the PHOC vector. Sigmoid activations are then employed to the last fully-connected layer and provide the network's output. The output of a sigmoid activation represents the pseudo-probability of whether the corresponding attribute is present in the word image. Regarding the used PHOC representation, the network generates a vector of pseudo-probability indicating whether a specific character is present in a specific split of the word.

While the choice of attribute embedding and classification layers is analogue to [SF16], this work investigates the use of a different design for the convolutional part of the network. The original PHOCNet presented in section 3.2.1 employs 13 convolutional layers, two max pooling layers and a pyramidal pooling layer which connects convolutional part and MLP. The proposed method uses the dense connectivity pattern (see section 3.1.2). The following sections describe the densely connected architectures used in the latter experiments.

4.1.1 *DenseNet-121*

The concept of dense connectivity originally stems from the field of image classification. [HLMW17] proposes multiple network architecture, specifically designed for the ImageNet dataset. This dataset consists of 1.2 million training images of size 224×224 from 1000 different classes and it has become one of the benchmark tasks in image classification.

As a first approach, one of the network architectures, which was successfully employed on the ImageNet dataset, shall be adapted for word spotting. Therefore, the proposed network makes use of the same design as the DenseNet-121 presented in [HLMW17]. DenseNet-121 successfully employs the dense connectivity pattern for image classification and outperforms other popular networks as VGG-16 [SZ14] or ResNet-34 [HZRS16a].

The proposed network architecture uses four densely connected blocks. In general, the network uses the composition of batch normalization (BN), ReLU and convolutional layer as weight layers. Before the first block, a 7×7 convolutional layer and a 3×3 max pooling layer, both with a stride of 2, are used to reduce the size of the input images. Each block follows the dense connectivity pattern by bypassing the combination of a bottleneck and convolutional layer with an identity path. The convolutional layers

employ a kernel size of 1×1 in case of bottleneck layers and 3×3 otherwise. The number of output channels is mainly determined by the networks growth rate of $k = 32$. The first convolutional layer with kernel size 7×7 produces $2k$ feature maps. For bottleneck layers, the number of outputs channels is set to $4k$. Transition layers, which consist of a compression (BN, ReLU, 1×1 convolution) and a 2×2 average pooling layer, connect the dense blocks. Each compression layer drops half of the

Layer	**Output Size**	**DenseNet-121**	**PHOC-DenseNet-121**
Convolution	112×112	7×7 conv, stride 2	
Pooling	56×56	3×3 max pool, stride 2	
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	
Transition Layer (1)	56×56	1×1 conv, $\theta = 0.5$	
	28×28	2×2 ave pool, stride 2	
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	
Transition Layer (2)	28×28	1×1 conv, $\theta = 0.5$	
	14×14	2×2 ave pool, stride 2	
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	
Transition Layer (3)	14×14	1×1 conv, $\theta = 0.5$	
	7×7	2×2 ave pool, stride 2	
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	
Pooling	1×1	global ave pool	
Classification Layer		1000-d fc, Softmax	4096-d fc, ReLU, Dropout
			4096-d fc, ReLU, Dropout
			604-d fc, Sigmoid

Table 4.1.1: DenseNet-121 and the proposed architecture for word spotting based on PHOC vectors. The layers denoted as "conv" represent the composition of BN, ReLU and convolutional layer. The growth rate for both networks is $k = 32$. DenseNet-121 architecture taken from [HLMW17].

input feature maps, and therefore applies a compression factor of $\theta = 0.5$. After the last dense block a global average pooling layer is used to create a feature vector.

The original DenseNet-121 employs a fully-connected layer of size 1000 in combination with softmax activation to perform single label classification. In order to use the DenseNet-121 architecture for word spotting, the following changes are applied. The single fully-connected layer is replaced by the MLP discussed previously. Since learning the PHOC representation is not a single label classification task, softmax activation is replaced by sigmoid activation to generate the respective pseudo probabilities.

Table 4.1.1 shows the architecture of DenseNet-121 and the changes introduced for word spotting. A PHOC representation for the English alphabet plus digits is assumed, resulting in a vector of size 604. Additionally, the size of the feature maps in each block are given. Note, that the size is given with respect to an input image of size 224×224. In case of common word spotting datasets, images are usually of different sizes. The presented architecture also allows the use images of varying sizes, since the global average pooling layer generates a feature vector of fixed size. The global pooling fulfills a similar function as the pyramidal pooling layers used in [SF16] and [SF17]. In other words, the global pooling layer corresponds to a pyramidal pooling layer with only a single level.

4.1.2 *PHOC-DenseNet*

The architecture proposed in section 4.1.1 was specifically designed for the ImageNet dataset. [HLMW17] investigates a different approach for other benchmark datasets. This is mainly due to the differently sized images. While the ImageNet dataset consists of images of size 224×224, other benchmark datasets as CIFAR or SVHN only use images of size 32×32. Based on the sizes of its input, a suitable number of pooling layers is employed in the network. In case of CIFAR and SVHN, [HLMW17] proposes a general network architecture, using three dense blocks and different growth rates.

Consider the input data used for a word spotting task. Word images show highly varying aspect ratios, since the depicted word can contain numerous characters or be as short as a single one. Therefore, popular datasets used for word spotting usually contain images of varying sizes. To define a suitable pooling strategy, the proposed network architectures took inspiration from the PHOCNet, presented in section 3.2.1. The PHOCNet employs two traditional pooling layers early in the network and a pyramidal pooling layer right before its MLP. The proposed network architectures aims at replicating this pooling strategy, while following the dense connectivity pattern.

Based on the number of pooling layers, the following basic architectural considerations are made. The convolutional part of the network consists of either two or three densely connected blocks. In case of three dense blocks, the network contains two transition layers, each including a pooling layer. Since a network with two dense blocks only employs a single transition layer, another pooling layer is added before entering the first block. Furthermore, the transition layers include a 1×1 convolutional layer, which allows compression with θ. Inspired by [HLMW17], an initial convolutional layer with kernel size 3×3 is introduced at the beginning of the network, producing a fixed number of 32 output channels. After the last dense block, a temporal pyramid pooling layer with five levels produces a feature vector of fixed size. Again, the combination of MLP and sigmoid activation is used as classification layers. Each block

Layer	**PHOC-DenseNet-b2**	**PHOC-DenseNet-b3**
Convolution	3×3 conv, stride 1	
Pooling	3×3 max pool, stride 2	
Dense Block (1)	$[3 \times 3 \text{ conv}] \times b_1$	
Transition Layer (1)	1×1 conv, θ	
	2×2 ave pool, stride 2	
Dense Block (2)	$[3 \times 3 \text{ conv}] \times b_2$	
Transition Layer (2)		1×1 conv, θ
		2×2 ave pool, stride 2
Dense Block (3)		$[3 \times 3 \text{ conv}] \times b_3$
Pooling	5 level TPP	
Maxout	maxout K	
Classification Layer	4096-d fc, ReLU, Dropout	
	4096-d fc, ReLU, Dropout	
	604-d fc, Sigmoid	

Table 4.1.2: Densely connected networks for word spotting. The networks either employ two (PHOC-DenseNet-b2) or three densely connected blocks (PHOC-DenseNet-b3). Layers denoted as "conv" represent the composition of BN, ReLU and convolutional layer. The number of output channels of each weight layer is determined by the growth rate k.

is composed of densely connected weight layers, which consist of the composition of BN, ReLU and 3×3 convolutional layer.

The experiments in this work investigate different configurations of this densely connected architecture. A network configuration is determined by the number of weight layers in each dense block. The vectors $c_2 = [b_1, b_2]$ and $c_3 = [b_1, b_2, b_3]$ denote the configuration for networks using two or three densely connected blocks. Different growth rates are applied to investigate their influence on the network's performance. Additionally, compressed and uncompressed networks are evaluated. In the latter experiments, compressed means that the transition layers of the network employ a compression factor of $\theta = 0.5$. An uncompressed network corresponds to $\theta = 1$.

Motivated by the concept of compression, the proposed architectures uses a maxout layer between TPP layer and MLP. Assuming that the last block and the pyramidal pooling layer introduce redundancies into the network, it might be reasonable to combine multiple neurons. This is especially beneficial, since the number of parameters of the MLP grows with the number of feature maps generated by the network. Due to the dense connectivity paradigm, the number of feature maps increases with the depth of the network, which might lead to overparameterization when combined with the MLP. The maxout layer counters this effect by combining a number of K neurons, where $K = 1$ corresponds to a plain network without a maxout layer. Table 4.1.2 shows an overview of the proposed architecture.

4.1.3 *Hybrid Approach*

The third approach, which is referred to as Hybrid-PHOCNet, only partly applies the concept of dense connectivity. The PHOCNet provides a successful architecture and pooling strategy that achieves high accuracies on typical benchmark sets. Both pooling layers of the PHOCNet are employed relatively early in the network. Most convolutional layers are located after the second pooling layer. The proposed architecture, which combines dense connectivity with the successful PHOCNet, replaces the convolutional layers between second pooling layer and TPP layer by a single densely connected block. By using the same initial convolutional and pooling layers, it is ensured that the pooling strategy is suitable regarding the considered datasets. Furthermore, the experiments conducted with the PHOCNet showed that the proposed structure is able to learn a sufficient number of low-level features. Based on the feature-maps provided by the initial layers, the proposed architecture employs a dense

block. Given a suitable number of low-level features, the dense connectivity scheme is employed to learn features of increasing complexity.

The design of the dense block is analogue to section 4.1.2. Each weight layer uses the composition of BN, ReLU and 3×3 convolutional layer and it is bypassed by an identity path. [HLMW17] uses compression layers only in combination with pooling. While compression might also be a useful method to remove redundancies in the proposed architecture, no additional pooling layers are introduced to the network. The compressed variant of the proposed architecture uses a compression layer in the middle of its dense block, without introducing an additional pooling layer. The architecture combining the PHOCNet and dense connectivity is depicted in table 4.1.3. Latter experiments investigate the use of different growth rates and depth. Again, the same multi-label classification strategy is used to learn the PHOC attribute representation.

Layer	**Hybrid-PHOCNet**	**Hybrid-PHOCNet-C**
Convolution	3×3 conv, stride 1, output channels 64	
Convolution	3×3 conv, stride 1, output channels 64	
Pooling	3×3 max pool, stride 2	
Convolution	3×3 conv, stride 1, output channels 128	
Convolution	3×3 conv, stride 1, output channels 128	
Pooling	3×3 max pool, stride 2	
Dense Block (1)	$[3 \times 3 \text{ conv}] \times b_1$	
Compression Layer		1×1 conv, $\theta = 0.5$
Dense Block (2)		$[3 \times 3 \text{ conv}] \times b_2$
Pooling	5 level TPP	
Classification Layer	4096-d fc, ReLU, Dropout	
	4096-d fc, ReLU, Dropout	
	604-d fc, Sigmoid	

Table 4.1.3: Hybrid-PHOCNet and its compressed variant. Initial four convolutional layers and first two pooling layers are designed analogue to [SF16].

4.2 REGULARIZATION

The proposed network architectures constitute a complex computational model, which incorporates a high number of parameters. Since the proposed method for word spotting requires annotated word images, the available training data is limited [SF16]. This makes the networks prone to overfitting. Common regularization techniques are used to limit the effects of an overfitting model. To guarantee comparability, the same regularization methods as presented in [SF16] are applied.

In this work, all networks employ the same set of classification layers, consisting of a MLP followed by sigmoid activation. Encouraged by its successful use in different image classification tasks [KSH12], [SZ14], the PHOCNet applies Dropout to its fully-connected layers [SF16]. Dropout is based on randomly dropping neurons, which corresponds to an activation of zero [SHK+14]. Consider a neuron which is activated for a specific input image. By applying dropout, the activation is set to zero with a certain probability. Therefore, the following layer does not always see an activated neuron for a specific input. Since Dropout introduces additional variations to the neuron activations, the network is less likely to overfit the training data. Analogue to [SF16], the proposed architectures apply a Dropout of 0.5 to the first two fully-connected layers of the MLP.

Besides Dropout, the proposed method uses data augmentation to deal with the limited amount of training data. This work applies the same augmentation strategy used in [SF16]. Additional training word images are created synthetically, based on an affine transformation of the original image. The augmented training dataset consists of $500,000$ images, which is roughly five times the number of images in the biggest dataset used in the latter experiments. The proposed augmentation strategy ensures that the training set includes an equal number of images per class. If the number of original images exceeds the number of images per class, the additional images are dropped randomly. For all other classes, additional images are generated by the following procedure.

A number of word images is sampled from the respective class. For each sample, an affine transformation is derived from the homography, which transforms a previously defined set of three relative coordinates. Analogue to [SF16], $(0.5, 0.3)$, $(0.3, 0.6)$ and $(0.6, 0.6)$ are chosen as source coordinates. The destination coordinates are obtained by multiplying each coordinate value by a random factor drawn from a uniform distribution with limits $[0.8, 1.1]$. Since the newly generated word images share the same labels as the original images, no further annotation is required. Adding additional training images and balancing the classes imposes further regularization and helps to deal with the problem of overfitting.

5 EXPERIMENTAL EVALUATION

This chapter covers the experiments conducted to evaluate the method described in chapter 4. Most of the experiments are performed for two datasets, namely the IAM Handwritten (IAM) and Botany database, which are considered to be relatively complex. The final comparison (section 5.4.5) with the state-of-the-art also includes a dataset of lower complexity with the George Washington dataset. For a detailed description of the considered datasets, see section 5.1.

Evaluation is based on a commonly used protocol, which is also applied in [AGFV14] and [SF16]. As discussed in section 5.3, performance is measured in terms of mean average precision (mAP) with respect to a set of queries. Both scenarios, *query-by-example* and *query-by-string*, are considered. For most experiments, the training setup described in section 5.2 is applied. While this setup might not be optimal for all networks, it still allows to gather comparable results for the network's performances.

In a first series of experiments, section 5.4.1 provides an evaluation of the adapted DenseNet121. As discussed in section 4.1, PHOC-DenseNet-b2 and PHOC-DenseNet-b3 only constitute a general network structure. Their actual architecture is determined by multiple hyperparameters. Section 5.4.2 investigates their influence on the network's performance. Furthermore, the influence of compression and maxout layers is evaluated. All networks employ a pooling layer right before its fully-connected layers. An evaluation of different choices of types of pooling layers is provided in section 5.4.2. Since some of the design choices of the proposed method are directly linked to the respective datasets and especially their image sizes, section 5.4.3 evaluates how resizing the word images influences the performance. An evaluation of different networks following the hybrid approach is presented in section 5.4.4.

Some concepts employed in the presented densely connected network architectures are also a reasonable approach for other networks such as the PHOCNet. Therefore, the experiments on resized images, pooling and maxout layer are also conducted for respectively adapted PHOCNet architectures. The last set of experiments (section 5.4.5) compares the proposed method to the state-of-the-art.

5.1 DATASETS

The following experiments are conducted on multiple publicly available datasets. All datasets provide a collection of word images with a corresponding annotation. The problem of segmenting the entire document page images into word images has been solved for all datasets sets. Training and evaluation is therefore solely performed on the previously segmented word images provided by each dataset.

The majority of the experiments consider the IAM Handwritten Database (IAM) and the Botany Database. Both datasets are considered relatively complex, making them suitable benchmark tasks for this work. The IAM database stems from the field of handwriting recognition and was specifically created for academic purposes [MB02]. 687 different writers copied a designated set of texts to create the original document images. Segmentation and annotation is publicly available resulting in a total of 115320 annotated word images in modern handwriting. The experiments use the official writer independent text line recognition partition. This partition defines a training and test set, which are also used in several publications such as [SF16] and [AGFV14]. Due to the high number of writers, the visual appearance of the same word might be highly different. This high intra-class variation makes the IAM database an especially challenging dataset.

The second dataset used in our experiments is the Botany database. This historic document collection considers botanical observations in British India. It was used as one benchmark task in the 2016 Keyword Spotting Competition [PZG+16]. While the actual competition also considered multiple subsets of the botany database to investigate the influence of training set sizes, the experiments in this thesis are conducted with respect to the split Train III [PZG+16]. In this case, the training set consists of 16686 word images. Furthermore, a designated set of query strings and images is provided. The corresponding test set contains 3318 word images.

For a general comparison of our method to the state-of-the-art, the George Washington (GW) dataset is considered as well. This dataset has become one of the most prominent datasets and is considered a benchmark task for word spotting. The database is based on 20 document pages of a correspondence from George Washington. Segmentation and transcription result in a total number of 4860 annotated word images. For this data set, no official training and test sets have been published. The best practice in most publications [SF16],[AGFV14], [WB16] is to follow a four fold cross validation approach. Therefore, one quarter of the word images is used as a test set, while the remaining images constitute the training set. The experiment is then ran four times, such that each time a different split functions as a test set. The average over all accuracies gives the actual performance measures.

All datasets used in the following experiments consist of images of varying sizes. Some minor modification are introduced to the different datasets to meet the requirements of the networks. Current implementation of pooling layers require a feature map of a minimal size. Therefore, all images with either a height or width of fewer than 32 pixels are resized. The analysis of the different datasets shows that the number of images significantly larger than the average is comparable low. Since memory consumption is especially critical in the context of densely connected networks, all images consisting of more than 50000 pixels are resized as well. While only a small fraction of the dataset is resized and therefore only little information is lost, the memory requirements are significantly reduced. The aspect ratios of the images is kept constant to avoid an anisotropic transformation.

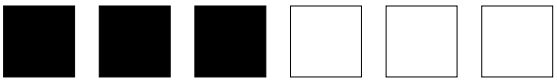
5.2 TRAINING SETUP

Since one goal of the latter experiments is to gain an insight on how different hyperparameters influence the performance of the network, most experiments follow the same training setup. The proposed training parameters might not be optimal for all networks, but still allow for a comparison of the different architectures.

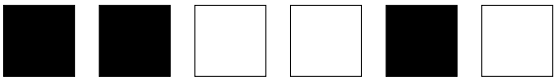
Training is performed by stochastic gradient descent and its parameters are based on [SF16]. All networks are trained with a mini-batch size of 10, momentum of 0.9, weight decay of $5 \cdot 10^{-5}$ and an initial learning rate of 10^{-4}. For all experiments except the final comparison in section 5.4.5, the training procedure terminates after 100000 iterations. This corresponds to running the training for two epochs with respect to the datasets, which were augmented according to section 4.2. After 70000 iteration the learning rate is reduced, by dividing the initial learning rate by 10.

As indicated in [SF17], the PHOCNet is able to improve its accuracy for the IAM database by training for more than 100000 iterations. Therefore, comparing the overall performance of the proposed method uses a training setup with an increased number of iterations. In this case, the network is trained for 250000 iterations, while the learning rate is divided by 10 after 100000 and 200000 iterations.

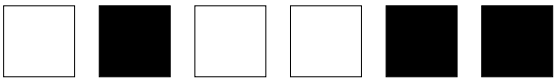
Weight initialization follows the same approach presented in [GB10]. Layer biases are initialized with zero. All other initial weights are randomly sampled from a zero-mean uniform distribution. The corresponding variance is set to $\frac{2}{n}$, where n corresponds to the number of parameters in the respective layer. Training is carried out on a single Nvidia Tesla P100 GPU with 16 Gb memory. Implementation is based on the Caffe Framework [JSD+14].



(a) Relevant items at 1,2,3.



(b) Relevant items at 1,2,5.



(c) Relevant items at 2,5,6.

Figure 5.3.1: Three example retrieval lists with six items. Items that are relevant with respect to a given query are shown in black, irrelevant items in white.

5.3 EVALUATION PROTOCOL

All experiments follow the commonly used evaluation protocol originally proposed in [AGFV14]. After training, the proposed networks approximate a function, which maps word image to the respective attribute representation. Regarding the previously discussed datasets, the task of word spotting corresponds to retrieving the subset of word images from the test set, which are relevant with respect to a designated query. First, the attribute representation of the query is derived. In case of PHOC vectors, the attribute vector can be directly derived from a string. While this is suitable for *query-by-string*, *query-by-example* requires to derive the attribute representation from a query image. Here, the previously trained network is used.

Word spotting is then performed by ranking all images in the test set with respect to the query, represented by a PHOC vector. Therefore, the network is used to map all test images on their respective attribute representation. The test set is then ranked by their distance to the query vector. Analogue to [SF17], the cosine distance is used as a distance metric. The cosine distance d between two vectors v and u, with $u \cdot v$ denoting the dot product, is then given by

$$d = 1 - \frac{u \cdot v}{||u||_2 ||v||_2} \tag{5.3.1}$$

where $||v||_2$ denotes the L_2 norm. If no specific queries are defined, each instance of the test set will serve as a query once.

This results in a sorted list of images for each query. For a perfectly accurate system this list starts with all word images relevant to a given query, while the rest of the list

only contains irrelevant items. The Average Precision (AP) constitutes a performance metric, which gives a measure of the accuracy of the retrieval list. Mathematically, Average Precision can be defined as:

$$AP = \frac{\sum_{i=1}^{n} p(i) \cdot r(i)}{t} \tag{5.3.2}$$

Here, n is the length of the retrieval list and $p(i)$ denotes the precision, with respect to the first i elements. Precision corresponds to the fraction of elements relevant to the query. $r(i)$ denotes an indicator function. It returns 0 if the i-th element is irrelevant with respect to the query and 1 otherwise. The denominator t denotes the total number of relevant items in the retrieval list.

Consider the following example. Figure 5.3.1 shows three retrieval lists with 6 elements. Those elements relevant with respect to a given query are depicted in black. The first retrieval list is perfectly accurate, since all relevant elements fill the top ranks. The corresponding Average Precision is 1 and can be computed by:

$$AP_a = \frac{1}{3} \cdot \left[\left(\frac{1}{1} \cdot 1 \right) + \left(\frac{2}{2} \cdot 1 \right) + \left(\frac{3}{3} \cdot 1 \right) + \left(\frac{3}{4} \cdot 0 \right) + \left(\frac{3}{5} \cdot 0 \right) + \left(\frac{3}{6} \cdot 0 \right) \right] = 1 \tag{5.3.3}$$

The second retrieval list still has two relevant items at its top ranks and the third one is ranked on position five. While this list is still fairly similar to the perfectly accurate one, the third list shows a high dissimilarity. The first relevant item is at position two, while the remaining relevant items are at the end of the list. Average Precision gives a numeric value to summarize the obvious observation that list (b) is more accurate than list (c). The respective precisions are given by:

$$AP_b = \frac{1}{3} \cdot \left[\left(\frac{1}{1} \cdot 1 \right) + \left(\frac{2}{2} \cdot 1 \right) + \left(\frac{2}{3} \cdot 0 \right) + \left(\frac{2}{4} \cdot 0 \right) + \left(\frac{3}{5} \cdot 1 \right) + \left(\frac{3}{6} \cdot 0 \right) \right] = 0.87 \tag{5.3.4}$$

$$AP_c = \frac{1}{3} \cdot \left[\left(\frac{0}{1} \cdot 0 \right) + \left(\frac{1}{2} \cdot 1 \right) + \left(\frac{1}{3} \cdot 0 \right) + \left(\frac{1}{4} \cdot 0 \right) + \left(\frac{2}{5} \cdot 1 \right) + \left(\frac{3}{6} \cdot 1 \right) \right] = 0.47 \tag{5.3.5}$$

The overall performance of the word spotting system is then given by the mean Average Precision (mAP). This corresponds to the mean over all Average Precisions computed for each query.

5.4 EXPERIMENTS AND RESULTS

The following sections discuss the results of the different experiments. In order to simplify the naming of the different architectures, the abbreviations given in table 5.4.1

are used. PHOC-DenseNet of configuration A refers to the network architecture presented in section 4.1.1 based on the DenseNet-121. Its hyperparameters always use the values of $k = 32, c_4 = [6, 12, 24, 16], \theta = 0.5$. Configuration B denotes the networks following the method discussed in section 4.1. Here, the two variants B2 and B3 are distinguished depending on the number of densely connected blocks. In both cases, the architecture is further specified by its growth rate k, the number of layers in each block denoted by the vector c_2, respectively c_3, and the compression factor θ. Section 4.1.3 introduces the hybrid approach strongly related to the original PHOCNet. Configuration C refers to the Hybrid-PHOCNet and is entirely defined by its growth rate k and the number of layers in the densely connected part of the network L . The compressed variant of the Hybrid-PHOCNet is referred to as configuration D. All networks of configuration D use a compression factor of $\theta = 0.5$. Furthermore, the overall number of densely connected layer is denoted as L.

Configuration	Method	Hyperparameters
A	PHOC-DenseNet-121	$k = 32, c_4 = [6, 12, 24, 16], \theta = 0.5$
B2	PHOC-DenseNet-b2	$k, c_2 = [b_1, b_2], \theta$
B3	PHOC-DenseNet-b3	$k, c_3 = [b_1, b_2, b3], \theta$
C	Hybrid-PHOCNet	$k, L = b_1$
D	Hybrid-PHOCNet-C	$k, L = b_1 + b_2, \theta = 0.5$

Table 5.4.1: Abbreviations of the different network configurations and their corresponding method. The actual network architectures depend on the given hyperparameters which may vary for different experiments.

5.4.1 *DenseNet-121*

The first set of experiments investigate the use of a densely connected network architecture, which has been designed for the purpose of image classification. As presented in section 4.1.1, only some minor modifications are introduced to the original DenseNet-121 architecture proposed by [HLMW17]. These modifications are only introduced to the classification layers. The original fully-connected layer with softmax activation is replaced by a MLP and sigmoid activation to serve the purpose of learning the binary attribute representation. Training is performed for the IAM and Botany database and follows the approach discussed in section 5.2.

Table 5.4.2 shows the results for QbE and QbS. For all datasets, the network is not able to learn a mapping from word image to PHOC representation. The resulting mAP scores show that no distinctive attribute representation is learned. Based on the extremely low accuracies one can conclude that the network is not able to perform the task of word spotting. Only the mAP in case of QbE for the IAM dataset surpasses 10%, which indicates that the trained network at least maps word images with the same annotation on similar attribute representations. Nonetheless, the resulting representation probably shows no similarity with the actual PHOC vector, which explains the low accuracy for QbS.

Comparing the proposed architecture with the well-performing PHOCNet gives some possible explanations why the proposed network fails. The probably most apparent difference between both network architectures lies in their pooling strategy. While the PHOCNet uses a total of three pooling layers, the DenseNet-121 employs five. An additional convolutional layer, which also serves the purpose of down sampling, is used at the beginning of the network. The pooling strategy of the DenseNet-121 is specifically designed for the ImageNet database. In this case the size of its input images is $[224 \times 224]$. For its four densely connected blocks this gives a feature map sizes of $[56 \times 56]$, $[28 \times 28]$, $[14 \times 14]$ and $[7 \times 7]$. The datasets used for word spotting highly differ from the ImageNet dataset. Word images of variable sizes are used and especially their heights are significantly smaller that 224 pixels. The smallest images in both datasets only consists of $[32 \times 32]$ pixels. The extensive use of pooling layers in the DenseNet-121 results in comparable small feature maps. This might prevent the network form learning distinctive features, which are necessary to learn the PHOC embedding based on character locations and occurrences.

Furthermore, the temporal pyramid pooling layer used in the PHOCNet is not directly applicable to the DenseNet121. As discussed in [SF17], this layer splits the generated set of feature maps following a pyramidal approach. Separating and pooling over individual splits of the feature-maps benefited the performance of the PHOCNet. Again, the small sizes of the resulting feature maps do not allow the application of a TPP in a similar manner.

Method	IAM		Botany	
	QbE	QbS	QbE	QbS
DenseNet-121	29.17	2.21	6.59	7.39

Table 5.4.2: Results for the QbE and QbS experiments in mAP [%].

Overall, the direct application of the DenseNet-121 with only small modifications does not yield any satisfying results. This might be explained by its pooling strategy, which is not suitable for the data used for the word spotting experiments. Since the PHOCNet provides a well performing architecture, the following experiments investigate dense network architectures with a more similar pooling strategy.

5.4.2 *PHOC-DenseNet*

As indicated by the experiments presented in section 5.4.1, the network architecture of the DenseNet-121 is not suitable for the given datasets and their requirements. The following sections present a series of experiments focused on the network architectures PHOC-DenseNet-b2 and PHOC-DenseNet-b3, respectively configurations B2 and B3, as discussed in section 4.1. Their architectures follow a pooling strategy closely related to the TPP-PHOCNet and took inspiration from the DenseNets used for datasets with small input images presented in [HLMW17].

Section 5.4.2 investigates the performance of DenseNet configurations B2 and B3. The presented experiments are concerned with different depth, growth rates and the use of compression layers. Furthermore, the use of maxout layers and different types of pooling layers are evaluated.

Depth and Growth Rate

The experiments on PHOC-DenseNet-b2 and PHOC-DenseNet-b3 investigate different configurations of the basic architectural structure discussed in section 4.1.1. The three different growth rates of $k = 12, 24, 32$ are used. Even though the resulting convolutional layers are quite narrow, the experiments in [HLMW17] show that they still perform well in several networks. For each of the different growth rates, multiple network configurations are evaluated, resulting in increasingly deep networks. The configurations of the PHOC-DenseNet-b2, which employs two densely connected blocks, are $c_2 = [j, 2j]$ with $j = 5, 10, 15, 20$. Using twice the number of layers in the second block of the network is motivated by the DenseNet-121. Furthermore, the size and therefore also the memory consumption of the feature maps in the first block are comparable high. Since the naive implementation of densely connected network is highly memory inefficient, memory is a critical resource for increasingly deep models. A small number of layers at the early stages of the network in combination with an increased number for smaller feature maps sizes, helps in meeting the memory requirements of the networks. The experiments on the PHOC-DenseNet-b3 consider a similar set of configurations with $c_3 = [j, j, 2j]$ for $j = 5, 10, 15$. In comparison to

the PHOC-DenseNet-b2, this corresponds to adding an additional densely connected block before the first pooling layer. Note that some combinations of high growth rates and deep architectures are not evaluated, since they exceed the accessible memory.

All networks are trained following the training setup described in section 5.2. The performances for QbE and QbS on the IAM and Botany database are evaluated according to section 5.3. See table 5.4.3 for an overview on the results of the conducted experiments.

The results show that the given networks are able to learn a set of weights which allow the approximation of the mapping between word image and PHOC vector. Consider the network configuration $c_2 = [5, 10]$ with a growth rate of $k = 12$. Already this comparable shallow network with a considerable small growth rate achieves mAPs above 50% on both datasets. Compared to the approach based on the DenseNet121, the given architectures seem clearly superior. This reinforces the observation that the pooling strategy used in the DenseNet-121 is highly unfavorable for the given data.

Considering the experiments concerned with the PHOC-DensNet-b2, two general trends can be observed and are best demonstrated by the experiments regarding the IAM database. First, the networks' performances increase with an increasingly deep network structure. If the growth rate remains unchanged, a deeper model will perform at least equally well as its shallower counterpart. The performance gain can be explained by the increased model capacity. By adding more layers to the network, the number of parameters increases. The investigated networks seem able to use this additional representational power to learn more complex filters and a more accurate mapping between word image and PHOC vector.

A similar observation can be made with respect to the growth rate. For the IAM database, increasing the network's growth rates results in an improved or at least equal accuracy. A bigger growth rate corresponds to an increased number of filters for each convolutional layer, which also leads to an increased number of parameters. Again, the different networks benefit from the additional parameters in terms of accuracy.

That the proposed densely connected networks generally benefit from an increased model capacity, can also be easily seen in figure 5.4.1. Here, the mAP for the IAM database is plotted with respect to the number of parameters of the different networks. While the performance generally improves with an increased number of parameters, the different growth rates all seem to saturate at a comparable level. Therefore, none of the tested growth rate can be considered superior to the others regarding the overall performance. Nevertheless, networks applying a small growth rate of $k = 12$ seem to exploit their parameters most efficiently. For both scenarios, QbE and QbS, the networks with $k = 12$ require less parameters to achieve similar accuracies as networks with bigger growth rates. The experiments indicate that a small growth rate most

strongly profits from the concept of feature reuse inherent to the dense connectivity pattern.

While the previously discussed observations hold true for the experiments on the Botany database in case of a growth of $k = 12$ and $k = 24$, the networks with $k = 32$ slightly deviate. When the growth rate is increased from $k = 24$ to $k = 32$ for the shallowest network, the accuracy increases in case of QbS, but slightly decreases for

IAM	$c_2 = [5, 10]$	$c_2 = [10, 20]$	$c_2 = [15, 30]$	$c_2 = [20, 40]$
$k = 12$	60.08	64.14	69.39	71.16
$k = 24$	64.04	69.42	70.94	$-^*$
$k = 32$	66.37	70.51	71.33	$-^*$

(a) Results for the QbE experiments on IAM.

IAM	$c_2 = [5, 10]$	$c_2 = [10, 20]$	$c_2 = [15, 30]$	$c_2 = [20, 40]$
$k = 12$	75.19	80.89	82.33	83.61
$k = 24$	78.18	82.86	83.31	$-^*$
$k = 32$	79.82	83.09	83.34	$-^*$

(b) Results for the QbS experiments on IAM.

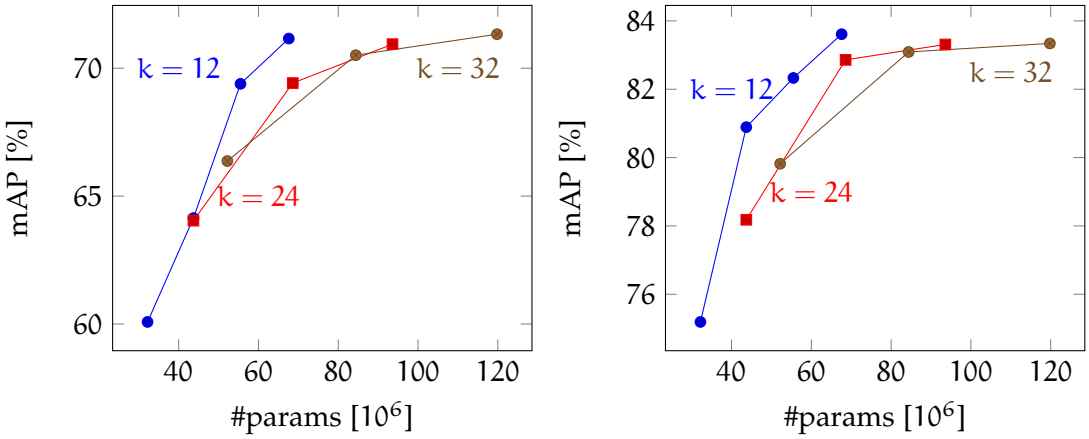
Botany	$c_2 = [5, 10]$	$c_2 = [10, 20]$	$c_2 = [15, 30]$	$c_2 = [20, 40]$
$k = 12$	54.57	63.08	65.93	67.49
$k = 24$	61.77	63.83	67.92	$-^*$
$k = 32$	61.16	60.14	62.88	$-^*$

(c) Results for the QbE experiments on Botany.

Botany	$c_2 = [5, 10]$	$c_2 = [10, 20]$	$c_2 = [15, 30]$	$c_2 = [20, 40]$
$k = 12$	73.69	78.51	79.71	82.43
$k = 24$	75.98	79.21	80.18	$-^*$
$k = 32$	77.72	76.83	78.41	$-^*$

(d) Results for the QbS experiments on Botany.

Table 5.4.3: Results for different hyperparameter settings of DenseNet configuration B2. All results are given in mAP [%]. The experiments marked with $-^*$ were not conducted since they exceed the available memory resources.



(a) QbE (b) QbS

Figure 5.4.1: QbE and QbS accuracies for the IAM database with respect to the number of parameters of each network.

QbE. In case of the two deeper architectures, the networks perform worse than their counterpart with a smaller growth rate.

This observation might possibly be explained by considering the proposed network architecture. Since all feature maps generated in a densely connected network are passed on to the succeeding layers, increasing depth or growth rate results in a higher number of feature maps at the end of the network. The networks used for the experiments employ a TPP-layer in combination with a MLP. If the TPP-layer considers an increased number of feature maps, the size of the resulting feature vector and thereby also the number of parameters of the MLP grow significantly. This increasingly big MLP considers all feature maps generated by the network. As indicated by the consideration on parameter efficiency, a smaller growth rate generates a set of feature maps which are more distinctive compared to those generated with higher growth rates. The combination of this increased number of feature maps of lesser individual significance with an oversized MLP, constitutes a model which is more complex to optimize and might suffer under overparameterization. Regarding the results on the Botany database, which is prone to overfitting due to its comparable small number of training images, this condition might explain why the networks are not able to benefit from the additional representational power. Furthermore, it offers a possible explanation for the saturating behavior of the increasingly deep models.

The results of the experiments using configuration B3 are summarized in table 5.4.4. The same general behavior regarding growth rates and depth of the network can be observed. In both cases an increase in number of parameters benefits the network's performance, while all growth rates achieve similar accuracies. Interestingly, adding an additional densely connected block before the first pooling layer does not significantly increase the network performance. In some cases, consider for example the network

IAM	$c_3 = [5, 5, 10]$	$c_3 = [10, 10, 20]$	$c_3 = [15, 15, 30]$
$k = 12$	57.31	63.63	69.64
$k = 24$	64.44	71.69	$-^*$
$k = 32$	65.89	$-^*$	$-^*$

(a) Results for the QbE experiments on IAM.

IAM	$c_3 = [5, 5, 10]$	$c_3 = [10, 10, 20]$	$c_3 = [15, 15, 30]$
$k = 12$	75.72	81.37	82.86
$k = 24$	79.88	83.47	$-^*$
$k = 32$	80.35	$-^*$	$-^*$

(b) Results for the QbS experiments on IAM.

Botany	$c_3 = [5, 5, 10]$	$c_3 = [10, 10, 20]$	$c_3 = [15, 15, 30]$
$k = 12$	52.95	60.42	60.37
$k = 24$	59.09	64.13	$-^*$
$k = 32$	58.33	$-^*$	$-^*$

(c) Results for the QbE experiments on Botany.

Botany	$c_3 = [5, 5, 10]$	$c_3 = [10, 10, 20]$	$c_3 = [15, 15, 30]$
$k = 12$	72.07	76.50	77.37
$k = 24$	75.68	75.68	$-^*$
$k = 32$	75.19	$-^*$	$-^*$

(d) Results for the QbS experiments on Botany.

Table 5.4.4: Results for different hyperparameter settings of DenseNet configuration B3. All results are given in mAP [%]. The experiments marked with $-^*$ were not conducted since they exceed the available memory resources.

configuration $c_3 = [5, 5, 10]$ with $k = 24$, the introduction of additional layers even decreases the accuracies compared to the corresponding PHOC-DenceNet-b2 ($c_2 = [5, 10], k = 24$). This leads to the conclusion that the PHOC-DenseNet-b2 architecture is more suitable for the considered scenarios.

Compression

All feature maps in a densely connected networks are passed on to their succeeding layers. As discussed in [HLMW17], transition and especially classification layers are highly focused on those feature maps generated at the end of each dense block. Due to this design pattern, which is based on feature reuse, the network contains a comparable high amount of redundancies. [HLMW17] proposes the idea of compression layers to reduce these redundancies and improve model compactness. The following experiments investigate the use of compression layers in the proposed densely connected networks for word spotting. As presented in section 4.1, the DenseNets with configuration B2 and B3 use 1×1 convolution layers as compression layers. The experiments on growth rates and network depth apply a compression factor of $\theta = 1.0$ which corresponds to an uncompressed network. In contrast to its uncompressed version, the following experiments investigate a compression factor of $\theta = 0.5$ for different variants of the configurations B2 and B3. For a compression factor of $\theta = 0.5$, half of the feature maps are dropped at each transition layer. Furthermore, reducing the number of feature maps reduces the memory consumption of the network. This additional capacity in terms of memory is used to train the network configuration B2 with $c_2 = [30, 60]$ and B3 with $c_3 = [15, 20, 40]$. All networks apply a growth rate of $k = 12$, which has shown to be the most parameter efficient.

See tables 5.4.5a and 5.4.5b for the QbE and QbS results for the IAM and Botany database using network configuration B2. The results show that almost all compressed networks achieve accuracies on a comparable level to its uncompressed counterpart. Especially in case of the Botany database, introducing compression to the network architecture even results in an accuracy gain. This observation reinforces the argumentation of [HLMW17]. The significance of feature maps differs and classification, or in this case learning an attribute representation, is not necessarily based on all previously generated feature maps. Although early feature maps are used to learn more complex ones, dropping them at transition layers seems not to significantly affect the network's performance. The introduction of compression yields a more compact and smaller set of feature maps. Regarding the proposed architectures, this also means a reduced number of parameters in the MLP. This increased compactness might explain the slight performance gains, which can be observed in some experiments.

Dataset	θ	$c_2 = [5, 10]$	$c_2 = [10, 20]$	$c_2 = [15, 30]$	$c_2 = [20, 40]$	$c_2 = [30, 60]$
IAM	1.0	60.08	64.14	69.39	71.16	$-^*$
IAM	0.5	58.79	64.84	66.54	71.25	72.06
Botany	1.0	54.57	63.08	65.93	67.49	$-^*$
Botany	0.5	58.26	64.29	66.01	65.96	70.26

(a) Results for QbE experiments with configuration B2.

Dataset	θ	$c_2 = [5, 10]$	$c_2 = [10, 20]$	$c_2 = [15, 30]$	$c_2 = [20, 40]$	$c_2 = [30, 60]$
IAM	1.0	75.19	80.89	82.33	83.61	$-^*$
IAM	0.5	75.71	80.79	81.79	83.57	84.11
Botany	1.0	73.69	78.51	79.71	82.43	$-^*$
Botany	0.5	74.55	80.51	80.75	82.08	82.95

(b) Results for QbS experiments with configuration B2.

Dataset	θ	$c_3 = [5, 5, 10]$	$c_3 = [10, 10, 20]$	$c_3 = [15, 15, 30]$	$c_3 = [15, 20, 40]$
IAM	1.0	57.31	63.63	69.64	$-^*$
IAM	0.5	59.05	67.33	70.32	69.50
Botany	1.0	52.95	60.42	60.37	$-^*$
Botany	0.5	55.92	61.96	62.22	63.64

(c) Results for QbE experiments with configuration B3.

Dataset	θ	$c_3 = [5, 5, 10]$	$c_3 = [10, 10, 20]$	$c_3 = [15, 15, 30]$	$c_3 = [15, 20, 40]$
IAM	1.0	75.72	81.37	82.86	$-^*$
IAM	0.5	76.22	80.98	83.20	82.90
Botany	1.0	72.07	76.50	77.37	$-^*$
Botany	0.5	72.09	77.97	75.43	79.34

(d) Results for QbS experiments with configuration B3.

Table 5.4.5: Results for different compressed DenseNets of configuration B2 and B3. All results are given in mAP [%]. The experiments marked with $-^*$ were not conducted since they exceed the available memory resources.

Another observation can be made with respect to the increasingly deep models. For both databases the additional deep compressed networks perform better compared to the uncompressed shallower variants. The investigated network architecture seems to still improve their performance from increasingly deep models. The application of compression layers reduces memory requirements and the number of parameters of the model, without significantly harming its representational power.

The results on the network configuration B3 which are shown in tables 5.4.5c and 5.4.5d confirm this observation. As discussed previously, the additional block employed by the network configuration B3 does not significantly improve the network's performance. This holds also true for its compressed variants. Additionally, the results show a performance gain for almost all compressed networks compared to the uncompressed variants. Compared to configuration B2, the additional densely connected block in configuration B3 seems to introduce redundancies without providing significant representational power. Therefore, introducing compression layers enforces a more compact model and improves the achieved accuracies. The experiments show that also in its compressed version configuration B2 seems more favorable for the given task and data. Consider for example the network of configuration B2 with $c_2 = [20, 40]$. Increasing the number of layers in each of the two densely connected blocks to $c_2 = [30, 60]$ results in a performance gain for both datasets and query strategies. Instead of increasing the numbers of layers inside the blocks, configuration B3 with $c_3 = [15, 20, 40]$ adds another block with 15 layers. In contrast to configuration B2, the network does not benefit from the additional layers and even shows lower accuracies.

In summary, the introduction of compression layers to the proposed network architecture has shown to result in more compact models. Compression constitutes a suitable method to increase model compactness and reduces the number of parameters and memory consumption.

Maxout

As the experiments on compression layers show, the set of feature maps generated by the proposed architectures contain a lot of redundancies. Compression layers offer a method to reduce the amount of redundancies at the transition between two blocks. Nonetheless, the resulting set of feature maps, which is connected to the classification layers by the last pooling layer, most likely still include redundant information. Additionally, it is probable that the use of a TPP-layer introduces further redundancies by its pyramidal pooling approach. A possible strategy to remove these redundancies and thereby also reduce the number of parameters in the MLP, is the

introduction of a maxout layer between the TPP-layer and the fully-connected part of the network, as presented in section 4.1. Removing redundancies and reducing the size of the feature vector generated by the TPP-layer, also affects the number of parameters in the fully-connected layers. This is especially reasonable, since the combination of an oversized MLP with a redundant set of feature maps, might hinder the network from learning a well-performing set of weights.

The experiments in this section investigate how the introduction of a maxout layer affects the performance of different network architectures. The number of neurons combined by each maxout unit is determined by the hyperparameter K. For each network the values of $K = 2, 4, 8$ are evaluated. $K = 1$ corresponds to the network without maxout layer. Two variants of the dense network configuration B2 are considered. First, an uncompressed network ($\theta = 1.0$) with a high growth rate of $k = 32$ and the configuration $c_2 = [15, 30]$ is evaluated. The high growth rate results in reasonably high numbers of feature maps at the end of the network. Since no compression layers and a high growth rate are employed, the network probably contains many redundancies, which can be reduced by the introduction of a maxout layer. The second architecture employs a small growth rate of $k = 12$, the configuration $c_2 = [30, 60]$ and a compression factor of $\theta = 0.5$. Compared to the first dense network architecture, the application of a small growth rate and a compression layer already gives a more compact model.

Not only the dense connectivity pattern is a possible source of redundancies, but also the TPP-layer. Taking this as a motivation, a third series of experiments considers an adapted TPP-PHOCNet. A maxout layer is introduced to the TPP-PHOCNet architecture presented in section 3.2.1 in between the TPP-layer and the fully-connected part of the network.

All networks are trained on the IAM and Botany database using the training setup presented in section 5.2. Consider table 5.4.6 for the results for QbE and QbS evaluation following the protocol in section 5.3. Regarding the IAM dataset, most networks show only a marginal drop in accuracy for $K = 2$. In case of the uncompressed network with a growth rate of $k = 32$, a slight increase in accuracy can be observed, which reinforces the argument that the given network contains a significant amount of redundancies and therefore even benefits from a reduced number of parameters. For an increased value of $K = 4$ and $K = 8$ the accuracy loss is more significant.

The same observations holds true for the TPP-PHOCNet with respect to the Botany dataset. While the underlying behavior seems similar, the loss of accuracy on the Botany dataset is higher compared to the IAM dataset for both densely connected networks. In case of QbE with the compressed network, a strong decrease in accuracy can be observed already for a value of $K = 2$.

Network	Dataset	$K = 1$	$K = 2$	$K = 4$	$K = 8$
$k = 32, c_2 = [15, 30], \theta = 1.0$	IAM	71.33	71.51	68.73	65.31
$k = 12, c_2 = [30, 60], \theta = 0.5$	IAM	72.06	71.69	69.58	65.85
TPP-PHOCNet	IAM	78.41	78.46	77.03	75.47
$k = 32, c_2 = [15, 30], \theta = 1.0$	Botany	62.88	61.58	60.47	58.41
$k = 12, c_2 = [30, 60], \theta = 0.5$	Botany	70.26	64.17	63.15	61.50
TPP-PHOCNet	Botany	91.44	91.11	88.79	87.88

(a) Results for QbE experiments in mAP [%].

Network	Dataset	$K = 1$	$K = 2$	$K = 4$	$K = 8$
$k = 32, c_2 = [15, 30], \theta = 1.0$	IAM	83.34	84.01	81.42	79.38
$k = 12, c_2 = [30, 60], \theta = 0.5$	IAM	84.11	83.68	82.02	79.94
TPP-PHOCNet	IAM	91.11	90.48	88.93	87.49
$k = 32, c_2 = [15, 30], \theta = 1.0$	Botany	78.41	76.09	78.09	74.65
$k = 12, c_2 = [30, 60], \theta = 0.5$	Botany	82.95	80.44	78.77	76.62
TPP-PHOCNet	Botany	96.51	96.49	95.36	94.35

(b) Results for QbS experiments in mAP [%].

Table 5.4.6: Results for QbE (a) and QbS (b) experiments on the IAM and Botany datasets. K denotes the number of neurons combined by each maxout unit.

The experiments show that the introduction of a maxout layer efficiently serves the purpose of redundancy reduction. Most networks only show a marginal loss of accuracy for $K = 2$. This is especially interesting with respect to the reduced number of parameters in the network. While the achieved accuracies are only slightly reduced, a value of $K = 2$ corresponds to a parameter reduction of 35% for the compressed and 38% for the uncompressed densely connected network. The observation that the feature vector generated by the TPP-layer contains a high amount of redundancies is also true for the TPP-PHOCNet. Introducing a maxout layer with $K = 2$ reduces the number of parameters in the TPP-PHOCNet by 26% without affecting the achieved accuracy.

Pooling

As discussed in the context of the experiments on maxout layers, the TPP-layer might be a possible source of redundancy in the proposed network architectures. The main motivation for using a pyramidal pooling approach is that it allows differently sized word images as inputs. Since the aspect ratios of word images strongly vary depending on the length of the word, the approach presented for the PHOCNet avoids to resize the images to a fix size. The architecture used for the DenseNet121 offers another possibility to handle input images of varying sizes. All layers except the fully-connected ones operate independently from the size of the input image or feature map. Therefore, in order to consider a dataset with images of different sizes the final set of feature maps needs to be mapped on a fixed number of values, which are connected to the fully-connected layers. In case of the TPP-PHOCNet this mapping is performed by the TPP-layer. Regarding the DenseNet121, the use of a global pooling layer is able to serve the same purpose. Since the number of feature maps which are generated over the course of the network does not depend on the size of the input image, global pooling always results in a vector of the same size. From a theoretical point of view, a global pooling layer is equivalent to a TPP-layer with only a single level.

In this series of experiments the influence of different types of pooling layers on the performance of three network architecture is considered. A TPP-layer with five levels, a global max pooling or a global ave pooling layer, is employed as the last pooling layer of a DenseNet of configuration B2 with $k = 12$, $c_2 = [30, 60]$ and $\theta = 0.5$. The same types of pooling layers are used in the PHOCNet architecture to connect convolutional and fully-connected part of the network. The experiments on the DenseNet configuration A, which is based on the DenseNet121, only investigate the use of global pooling layers. Due to the extensive number of pooling layers in the network, the size of the smallest feature maps does not allow a pyramidal pooling approach with multiple levels.

Again, the training procedure uses the setup presented in section 5.2 and evaluation is performed according to section 5.3. See table 5.4.7 for the results on the IAM and Botany database for QbE and QbS. Considering the performance of the DenseNet configuration B2, a decrease of accuracy can be observed when the TPP-layer is replaced by a global pooling layer. Despite being inferior to the accuracies achieved with a TPP-layer, the network with a max pooling layer is till able to learn the desired mapping to some extend. Compared to the max pooling approach, the use of average pooling results in a network unable to learn a set of weights, which allows to perform the task of word spotting. A similar behavior can be observed with respect to the

PHOCNet architecture. The use of the TPP-layer achieves the highest accuracies. While being clearly outperformed by the max pooling approach, the average pooling layer still gives accuracies around 60% for both datasets and query paradigms. Changing the last pooling layer from average to max pooling for the DenseNet configuration A does not solve its inherent problems. For both pooling layers the networks fails in learning a successful mapping from word image on PHOC embedding.

The results indicate that the activation of the feature maps is restricted to limited areas. While this activation is preserved by a global max pooling layer, average pooling averages over the entire feature-map. Especially in case of a big feature-map which is only activated in a small region, this could result in a loss of information, explaining the lower accuracies compared to a max pooling approach. The TPP-layer also achieves the best results for the DenseNet configuration B2. The increased performance with respect to global max pooling, indicates that the TPP-layer preserves some of the spatial information of the feature map activation. This seems especially suitable for learning PHOC vectors which are based on the spatial occurrences of characters. Even though they might introduce additional redundancies, TPP-layers seem to be the best approach for word spotting with densely connected networks.

Network	Pooling	IAM		Botany	
		QbE	QbS	QbE	QbS
$k = 12, c_2 = [30, 60], \theta = 0.5$	TPP	72.06	84.11	70.26	82.95
$k = 12, c_2 = [30, 60], \theta = 0.5$	MAX	60.22	74.44	52.72	69.76
$k = 12, c_2 = [30, 60], \theta = 0.5$	AVE	14.05	0.22	6.85	1.95
PHOCNet	TPP	78.41	91.11	91.44	96.51
PHOCNet	MAX	72.35	81.53	83.45	90.73
PHOCNet	AVE	58.07	60.31	58.32	61.53
DenseNet121	MAX	3.52	0.10	8.34	9.45
DenseNet121	AVE	29.17	2.21	6.59	7.39

Table 5.4.7: Results for QbE and QbS experiments in mAP [%] for different types of pooling layers.

5.4.3 *Resized Images*

All networks used in the previously conducted experiments operate on word images of varying sizes. This approach is also pursued by the PHOCNet which set state-of-the-art accuracies on several benchmark datasets [SF17]. The Triplet-CNN approach discussed in section 3.2.2 does not avoid resizing the original word images. Instead, all images in the respective databases are converted to the fixed size of $[60 \times 160]$ pixels. Even though this approach is beaten by the PHOCNet for most scenarios, it still achieves competitive results.

The following experiments investigate how resizing the training and test images to a fixed size influence network performances. This is of special interest in case of the DenseNet configuration A where the pooling strategy seems unsuitable especially with respect to considerably small images. Following the Triplet CNN approach proposed in [WB16], all training and test images are converted to the fixed sizes of $[30 \times 80]$, $[60 \times 160]$ and $[120 \times 320]$. Training is then performed on the resized IAM and Botany datasets according to the setup presented in section 5.2. QbS and QbE following the protocol in section 5.3 are also performed on a resized test set and in case of QbE with respect to resized query images. Besides the architectures following DenseNet configuration A and B2 with $k = 12$, $c_2 = [30, 60]$ and $\theta = 0.5$, the TPP-PHOCNet is considered.

Table 5.4.8 summarizes the results for QbE and QbS word spotting. In case of DenseNet configuration B2, resizing the word images results in a significant drop of accuracy. For all scenarios, higher accuracies are achieved when word images of variable sizes are used. The size of $[60 \times 160]$ performs best for most scenarios, which indicates that it is closest to the original average image sizes and preserves most of the information. This observation is further supported by the results regarding the TPP-PHOCNet architecture. The use of an image size of $[60 \times 160]$ outperforms the other two sizes. Interestingly, the application of a fixed image size of $[60 \times 160]$ achieves higher accuracies compared to the experiments regarding variable sizes for all scenarios expect for QbE on the IAM dataset. Thereby, one can conclude that the use of variable image sizes is not strictly superior and highly competitive results can be achieved with images of fixed size. Nevertheless, resizing and using word images of bigger sizes does not significantly improve accuracies for the DenseNet configuration A. The combination of the dense connectivity pattern, a high number of pooling stages and a final average global pooling layer, results in an architecture which seems clearly inferior to the other networks.

Network	Dataset	Variable	30×80	60×160	120×320
$k = 12, c_2 = [30, 60], \theta = 0.5$	IAM	72.06	46.36	42.52	39.21
TPP-PHOCNet	IAM	78.41	75.42	75.09	69.09
DenseNet121	IAM	29.17	9.24	22.46	16.69
$k = 12, c_2 = [30, 60], \theta = 0.5$	Botany	70.26	47.72	62.59	50.47
TPP-PHOCNet	Botany	91.44	88.89	93.30	92.04
DenseNet121	Botany	6.59	1.41	13.09	9.77

(a) Results for QbE experiments.

Network	Dataset	Variable	30×80	60×160	120×320
$k = 12, c_2 = [30, 60], \theta = 0.5$	IAM	84.11	72.30	73.59	70.09
TPP-PHOCNet	IAM	91.11	90.71	91.45	87.57
DenseNet121	IAM	2.21	0.39	29.55	6.59
$k = 12, c_2 = [30, 60], \theta = 0.5$	Botany	82.95	67.27	74.87	65.59
TPP-PHOCNet	Botany	96.51	94.34	96.82	96.17
DenseNet121	Botany	7.39	0.44	13.43	10.03

(b) Results for QbS experiments.

Table 5.4.8: Results for QbE and QbS experiments in mAP [%] for different images sizes. Training and testing is performed with respect to the fixed image sizes.

5.4.4 *Hybrid Approach*

The hybrid approach, presented in section 4.1.3, combines the basic architectural structure of the TPP-PHOCNet and the dense connectivity pattern. Therefore, the proposed architecture employs the same number and types of pooling layers as the TPP-PHOCNet. Additionally, the first four convolutional layers as well as the classification layers are chosen analogue to the TPP-PHOCNet. Based on the set of low-level feature maps generated before the second pooling layer, a single densely connected block is used. As shown by the success of the TPP-PHOCNet, first four convolutional layers are able to learn a suitable set of low-level features. The densely connected block serves the purpose of learning features of increasing complexity.

The following series of experiments investigates the proposed architecture and different choices of hyperparameters. Motivated by the experiments in section 5.4.2, all networks use a growth rate of $k = 12$. The uncompressed variant following DenseNet

configuration C employs a compression factor of $\theta = 1.0$. All compressed networks, respectively DenseNet configuration D, have a corresponding compression factor of $\theta = 0.5$.

The experiments consider different numbers of layers in the densely connected part of the network. In terms of the number of feature maps considered by the TPP-layer, the uncompressed network with $k = 12, L = 32, \theta = 1.0$ is most similar to the TPP-PHOCNet. Both architectures generate 512 feature maps, which are then given to the TPP and classification layers. The other networks with an increased number of layers L also generate a higher number of feature maps.

All networks are trained following the training setup presented in section 5.2. Evaluation is performed according to section 5.3 for the IAM and Botany database. Table 5.4.9 shows the results for the QbE and QbS experiments for four different sets of hyperparameters. The networks show a similar behavior as observed for the DenseNets following configuration B, see section 5.4.2. Consider the uncompressed networks with $\theta = 1.0$. While the mAP for the IAM database increases with the increase of layers from 32 to 64, the accuracies drop for Botany. This shows that also the hybrid approach probably suffers from the same architectural drawback like the networks based entirely on the dense connectivity pattern. Increasing the number of layers also increases the number of feature maps, which might lead to an oversized MLP. This observation is further supported by the experiments on configuration D.

Introducing a compression layer with $\theta = 0.5$ yields a more compact set of feature maps, which results in higher accuracies for both datasets. Even though the resulting model incorporates less parameters, a performance gain can be observed. In its compressed variant the network also benefits from an even deeper network structure. The deepest network trainable under the given memory restrictions incorporates 96 layers in the densely connected part of the network. For both datasets, the deepest networks with a compression factor of $\theta = 0.5$ gives the highest accuracies.

Network			IAM		Botany	
k	L	θ	QbE	QbS	QbE	QbS
12	32	1.0	66.95	80.81	73.01	84.71
12	64	1.0	70.30	81.89	71.80	83.96
12	64	0.5	71.05	83.59	73.18	86.33
12	96	0.5	72.54	84.37	75.78	86.78

Table 5.4.9: Results for QbE and QbS experiments in mAP [%] for the IAM and Botany database.

As discussed previously in the context of the experiments on DenseNet configuration B, densely connected networks contain significant redundancies. The use of a compression layer in DenseNet configuration D removes some redundancies explaining the improved accuracies. As an additional measure, the following experiments investigate the use of maxout units for DenseNet configuration D. Analogue to configuration B, a maxout layer is introduced between TPP-layer and the fully-connected part of the network. The following experiments consider the previously used networks of configuration D with $k = 12, \theta = 0.5, L = 64$ and $L = 96$. Both networks are evaluated for values of $K = 2, 4, 8$, while $K = 1$ corresponds to the network without maxout layer. See table 5.4.10 for the results for QbE and QbS experiments on the IAM and Botany dataset.

The results show that combining two neurons by use of a maxout unit results in a slight decrease in accuracy, which can be explained by the reduced number of parameters. Increasing the number of neurons combined by each maxout unit gives an increasingly severe performance loss. While the compressed DenseNets of configuration D without maxout units benefit from an increased number of layers, this observation does not hold true for the use of a maxout layer with $K = 2$ for the Botany datasets. For QbE and QbS, the shallower network with maxout layer achieves

Network	Dataset	$K = 1$	$K = 2$	$K = 4$	$K = 8$
$k = 12, L = 64, \theta = 0.5$	IAM	71.05	68.75	65.28	59.78
$k = 12, L = 96, \theta = 0.5$	IAM	72.54	71.05	69.33	64.17
$k = 12, L = 64, \theta = 0.5$	Botany	73.18	72.12	66.77	59.67
$k = 12, L = 96, \theta = 0.5$	Botany	75.78	71.45	71.01	66.06

(a) Results for QbE experiments.

Network	Dataset	$K = 1$	$K = 2$	$K = 4$	$K = 8$
$k = 12, L = 64, \theta = 0.5$	IAM	83.59	83.02	80.55	75.44
$k = 12, L = 96, \theta = 0.5$	IAM	84.37	83.18	81.69	79.07
$k = 12, L = 64, \theta = 0.5$	Botany	86.33	85.73	79.13	75.48
$k = 12, L = 96, \theta = 0.5$	Botany	86.78	85.31	82.65	80.89

(b) Results for QbS experiments.

Table 5.4.10: Results for QbE and QbS experiments in mAP [%] for the IAM and Botany database.

a higher mAP than its deeper counterpart. On the IAM dataset, the deeper network with maxout layer achieves better results than the shallower one for QbE and QbS.

Overall, the hybrid approach shows a similar behavior as the entirely densely connected networks. In general, the models benefit from a deeper structure, but their absolute performance is limited by redundancies and overparameterization. Including a maxout layer which combines 2 neurons can be used to significantly reduce the number of parameters in the MLP with only a little influence on the achieved accuracies.

5.4.5 *Comparison*

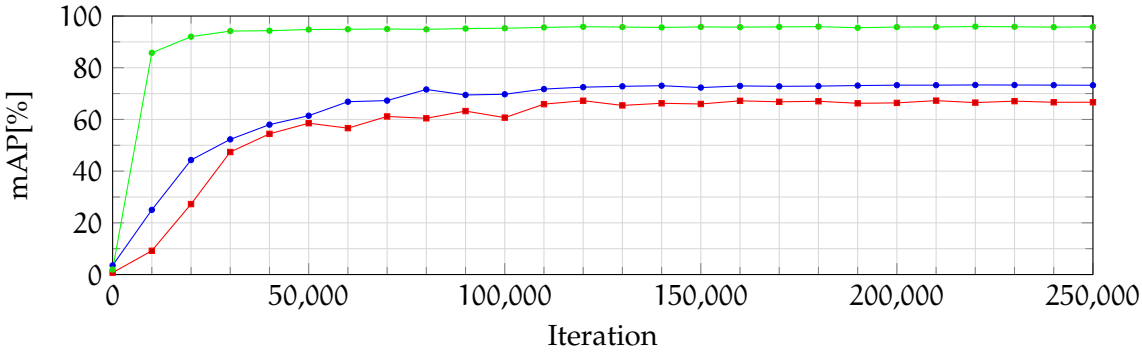
In order to compare the proposed methods with the state-of-the-art, the following experiments are considered. The TPP-PHOCNet and two DenseNets with configuration B2 and D are trained on the IAM, Botany and George Washington datasets. Both densely connected networks use a learning rate of $k = 12$ and employ a compression factor of $\theta = 0.5$. Since the combination of a deep network with a small learning rate gave the best results for previous experiments, the DenseNet of configuration B2 employs 90 densely connected layers with $c_2 = [30, 60]$. For DenseNet configuration D, the deepest trainable model with respect to memory limitations is used as well and it incorporates a number of densely connected layers of $L = 96$.

As shown by [SF17], the TPP-PHOCNet improves its performance after surpassing 100000 training iterations, especially with respect to the IAM database. Therefore, the previously used training setup, presented in section 5.2, is slightly modified. All networks are trained for 250000 iterations. The initial learning rate of 10^{-4} is divided by 10 after 100000 and 200000 iterations.

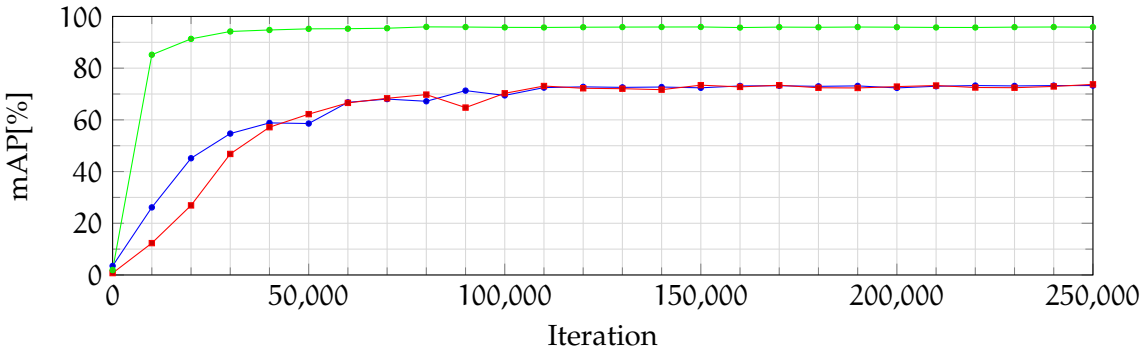
Figure 5.4.2 shows the evolution of the achieved mAP for QbE on all three datasets with respect to the different network architectures.

Both DenseNet configurations achieve high accuracies for the George Washington datasets already for a small number of training iterations. Compared to the TPP-PHOCNet, both DenseNets achieve higher accuracy after 10000 iterations. Despite the slightly faster convergence of the DenseNets, all considered network architectures converge to a mAP of above 95% in case of the George Washington dataset. Only a small improvement in performance can be observed beyond 50000 iterations.

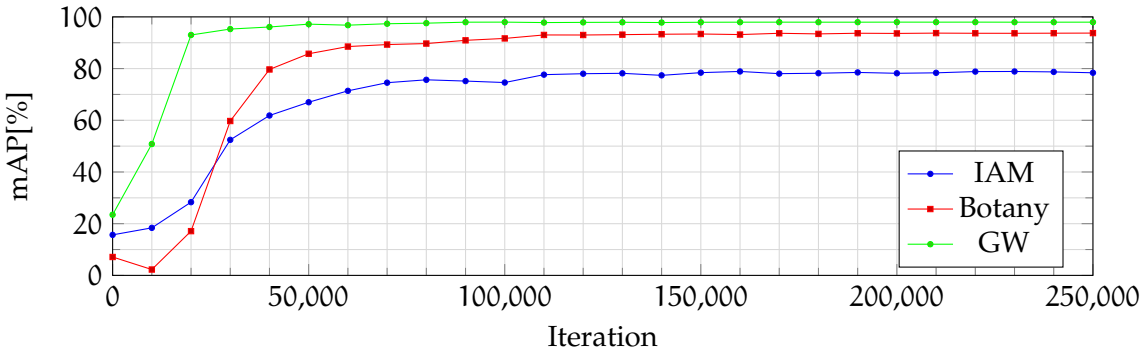
The observation of DenseNets converging slightly faster can also be made for the IAM and Botany datasets. While the TPP-PHOCNet shows only minor accuracy gains



(a) DenseNet configuration B2



(b) DenseNet configuration D



(c) TPP-PHOCNet

Figure 5.4.2: Evolution of mAP [%] for QbE experiments over the course of training. Each network architecture is evaluated for the three dataset of George Washington (GW), Botany and IAM.

Method	GW		IAM		Botany	
	QbE	QbS	QbE	QbS	QbE	QbS
DenseNet B2	95.80	96.66	73.81	85.28	66.49	82.27
DenseNet D	95.87	97.32	74.22	84.79	76.45	86.28
TPP-PHOCNet	97.75	97.39	81.03	92.42	93.75	97.35
Attribute SVM [1]	93.04	92.64	55.73	73.72	75.77[2]	65.69[2]
Deep Feat. Embedding [3]	94.41	92.84	84.24	91.58	–	–
Triplet CNN [4]	98.00	93.69	81.58	89.49	54.95[2]	3.40[2]

Table 5.4.11: Results for QbE and QbS experiments in mAP [%] for the IAM and Botany database.

for less than 20000 iterations, both DenseNets start to converge from the beginning of the training procedure. In terms of final accuracies, the TPP-PHOCNet outperforms both DenseNets on all considered datasets. Independent of the number of training iterations, the DenseNet B2 achieves higher accuracies on the IAM dataset compared to the Botany datasets. Considering DenseNet D, accuracies for the IAM dataset are slightly higher than for Botany for less than 40000 iterations, but converge to a similar level for both datasets. The TPP-PHOCNet performs clearly best on the Botany datasets. Accuracies for Botany surpass those for IAM already after a short training period and converge to a high level above 90%. An increase in number of iterations yields only minor improvements for all networks. Nonetheless, mAPs increase by roughly 2% by training for more than 100000 iterations on IAM and Botany.

See table 5.4.11 for a summary of the results for QbE and QbS experiments. Both DenseNets perform best on the George Washington datasets, which can be explained by its relatively low complexity. While DenseNet B2 beats DenseNet D on the IAM database, DenseNet D achieves slightly higher accuracies on the Botany datasets. On all three datasets the TPP-PHOCNet performs best and achieves the highest accuracies.

To compare the proposed method with the state-of-the-art, table 5.4.11 also shows the accuracies reached by three other popular methods. [AGFV14] proposed to use a set of support vector machines to learn a PHOC representation from a fisher vector based on the word image. This method is referred to as Attribute SVM, since it

1 results obtained from [AGFV14]
2 results obtained from [PZG+16]
3 results obtained from [KDJ16]
4 results obtained from [WB16]

employs a set of SVMs to learn an attribute representation. The method of Deep Feature Embedding, proposed in [KDJ16], follows a similar strategy. Instead of using Fisher Vectors, a feature embedding is derived by using a CNN. The CNN is trained on a given datasets and learns to map a word image on its designated class. The vector, generated by the second fully-connected layer, is then taken as a feature vector. Instead of performing 1 out of k classification, the feature vector is used to train a set of SVMs analogue to [AGFV14]. Triplet CNN refers to the method proposed by [WB16], which is based on using three identical residual networks. For a more detailed discussion on this method see section 3.2.2.

Both DenseNet architectures achieve highly competitive results on the George Washington dataset. Except for the results obtained by the Triplet CNN for QbE, both DenseNets are only outperformed by the TPP-PHOCNet. Considering the IAM dataset, the proposed method performs better than the Attribute SVM method, but it is not able to accomplish accuracies competitive to the other CNN based methods. On Botany, the DenseNets achieve better results for QbS compared to the Attribute SVM method, which performs better for QbE. While no experiments for the Deep Feature Embedding approach have been published with respect to the Botany datasets, the Triplet CNN method only achieves extremely low accuracies that are clearly inferior to the performance of the proposed method. Especially, in the case of Botany, the TPP-PHOCNet clearly outperforms all other methods.

As shown by the results for the different methods, CNN based approaches seem highly suitable for the task of word spotting. Furthermore, using a CNN to learn an attribute representation is a highly competitive approach and achieves state-of-the-art accuracies. Although the proposed method follows this general approach, which is also pursued by the well-performing TPP-PHOCNet, the achieved accuracies are highly determined by the architecture of the CNN. Especially for complex datasets as IAM or Botany, the proposed densely connected networks are not able to compete with the robust TPP-PHOCNet. This might not be a direct consequence of the dense connectivity pattern, but is probably rather the result of combining the dense architecture with the same classification strategy applied in the TPP-PHOCNet.

6 CONCLUSIONS

Convolutional neural networks have been proven to be a powerful tool for word spotting in handwritten documents. State-of-the-art performances on most benchmark datasets were set by the PHOCNet, which learns a mapping between word images and an attribute embedding. This mapping from image to PHOC vector, where image attributes correspond to character occurrences and location, allows for successful QbE and QbS word spotting. Inspired by its success in the field of image classification, this thesis investigated the special design pattern of densely connected networks in the context of word spotting. The proposed method follows the same concept as the PHOCNet in terms of learning an attribute representation.

In a first series of experiments, it was shown that in order to successfully use a densely connected network for word spotting a suitable pooling strategy is required. Only adapting the classification layers of the DenseNet-121, which achieves state-of-the-art accuracies in image classification, does not give any satisfactory results. This insight motivated the proposed dense architectures based on the the TPP-PHOCNet. Replacing the convolutional parts of the TPP-PHOCNet by densely connected blocks and keeping the original pooling strategy, resulted in significantly improved performances. The pooling strategy provided by the TPP-PHOCNet limits the dense architecture to either employ two or three densely connected blocks. Experiments have shown that the use of two blocks generally performs better than a three block based approach. Principally, the densely connected networks benefit from an increasingly deep architecture. Also, the growth rate of the network strongly influences its properties. No growth rate was found strictly superior in terms of achieved accuracies. The conducted experiments have shown that a small growth rate of $k = 12$ is more parameter efficient compared to higher growth rates and, given a suitable number of layers, achieves competitive accuracies. Nonetheless, accuracies saturate for increasing numbers of parameters. This might be explained by an unfavorable architectural drawback of the proposed architecture. An increased number of layers or a higher growth rate is directly linked to an increased number of feature maps at the end of the network. Therefore, increasing the number of parameters and thereby also the representational power of the convolutional part of the network, corresponds at the same time to a higher number of parameters in the MLP, which is used to learn the attribute representations. The resulting model, which combines a possibly oversized MLP with an only slightly

more powerful convolutional part, is probably harder to optimize and it is prone to overfitting which explains the observed saturating behavior.

In this thesis two methods to reduce the number of parameters in the fully-connected part of the networks are evaluated. Experiments on different types of pooling layers have shown that even though a TPP-layer might introduce redundancies to the network it still gives the best results for word spotting with the PHOCNet and the proposed dense architectures. The introduction of a maxout layer to the investigated architectures proved to be a successful method for parameter reduction. As discussed in section 5.4.2, the overall number of parameters in the TPP-PHOCNet may be reduced by 26% without significantly affecting the achieved accuracy. With respect to the investigated DenseNets, the parameter reduction is even more significant, but compared to the PHOCNet the performance is less robust to the removed parameters.

Instead of using a specifically designed pooling strategy, another set of experiments is concerned with changing the input images by resizing. This approach does not improve accuracies for the adapted DenseNet-121. The architecture based on two densely connected blocks shows high performance losses for fixed image sizes. Only the TPP-PHOCNet is robust to resizing images to a fixed size. Using an image size of 60×160, it even achieves higher accuracies for QbE and QbS on Botany and QbS on IAM. This leads to the conclusion that variable image sizes are not necessarily superior to an approach based on resizing. The hybrid approach, which combines TPP-PHOCNet with a single densely connected block, shows a similar behavior as the entirely dense architectures. This shows that the actual performance of the proposed architectures is highly determined by the MLP used as classification layers.

While the PHOCNet provides a robust architecture performing well on multiple benchmark datasets, the proposed densely connected networks are unable to achieve competitive accuracies. The dense connectivity pattern in combination with a classification strategy based on a MLP with Sigmoid activation seems to result in a unfavorable architecture for word spotting. Following the proposed approach, increasing the number of layers in the network results in an increased number of feature maps. Future work might focus on networks which incorporate identity paths without directly linking depth and the generated number of feature maps. Here, a suitable choice could be residual networks, which also provided highly competitive results in image classification. The use of maxout units in combination with densely connected networks is probably also a suitable choice in other applications. While the dense connectivity pattern inherently includes a high amount of redundancies, reducing the number of parameters in a DenseNet can as well be of interest in other fields such as image classification.

BIBLIOGRAPHY

[AFV13] Almazan, Jon ; Fornes, Alicia ; Valveny, Ernest: Deformable HOG-Based Shape Descriptor. In: *International Conference on Document Analysis and Recognition*, 2013

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

[ARTL13] Aldavert, David ; Rusinol, Marcal ; Toledo, Ricardo ; Llados, Josep: Integrating Visual and Textual Cues for Query-by-String Word Spotting. In: *International Conference on Document Analysis and Recognition*, 2013

[Bis11] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. Springer-Verlag GmbH, 2011

[BJTM16] Balntas, Vassileios ; Johns, Edward ; Tang, Lilian ; Mikolajczyk, Krystian: PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors. In: *Computing Research Repository* abs/1601.05030 (2016)

[Dan80] Danielsson, Per-Erik: Euclidean distance mapping. In: *Computer Graphics and Image Processing* 14 (1980), Nr. 3, S. 227–248

[FEHF09] Farhadi, A. ; Endres, I. ; Hoiem, D. ; Forsyth, D.: Describing objects by their attributes. In: *Conference on Computer Vision and Pattern Recognition*, 2009

[FFJK17] Fei-Fei, Li ; Johnson, Justin ; Karpathy, Andrej: *CS231n: Convolutional Neural Networks for Visual Recognition*. http://cs231n.stanford.edu/. Version: 2017

[FFMB12] Frinken, V. ; Fischer, A. ; Manmatha, R. ; Bunke, H.: A Novel Word Spotting Method Based on Recurrent Neural Networks. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (2012), Nr. 2, S. 211–224

[FKFB10] Fischer, Andreas ; Keller, Andreas ; Frinken, Volkmar ; Bunke, Horst: HMM-based Word Spotting in Handwritten Documents Using Subword Models. In: *International Conference on Pattern Recognition*, IEEE, 2010

[FM82] Fukushima, Kunihiko ; Miyake, Sei: Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. In: *Pattern Recognition* 15 (1982), Nr. 6, S. 455–469

[GB10] Glorot, Xavier ; Bengio, Yoshua: Understanding the difficulty of training deep feedforward neural networks. In: *International Conference on Artificial Intelligence and Statistics* Bd. 9, 2010, S. 249–256

[GBB11] Glorot, Xavier ; Bordes, Antoine ; Bengio, Yoshua: Deep Sparse Rectifier Neural Networks. In: *Proceedings of the International Conference on Artificial Intelligence and Statistics* Bd. 15, 2011, S. 315–323

[GBC17] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. The MIT Press, 2017

[GMAJ13] Goel, Vibhor ; Mishra, Anand ; Alahari, Karteek ; Jawahar, C.V.: Whole is Greater than Sum of Parts: Recognizing Scene Text Words. In: *International Conference on Document Analysis and Recognition*, 2013

[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

[GWFM+13] Goodfellow, Ian J. ; Warde-Farley, David ; Mirza, Mehdi ; Courville, Aaron ; Bengio, Yoshua: Maxout networks. In: *arXiv preprint arXiv:1302.4389* (2013)

[HLMW17] Huang, Gao ; Liu, Zhuang ; Maaten, Laurens van d. ; Weinberger, Kilian Q.: Densely Connected Convolutional Networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2017

[HZRS15] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (2015), Nr. 9, S. 1904–1916

[HZRS16a] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2016

[HZRS16b] He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Identity Mappings in Deep Residual Networks. In: *European Conference on Computer Vision*, 2016, S. 630–645

[IS15] Ioffe, Sergey ; Szegedy, Christian: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Computing Research Repository* abs/1502.03167 (2015)

[JSD+14] Jia, Yangqing ; Shelhamer, Evan ; Donahue, Jeff ; Karayev, Sergey ; Long, Jonathan ; Girshick, Ross ; Guadarrama, Sergio ; Darrell, Trevor: Caffe. In: *Proceedings of the ACM International Conference on Multimedia - MM '14*, ACM Press, 2014

[KDJ16] Krishnan, Praveen ; Dutta, Kartik ; Jawahar, C.V.: Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text. In: *International Conference on Frontiers in Handwriting Recognition*, 2016

[Kon05] Konar, Amit: *Computational Intelligence*. Springer-Verlag GmbH, 2005

[KSH12] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems* 60 (2012), Nr. 6, S. 84–90

[LBBH98] Lecun, Y. ; Bottou, L. ; Bengio, Y. ; Haffner, P.: Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE* 86 (1998), Nr. 11, S. 2278–2324

[MB02] Marti, U.-V. ; Bunke, H.: The IAM-database: an English sentence database for offline handwriting recognition. In: *International Journal on Document Analysis and Recognition* 5 (2002), Nr. 1, S. 39–46

[MHRC96] Manmatha, R. ; Han, Chengfeng ; Riseman, E. M. ; Croft, W. B.: Indexing handwriting using word matching. In: *Proceedings of the ACM international conference on Digital libraries -*, 1996

[MP43] McCulloch, Warren S. ; Pitts, Walter: A logical calculus of the ideas immanent in nervous activity. In: *The Bulletin of Mathematical Biophysics* 5 (1943), Nr. 4, S. 115–133

[MP69] Minsky, Marvin ; Papert, Seymour: *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969

[MPCB14] Montúfar, Guido ; Pascanu, Razvan ; Cho, Kyunghyun ; Bengio, Yoshua: On the Number of Linear Regions of Deep Neural Networks. In: *International Conference on Neural Information Processing Systems*, 2014, 2924–2932

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

[PMB12] Pascanu, Razvan ; Mikolov, Tomas ; Bengio, Yoshua: Understanding the exploding gradient problem. In: *Computing Research Repository* abs/1211.5063 (2012)

[PSM10] Perronnin, Florent ; Sánchez, Jorge ; Mensink, Thomas: Improving the Fisher Kernel for Large-Scale Image Classification. In: *European Conference on Computer Vision*, 2010, S. 143–156

[PZG+16] Pratikakis, Ioannis ; Zagoris, Konstantinos ; Gatos, Basilis ; Puigcerver, Joan ; Toselli, Alejandro H. ; Vidal, Enrique: ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016). In: *International Conference on Frontiers in Handwriting Recognition*, 2016

[RATL11] Rusinol, Marcal ; Aldavert, David ; Toledo, Ricardo ; Llados, Josep: Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method. In: *International Conference on Document Analysis and Recognition*, 2011, S. 63–67

[RATL15] Rusiñol, Marçal ; Aldavert, David ; Toledo, Ricardo ; Lladós, Josep: Efficient segmentation-free keyword spotting in historical document collections. In: *Pattern Recognition* 48 (2015), Nr. 2, S. 545–555

[RF15] Rothacker, Leonard ; Fink, Gernot A.: Segmentation-free query-by-string word spotting with Bag-of-Features HMMs. In: *International Conference on Document Analysis and Recognition*, 2015

[RGF17] Rueda, Fernando M. ; Grzeszick, Rene ; Fink, Gernot A.: Neuron Pruning for Compressing Deep Networks Using Maxout Architectures. In: *Proceedings of the German Conference on Pattern Recognition* (2017), S. 177–188

[RHW86] Rumelhart, David E. ; Hinton, Geoffrey E. ; Williams, Ronald J.: Learning representations by back-propagating errors. In: *Nature* 323 (1986), Nr. 6088, S. 533–536

[RM07] Rath, Toni M. ; Manmatha, R.: Word spotting for historical documents. In: *International Journal of Document Analysis and Recognition* 9 (2007), Nr. 2-4, S. 299–299

[Roj96] Rojas, Raul: *Neural Networks*. Springer Berlin Heidelberg, 1996

[Ros58] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. In: *Psychological Review* 65 (1958), Nr. 6, S. 386–408

[RSP09] Rodríguez-Serrano, José A. ; Perronnin, Florent: Handwritten word-spotting using hidden Markov models and universal vocabularies. In: *Pattern Recognition* 42 (2009), Nr. 9, S. 2106–2116

[SF15] Sudholt, Sebastian ; Fink, Gernot A.: A Modified Isomap Approach to Manifold Learning in Word Spotting. In: *German Conference on Pattern Recognition* (2015), S. 529–539

[SF16] Sudholt, Sebastian ; Fink, Gernot A.: PHOCNet : A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents. In: *International Conference on Frontiers in Handwriting Recognition*, 2016

[SF17] Sudholt, Sebastian ; Fink, Gernot A.: Evaluating Word String Embeddings and Loss Functions for CNN-Based Word Spotting. In: *International Conference on Document Analysis and Recognition*, 2017

[SGS15] Srivastava, Rupesh K. ; Greff, Klaus ; Schmidhuber, Jürgen: Training Very Deep Networks. In: *International Conference on Neural Information Processing Systems*, 2015, 2377–2385

[SHK+14] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* 15 (2014), Nr. 1, S. 1929–1958

[SK99] Sankoff, David ; Kruskal, Joseph: *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Center for the Study of Language and Inf, 1999

[SVI+16] Szegedy, Christian ; Vanhoucke, Vincent ; Ioffe, Sergey ; Shlens, Jon ; Wojna, Zbigniew: Rethinking the Inception Architecture for Computer Vision. In: *Conference on Computer Vision and Pattern Recognition*, 2016

[SZ14] Simonyan, Karen ; Zisserman, Andrew: Very deep convolutional networks for large-scale image recognition. In: *Proceedings of the International Conference on Learning Representations* (2014)

[WB16] Wilkinson, Tomas ; Brun, Anders: Semantic and Verbatim Word Spotting Using Deep Neural Networks. In: *International Conference on Frontiers in Handwriting Recognition*, 2016, S. 307–312