Exploring Architectures for CNN-Based Word Spotting

Anonymous

Abstract—**The goal in word spotting is to retrieve parts of document images which are relevant with respect to a certain user-defined query. The recent past has seen attribute-based Convolutional Neural Networks take over this field of research. As is common for other fields of computer vision, the CNNs used for this task are already considerably deep. The question that arises, however, is: How complex does a CNN have to be for word spotting? Are increasingly deeper models giving increasingly better results or does performance behave asymptotically for these architectures? On the other hand, can similar results be obtained with a much smaller CNN? The goal of this paper is to give an answer to these questions. Therefore, the recently successful TPP-PHOCNet will be compared to a Residual Network, a Densely Connected Convolutional Network and a LeNet architecture empirically. As will be seen in the evaluation, a complex model can be beneficial for word spotting on harder tasks such as the IAM Offline Database but gives no advantage for "easier" benchmarks such as the George Washington Database.**

I. Introduction

Word Spotting in handwritten documents refers to the image retrieval task with the goal to obtain a list of word images based on a relevance ranking for a given user's query. Since the introduction of word spotting in handwritten documents by Manmatha et. al. [1], many promising approaches were proposed. Over the years the performance achieved by different approaches steadily increased mainly by introducing more powerful feature representations.

In [2] a *Convolutional Neural Network (CNN)*, trained with the backpropagation algorithm, was successfully applied on a recognition task consisting of handwritten digits. Encouraged by the great success of CNNs in recognition tasks [3] [4], Sudholt et. al. [5] proposed the TPP-PHOCNet, a CNN based on the attribute representation proposed by Almazan et. al [6] namely the *Pyramidal Histogram of Characters (PHOC)*. In 2015 He et. al. [15] introduced the Residual Network (ResNet) and achvied state-of-the-art results in image classification. Despite of an enormous depth the ResNets can be efficiently trained by proposing residual modules containing skip-connections. Only one year later the Densely Connected Convolutional Network (DenseNet) [7] was introduced with an enhanced approach concerning skip-connections. In contrast to ResNets, which are based on the summation of layer output and skip-connection, the DenseNet is built upon the respective concatenation. This results in the connection of every layer to all subsequent layers and hence the name "densely connected".

In this work, we explore the TPP-PHOCNet, ResNet, and DenseNet on three standard word spotting benchmarks. All three architectures were adjusted and specifically designed for word spotting, by using the PHOC attribute representation. We evaluate the CNNs for both the Query-by-Example (QbE) and Query-by-String (QbS) scenario. This work shows that deeper networks as well as newer architectures do not necessarily perform better.

Sec. II presents segmentation-based word spotting methods and gives a brief review of the explored CNN architectures. The CNN architectures we used for this work are presented in Sec. III followed by the evaluation in Sec. IV. Finally a conclusion is drawn over the proposed exploration in Sec. V.

II. Related Work

A. Word Spotting

The goal in word spotting is to retrieve word images from a given document image collection, rank these word images by a certain criteria, and sort them by their relevance with respect to a certain query. These queries can either be word images, defined by a user cropping a snipped from a document page, or defining a word string which needs to be retrieved. Based on these two query definitions, word spotting distilled two scenarios namely QbE and QbS. For QbE, queries are given as word images and retrieval is based on a distance-based comparison. To perform such a comparison, a mapping needs to be find from a visual to a vector representation. In the QbS scenario, queries are given as word strings. In contrast to QbE, the user is not required to search for a visual example of a word in a document collection.

Next to several approaches to address the problem of query representations based on visual similarity [8], [9]. A very influential approach for learning a mapping from a visual to textual representation was presented in [6]. The authors proposed a word string embedding in order to project word images into a common space based on a binary histogram of character occurrences namely *Pyramidal Histogram of Characters PHOC*. A given word string is partitioned based on pyramidal levels, where each level splits the string with respect to the level number. For example, level one do not splits the string, whereas level two splits the string in two halves (ex. "home" into "ho" and "me"), level three splits the string in three parts, and so on.

Due to the great success of the PHOC word embedding, the focus concerning feature representation shifted towards trainable feature extractions. The PHOC embedding was originally proposed using low-level image descriptors like SIFT [10] in combination with Fisher-Vectors [11] and a linear SVM as classifier. Inspired by an outstanding performance of CNNs in many computer vision classification tasks [3], [4], Sudholt et al proposed the TPP-PHOCNet with the idea to replace the classification pipeline in [6] with jointly trainable CNNs.

B. *Deep Learning*

Convolutional Neural Networks (CNNs) have made their mark in various fields of computer vision and become the dominant pattern recognition approach for visual object recognition. Introduced over 20 years ago [12], CNNs were able to achieve comparable results but quickly comes to their limits. Based on the usage of a sigmoidal activation function, CNNs were limited to have deep architectures exploiting more convolutional layers. The key contribution was made by introducing the *Rectified Linear Unit (ReLU)* as activation function [13]. Since that CNNs were designed to have deeper architectures, starting with the AlexNet [3] using 5 convolutional layers. In 2015 the VGGNet [4] was proposed using 19 convolutional layers. One year later the first two networks were capable of surpassing the 100-layer barrier, namely the Highway Networks [14] and the Residual Networks (ResNets) [15]. A *degradation* problem has been exposed when deeper networks are able to start converging. The accuracy gets saturated with the increasing network depth and than rapidly degrades. Adding more layers to the network leads to higher training error. Against the first intuition this degradation is not caused by overfitting as reported in [16] and [14].

1) ResNet: The *Residual Networks (ResNets)* address the problem of degradation, using so-called *Residual Blocks* in the network architecture. The authors let the layers explicitly fit a residual mapping [15], formally denoting the mapping as $\mathcal{H}(x) = \mathcal{F}(x) + x$. Here x is the input of a residual building block and $\mathcal{F}(x)$ the output of the layers within a building block. $\mathcal{H}(x)$ is than the summation of both in order to obtain the mapping, with the identity x as a "shortcut connection". The ResNet comes with two versions of residual building blocks. For "shallower" residual networks (18- or 34-layers) the building blocks contain two 3×3 convolution layers. Whereas the "deeper" networks (50-, 101-, and 151-layers) are built on so-called residual bottleneck blocks, containing 1×1, 3×3, and 1×1 convolution layers consecutive. As the first 1×1 layer reduces the dimensionality, the second 1×1 convolution layer is restoring the dimensionality. Each residual bottleneck block has an ordinal n which determines the number of filter kernels used in a specific block as follows: The first and second layer in the block have $n/4$ filters while the third layer makes use of n filters. For example if $n = 256$ than layer one and two have 64 filter kernels and layer three has 256 filter kernels.

2) DenseNets: Similar to the ResNet architecture, DenseNets [7] also utilize identity connections. A skip-connection bypasses each layer. This identity path is then concatenated with the feature-maps generated by the layer. Following this connectivity scheme, each layer receives all preceding feature-maps as an input. The number of feature-maps added by each layer is referred to as the network's growth rate. Already small growth rates are sufficient to achieve state-of-the-art results, resulting in very narrow layers. To avoid the concatenation of differently sized feature-maps, DenseNets are organized into densely connected blocks. Consequently, pooling layers are only used outside the dense blocks. The model compactness of DenseNets is further improved by the introduction of compression layers. A compression layer corresponds to a convolutional layer with kernel size one. DenseNets tend to make stronger use of high-level features learned at the end of a dense block. Therefore, the convolutional layer reduces the number of features maps by a compression factor. The DenseNet architecture outperforms other networks such as ResNets in various state-of-the-art benchmark experiments in the field of image classification. It has been shown, that the dense architecture is especially parameter efficient and achieves competitive results, with a significantly less numbers of parameters.

III. Architectures for Word Spotting

In this section, we describe the different architectures we evaluated for the word spotting task. The order of the different architectures is chosen such that the networks get increasingly deeper and more complex from an architectural point of view. All presented architectures are trained to predict the PHOC representation [6] using the Binary Cross Entropy Loss (BCEL). While there exist other word string embedding types for word spotting, this combination achieves consistently good results for a number of different word spotting benchmarks and has the fastest training times [17].

A. *PHOCLeNet*

Although the title of first CNN goes to the Neocognitron [18], the *LeNet* was the first CNN to be trained in a supervised fashion with the backpropagation algorithm [2]. It consists of seven layers: The first four layers build the convolutional part of the CNN and are made up of two consecutive combinations of convolutional layer and max pooling layers. While the first of these two layers uses filter 28×28 filter kernels in order to produce six feature maps, the second convolutional layer uses 14×14 kernels and produces 16 feature maps. After the convolutional part, the LeNet employs a Multilayer Perceptron (MLP) with two hidden layers. The number of the neurons in these two layers are 120 and 84. The size of the output layer depends on the number of classes to be predicted from the CNN. All hidden layers of the LeNet which carry weights make use of a hyperbolic tangent as activation function.

For the purpose of predicting attributes with the LeNet, we make three subtle changes to the original architecture: First, the hyperbolic tangent activation functions are replaced with

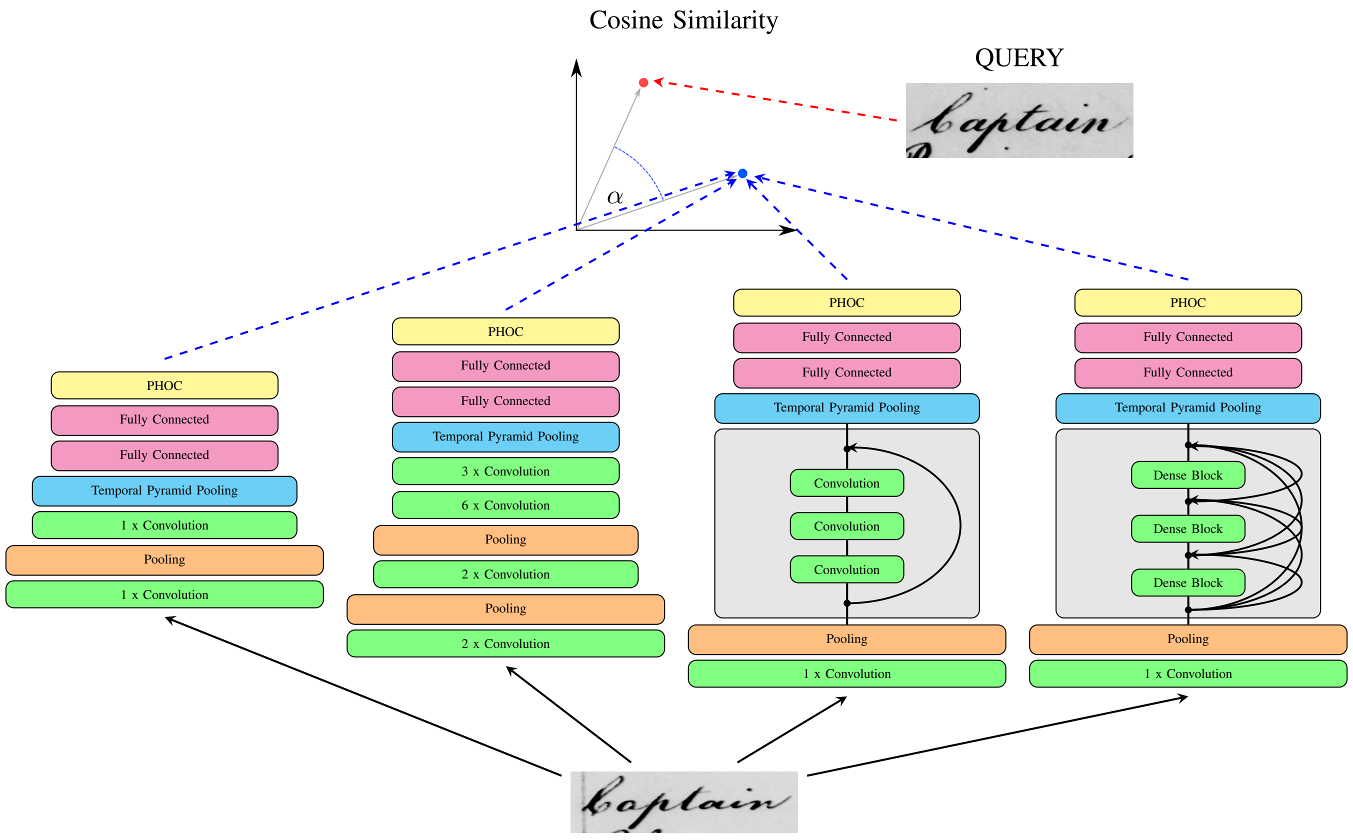

Figure 1. Overview of the explored CNN architectures. The networks are from left to right, PHOCLeNet, TPP-PHOCNet, PHOCResNet, and PHOCDenseNet. The CNNs are trained to predict a desired word string embedding generated from the annotation. Word spotting can then be performed in the embedding space through a simple nearest neighbor search using the cosine similarity as distance metric. Please note, that due to space issues this overview is not an exactly architecture description especially for the residual bottleneck and dense blocks. The exactly description is given in Sec. III.

Rectified Linear Unitss (ReLUs). The reason behind this is the vanishing gradient problem (cf. e.g. [19]): Even though the LeNet architecture is shallow compared to current deep learning architectures, it still employs four layers which use the hyperbolic tangent as activation function. Replacing them with ReLUs eliminates the vanishing gradient problem as well as speeds up the training process. The second change is replacing the last pooling layer with a Temporal Pyramid Pooling (TPP) layer [5]. This allows the CNN to process images of different sizes. The last change made to the architecture are the size of the convolutional and fully connected layers. Here, we use the same changes as are done for the LeNet architecture in the Caffe deep learning library [20] and increase the trainable parameters of the CNN: The number of filters in the first convolutional layer is set to 20 while for the second convolutional layer 50 filters are used. The hidden fully connected layers are enlarged to 500 neurons. While the network remains small from a parameter point of view compared to current architectures, it gains a considerable amount of power compared to the original version.

B. *TPP-PHOCNet*

The TPP-PHOCNet [5] is an enhanced version of the PHOCNet [21]. Its architecture is inspired by the successful VGG16 CNN [4]. The convolutional part consists of 13 convolutional layers with two max pooling layers inserted after the second and fourth convolutional layer. As is done in the original VGG16, all layers make use of 3×3 filter kernels. Compared to the LeNet and also the successful AlexNet [3], this choice of filter sizes serves as regularization as it substantially decreases the amount of parameters in the convolutional layers. While the original VGG16 also employed max pooling layers after the seventh and tenth convolutional layers, these max pooling layers are eliminated in the TPP-PHOCNet. The reason for this is that the word images used as input for the CNN are usually much smaller than the natural images the VGG16 was designed for. Removing them from the architecture allows for a smaller increase in receptive field for the later layers of the convolutional part. After the convolutional layer, the TPP-PHOCNet makes use of a TPP layer in order to obtain a fixed image representation for images of varying sizes. The MLP part of the architecture consists of two hidden layers with 4096 neurons each. This is in accordance with the AlexNet and VGG16 CNNs. As these two layers carry the most trainable parameters and are prone to overfitting, a dropout of 50% is applied to both hidden layers. In dropout, the output of a given neuron is randomly set to 0 during training. The probability for a neuron to be dropped out is a meta-parameter which has to be defined by the user. Using a dropout of 50% is a popular practice when using hidden layers of these sizes in current CNNs (cf. e.g. [3], [4], [22], [23]).

All convolutional and fully connected layers in the TPP-PHOCNet use ReLUs as activation function.

C. PHOCResNet

The PHOCResNet is a novel architecture we propose for word spotting. It is inspired by the successful ResNet50 architecture [15]. This CNN uses a 7×7 convolutional and a 3×3 max pooling layer in the beginning of the network which are then followed by 16 residual bottleneck blocks. The total number of convolutional layers in the CNN is 49. The ordinal for the blocks are as follows: Block 1 to 3 have ordinal 256, blocks 4 to 7 have 512, blocks 8 to 13 have 1024 and the final three blocks have ordinal 2048. As the VGG16, the ResNet50 was designed for natural images as input. In order to adapt it to attribute-based word spotting, we make four changes to the CNN's architecture: The original ResNet50 architecture uses strided convolutions for down sampling after the blocks 3, 7 and 13 in addition to the down sampling obtained from the pooling layer. As is done for the TPP-PHOCNet, we reduce the number of down sampling operations due to the word images being comparably small in size. Thus, we only use the first max pooling layer in addition to strided convolutions after the third residual block. Second, we replace the average pooling used in the original ResNet50 with a TPP layer in order to account for images of varying size. The third change is made with respect to the fully connected part of the ResNet50. Pre-experimental evaluations showed that learning a PHOC representation with the ResNet50 is quite difficult as it employs only a single fully connected layer in the end of the architecture. We obtained substantially better results when replacing this single layer with an MLP as was used in the TPP-PHOCNet. It seems, that the MLP is more able to learn the PHOC representation of correlated binary values than a single fully connected layer, i.e., a perceptron. Thus, the same MLP is used for the PHOCResNet as is done for the PHOCNet and TPP-PHOCNet (cf. Sec. III-B). The last change made in the PHOCResNet architecture with respect to the ResNet50 is that we delete all batch normalization layers. The reason for this is of a technical nature: We compute the gradients for mini-batches of varying sizes by obtaining the gradients for each image separately and then averaging them. This means, that our effective batch size is 1 and the weights are updated after performing backprop b times where b is the batch size. The batch normalization layers would thus only be presented with mini-batches of size 1 which would then be used for determining the mean and variance. In addition, the feature maps obtained for the different mini batches have different sizes which would skew the computation of a global mean and variance value. Hence, we elect to not use batch normalization in our network.

D. PHOCDenseNet

The PHOCDenseNet architecture is inspired by the the DenseNet structure proposed in [7]. The network employs two densely connected blocks and a growth rate of 12. Pre-experimental evaluations showed, that different growth rates perform comparable well, while small growth rates has shown to be more parameter efficient. Before entering the first block, a convolutional layer with 32 output channels and a 2×2 average pooling layer are employed. Following the dense connectivity pattern, the first block consists of 30 convolutional layers with kernel sizes 3×3. For the second block 60 densely connected convolutional layers are used. The transition layer between both blocks employ a convolutional layer with kernel size 1×1 and a 2×2 average pooling layer. The convolutional layer compresses the number of feature maps by a factor of 0.5 Analogue to the *TPP-PHOCNet* architecture, the PHOCDenseNet makes use of a 5-level TPP layer in combination with a Multilayer Perceptron.

IV. Experiments

For the experiments with the four architectures we used three benchmark datasets described in Sec. IV-A and a evaluation protocol (Sec. IV-B) for segmentation-based wordspotting commonly used in the literature. In Sec. IV-C we describe the training setup with all hyper-parameter used for training. Afterwards we discuss the retrieval results achieved by our methods and compare the architectures in Sec. IV-D.

A. Datasets

We evaluate our method on three publicly available data sets. The first is the **George Washington (GW) data set**. It consists of 20 pages that are containing 4 860 annotated words. The pages originate from a letterbook and are quite homogeneous in their visual appearance. However, particularly for smaller words the annotation is very sloppy. As the GW data set does not have an official partitioning into training and test pages, we follow the common approach and perform a four-fold cross validation. Thus, the data set is split into batches of five consecutive documents each.

The second benchmark is the large **IAM off-line dataset** comprising 1 539 pages of modern handwritten English text containing 115 320 word images, written by 657 different writers. We used the official partition available for writer independent text line recognition. We combined the training and validation set to 60 453 word images for training, and used the 13 752 word images in the test set for evaluation. We exclude the stop words as queries, as this is common in this benchmark.

Botany in British India (Botany) is the third benchmark introduced in the Handwritting Keyword Spotting Competition, held during the 2016 International Conference on

Table I
COMPARISON OF THE DIFFERENT RESULTS OBTAINED FOR THE DIFFERENT EXPERIMENTS [%].

Architecture/Method	George Washington		IAM		Botany *Train III*	
	QbE	QbS	QbE	QbS	QbE	QbS
LeNet	69.78	80.68	13.55	27.86	32.27	21.82
TPP-PHOCNet	97.29	**98.47**	85.03	93.22	92.89	96.61
PHOCResNet	95.90	97.91	**88.35**	**94.69**	**95.92**	**98.53**
PHOCDenseNet	95.20	94.71	72.06	84.11	70.26	82.95
Deep Features [25]	94.41	92.84	84.24	91.58	–	–
Triplet-CNN [26][1]	**98.00**	93.69	81.58	89.49	–	–
AttributeSVM [6]	93.04	91.29	55.73	73.72	–	–

Frontiers in Handwriting Recognition. The training data of Botany was partitioned into three different training sets from smaller to larger (*Train I* 1684, *Train II* 5295, and *Train III* 21981 images). For our experiments we only use the *Train III* partition.

B. Evaluation Protocol

We evaluate the four CNN architectures for the data sets GW, IAM and Botany in the segmentation-based word spotting standard protocol proposed in [6]. For the QbE scenario all test images are considered as query which occur at least twice in the test set. For QbS only unique string are used as queries. The CNNs predicte an attribute representation for each given query, afterwards a nearest neighbor search is performed by comparing the attribute vectors using the *cosine similarity*. The Retieval lists are obtained by sorting the list items by their cosine distances.

We use the *mean Average Precision (mAP)* as the performance metric by computing a *interpolated Average Precision* for each query as recommended in [21].

C. Training Setup

For training the different CNNs, we use hyper parameters as suggested by [17]: All networks are trained using the BCEL and Adaptive Moment Estimation (Adam) [27]. As labels, we use PHOC vectors with levels 1, 2, 4, 8 and omit bi- or trigrams. The momentum values β_1 for the mean and β_2 for the variance are set to the recommended 0.9 and 0.999 respectively while the variance flooring parameter ϵ is set to 10^{-8}. The initial learning rate is set to 10^{-4} and divided by 10 after $70\,000$ training iterations for the George Washington Database (GW) and Botany. Training on these two data sets is carried out for a total of $80\,000$ iterations For the IAM Handwriting Database (IAM-DB) the step size is set to $100\,000$ and training is run for $240\,000$ iterations.

All networks are trained from scratch without the help of additional data. As initialization, we follow the strategy proposed in [28]: For each layer l the weights are sampled from a zero-mean normal distribution with variance $\frac{2}{n_l}$ where n_l is the total number of trainable weights in layer l.

Similar to [17], the images are not preprocessed but the pixel values are scaled such that black pixels have a pixel value of 1.0 while white pixels 0.0. We also augment the training set using the algorithm presented in [17]. This way, training images are generated such that there exists an equal amount of images per class used during training and the total amount of images is $500\,000$.

D. Results & Discussion

Table I lists the results for the QbE and QbS experiments on three benchmarks. As expected the LeNet perform poorly on all three benchmarks, since this architecture has the least parameters and way too few feature maps. With only 70 $(20{+}50)$ filter kernels in summation, the LeNet is not powerful enough to learn a rich feature representation. Even with more feature maps using only 500 neurons in both hidden layers on the fully connected part, the MLP would still perform poor. Nevertheless, with a relatively small CNN like PHOCLeNet a mAP of 80.68% for QbS on the GW dataset is achieved. One can argue that word images from the GW dataset, written by the same writer, have very similar visual appearances, and thus a CNN do not require an enormous amount of parameters to learn a mapping from visual to textual representations. The TPP-PHOCNet slightly outperforms the other three architectures on the GW dataset, but the PHOCResNet achieves the best results on the IAM database. If we compare the TPP-PHOCNet with the PHOCResNet, we can argue that the GW dataset has less word instances and less word images. In contrast to the PHOCLeNet, the PHOCResNet has to many parameters (over 144 million) whereas the TPP-PHOCNet contain 50 million. Experiments showed, that even decreasing the depth of the TPP-PHOCNet results in equal mAPs for QbE and QbS on the GW dataset. We assume that it is this over parameterization resulting in a degradation of a CNN as mentioned in [15].

Suprisingly the PHOCDenseNet performs poor on both IAM and Botany. Even though the PHOCDenseNet employs significantly more layers than the TPP-PHOCNet and PHOCResNet, it is not able to achieve comparable results. We argue that the main disadvantage of the proposed architecture is the combination of a TPP layer and a MLP. While the TPP-PHOCNet contains 512 feature-maps in its final convolutional layer, the PHOCDenseNet has 916. Although an increased number of feature-maps are generated, we argue that their representational power is comparable low. As discussed in [7], the classification layer of a DenseNet heavily focuses

[1]results obtained with additional annotated training data [26]

on the complex feature-maps created late in the network. Pre-experimental evaluations have shown that increasing the number of feature-maps, by simply adding layers to the network, does not increase the mAP nether in QbE nor in QbS. An increased number of feature-maps also corresponds to a heavy increase of parameters in the MLP. We believe, this overparameterization prevents the network of benefiting from an increased network deep.

V. Conclusion

In this work, we explored four CNN architectures on three standard word spotting benchmarks for Query-by-Example and Query-by-String. The experiments conducted show that deeper CNN architectures do not necessarely perform better on word spotting tasks. On the contrary, the recently proposed DenseNet [7] performs the worst on all three benchmarks excluding the PHOCLeNet. This leads us to the hyperthesis that nether increasing the depth nor the number of parameters in the convolutional part achieves significantly higher performance in word spotting. Instead, future research needs to focus on word embeddings incorporating more informations than only character occurence or position.

References

[1] R. Manmatha, C. Han, and E. Riseman, "Word spotting: a new approach to indexing handwriting," in *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 1996.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[4] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proc. of the Int. Conf. on Learning Representations*, 2015.

[5] S. Sudholt and G. Fink, "Evaluating word string embeddings and loss functions for CNN-based word spotting," in *Proc. of the ICDAR*, Kyoto, Japan, 2017, to appear.

[6] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *TPAMI*, vol. 36, no. 12, pp. 2552–2566, 2014.

[7] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] T. Rath and R. Manmatha, "Word spotting for historical documents," *Int. Journal on Document Analysis and Recognition*, vol. 9, no. 2–4, 2007.

[9] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós, "Efficient segmentation-free keyword spotting in historical document collections," *Pattern Recognition*, vol. 48, no. 2, pp. 545 – 555, 2015.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. Journal of Computer Vision*, vol. 60, 2004.

[11] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. European Conf. on Computer Vision*, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 143–156.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec 1989.

[13] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudk, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[14] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," *arXiv preprint arXiv:1507.06228*, 2015.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[16] K. He and J. Sun, "Convolutional neural networks at constrained time cost," *CoRR*, vol. abs/1412.1710, 2014. [Online]. Available: http://arxiv.org/abs/1412.1710

[17] S. Sudholt and G. A. Fink, "Attribute CNNs for Word Spotting in Handwritten Documents," *International Journal on Document Analysis and Recognition*, 2018.

[18] K. Fukushima, "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[19] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, and U. C. B. Eecs, "Caffe : Convolutional Architecture for Fast Feature Embedding," in *Proc. of the ACM Conference on Multimedia*, 2014, pp. 675–678.

[21] S. Sudholt and G. A. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Proc. of the ICFHR*, Shenzhen, China, 2016, pp. 277 – 282.

[22] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," in *Proc. of the European Conference on Computer Vision*, 2014, pp. 818–833.

[23] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *Proc. of the Int. Conf. on Learning Representations*, 2014.

[24] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A. Toselli, and E. Vidal, "ICFHR2016 Handwritten keyword spotting competition (H-KWS 2016)," in *Proc. of the ICFHR*, 2016, pp. 613–618.

[25] P. Krishnan, K. Dutta, and C. Jawahar, "Deep feature embedding for accurate recognition and retrieval of handwritten text," in *Proc. of the ICFHR*, 2016, pp. 289 – 294.

[26] T. Wilkinson and A. Brun, "Semantic and verbatim word spotting using deep neural networks," in *Proc. of the ICFHR*, 2016, pp. 307 – 312.

[27] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of the Int. Conf. on Learning Representations*, 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," in *Proc. of the Int. Conf. on Computer Vision*, 2015, pp. 1026–1034.