Evaluating Word String Embeddings and Loss Functions for CNN-based Word Spotting

Sebastian Sudholt and Gernot A. Fink
Department of Computer Science
Technische Universität Dortmund University
44221 Dortmund, Germany
Email: {sebastian.sudholt, gernot.fink}@tu-dortmund.de

***Abstract*—The recent past has seen CNNs take over the field of word spotting. The dominance of these neural networks is fueled by learning to predict a word string embedding for a given input image. While the PHOC (Pyramidal Histogram of Characters) is most prominently used, other embeddings such as the Discrete Cosine Transform of Words have been used as well. In this work, we investigate the use of different word string embeddings for word spotting. For this, we make use of the recently proposed PHOCNet and modify it to be able to not only learn binary representations. Our extensive evaluation shows that a large number of combinations of word string embeddings and loss functions achieve roughly the same results on different word spotting benchmarks. This leads us to the conclusion that no word string embedding is really superior to another and new embeddings should focus on incorporating more information than only character counts and positions.**

I. Introduction

The field of word spotting has gained an increasing amount of research interest lately. The goal in word spotting is to retrieve parts of a document image collection with respect to a given query. Often times, this query representation is either an image (*Query-by-Example, QbE*) or a string defining the sought after word (*Query-by-String, QbS*). Recently, both segmentation-based (i.e. a word level segmentation is available) and segmentation-free (i.e. entire document images are used without any segmentation) word spotting scenarios have been investigated.

The best performing methods in segmentation-based word spotting currently all make use of the same framework: Word strings and word images are projected into an embedding space through suitable transformations which are estimated from annotated training data. This way, QbE as well as QbS word spotting can be performed through a simple nearest neighbor search. While early work in this regard made use of classical, handcrafted features and SVMs [1], *Convolutional Neural Networks (CNN)* have been used in a variety of ways recently to predict embeddings for word images [2]–[4].

As of now, embedding-based approaches in conjunction with deep neural networks represent the state-of-the-art in segmentation-based word spotting. The question that arises, though, is: Which word string embedding is the best and what is the most suitable way to train a system to predict this embedding?

In this work, we investigate different word string embeddings and loss functions for training a CNN in an end-to-end fashion. Our experiments are conducted in a segmentation-based scenario. Furthermore, we introduce a new global pooling layer for CNNs, which is especially suitable to be used for word images. We dub this new layer *Temporal Pyramid Pooling layer (TPP)*.

II. Related Work

Besides sequential models, e.g. [5]–[7], recent approaches for word spotting made heavy use of holistic representations of word images [1], [8], [9]. Here, especially local descriptor-based approaches were successfully applied to word spotting problems. In [10] the authors propose to build a spatial pyramid on top of a SIFT-based Bag-of-Features representation in order to form word image descriptors. In [11] this concept is extended to learn a common space between spatial pyramids and word strings to allow for Query-by-String word spotting.

Very influential work concerning common spaces for word spotting was presented in [1]. Here, the authors propose to project word images into a word string embedding space. As word string embedding, the authors propose to use a binary attribute representation which they name *Pyramidal Histogram of Characters (PHOC)*. They employ an ensemble of SVMs to learn predictions for each element of the PHOC individually. Each SVM takes a Fisher Vector representation of the word image and predicts one dimension of the PHOC. The presented approach could be shown to achieve state-of-the-art results.

Recently, the concept of embedding word strings into a vector space was extended by incorporating CNNs. In [2] the authors replace the Fisher Vector with a deep CNN trained on synthetic word image which is then fine-tuned to the specific data sets. The CNN acts as a feature extractor for the ensuing AttributeSVMs. In [3] the authors train a Triplet-CNN as feature extractor. For predicting a PHOC from these features, they employ an MLP. In addition to the PHOC, the authors propose and evaluate a different word string embedding called *Discrete Cosine Transform of Words (DCToW)*. In [4] a deep CNN called PHOCNet is used to learn the PHOC representation for a given word image. In contrast to [2] and [3], the training of the PHOCNet is conducted in an end-to-end fashion, thus optimizing features and prediction simultaneously. For this, the PHOC is interpreted as an n-out-of-k encoding which can then be learned through binary logistic regression.

III. METHOD

In this section, we describe our method to learn word string embeddings with a CNN. We base our system on the recently proposed PHOCNet [4] as it allows for easily learning a binary string embedding in an end-to-end fashion and achieves state-of-the-art results. Depending on the embedding to be learned, we make slight alterations to the PHOCNet architecture, namely replacing the loss function used for training. In the following we list all word string embeddings and loss functions used in our experiments. In addition, we design a new pooling layer, similar to the Spatial Pyramid Pooling layer which is especially suitable for word images.

A. Word String Embeddings

Pyramidal Histogram of Characters The PHOC representation [1] is the one most commonly used in the embedding framework for word spotting, e.g. [1]–[4]. It is a binary histogram of character occurrences at different splits of the string. For example, at the first layer of the PHOC, the string is not split at all but rather the presence of each character is indicated in a binary histogram. At the second level, the string is split into two equally sized portions along the middle and the binary histogram of character presence is built for each side. A character's membership to each partition is determined as follows: Assume all characters have equal width then a character belongs to a partition if at least 50% of it overlap with the partition.

A PHOC can principally feature as many layers as desired. The authors, however, argue to ignore the first layer as it is not discriminative for strings featuring the same characters such as *silent* and *listen* [1]. Instead, they augment the PHOC with two binary histograms showing presence or absence of bigrams in the left and right partition of the word.

In a number of pre-experiments, we found the first PHOC-level to actually help in discriminating between PHOCs predicted from a CNN as the scenario described above rarely occurs. In contrast, we found the bigrams to not have a significant impact on the entire system. Hence, for our experiments, we use a 5-level PHOC of partitions 1, 2, 3, 4 and 5 ignoring bigrams.

Discrete Cosine Transform of Words Another recently used word string embedding is the *Discrete Cosine Transform of Words (DCToW)* [3]. This representation is obtained by first representing each character as a one-hot encoded vector with respect to the alphabet. Stacking all vectors into a matrix, a discrete cosine transform is applied per row and all values but the highest three per row are discarded. The remaining values are concatenated into a vector to form the DCToW descriptor.

Spatial Pyramid of Characters The *Spatial Pyramid of Characters (SPOC)* [12] can be seen as a multinomial generalization of the PHOC. Here, instead of showing binary presence or absence of each character in each split, the corresponding characters are counted. At the first level, a simple *Bag-of-Characters (BoC)* representation is created meaning a histogram of character counts for each character of a given alphabet. For every consecutive level, the string is split into a number partitions equivalent to the level (e.g. two partitions at the second level, three at the third and so on). For each such partition a BoC is generated. Finally all BoC representations are concatenated to form the SPOC.

We make a slight modification to the original SPOC proposed in [12]. Instead of normalizing the individual BoC histograms or assigning partial values, we use the same strategy for counting characters to partitions as is done in the PHOC. Considering each character to have equal width, a character is counted as belonging to a partition if its overlap with the respective split is at least 50%. Essentially, the SPOC in our approach is a PHOC with counts instead of binary presence for each character in each partition. To the best of our knowledge, the SPOC has not been used in conjunction with a CNN or in a word spotting context before.

B. Loss Functions

Having defined the word string embeddings, this section describes which loss functions are used for the different experiments to train the PHOCNet.

Binary Logistic Loss When learning to predict binary n-out-of-k representations such as the PHOC, an oft-used loss function is the Binary Logistic Loss, which is also known as binary or sigmoid cross entropy loss. This loss can be motivated from a probabilistic point of view: Let θ be the parameters of a CNN and $\mathbf{y}^{(i)}$ the desired output for a given input $\mathbf{x}^{(i)}$. Then the goal is to find the parameters $\hat{\theta}$ which maximize the likelihood of predicting $\mathbf{y}^{(i)}$ for $\mathbf{x}^{(i)}$ for a total of n samples:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{n} p(\mathbf{y}^{(i)} | \mathbf{x}^{(i)}, \theta) \tag{1}$$

In order to derive a loss function from this expression, the assumption is made that the elements of the binary label vector $\mathbf{y}^{(i)}$ are pairwise independent. While this assumption is of course violated in the case of the PHOC, it can be empirically shown, that the approach still works well in practice [4].

Treating each $\mathbf{y}$ as a vector of pairwise independent and Bernoulli distributed variables, the Binary Logistic Loss computes to

$$l_{BL}(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{1}{D} \sum_{d=1}^{D} \left[y_d \log \hat{y}_d + (1 - y_d) \log(1 - \hat{y}_d) \right] \tag{2}$$

where $\hat{\mathbf{y}}$ is the prediction form the CNN, D its dimensionality and y_d the value at the d-th dimension of the vector $\mathbf{y}$.

Cosine Loss As the binary logistic loss is designed for learning a number of Bernoulli-distributed variables, it cannot be used for real-valued embeddings such as the DCToW. In order to allow a CNN to learn real-valued word string embeddings, the authors in [3] propose to make use of the Cosine Embedding Loss

$$l_{CE}(\hat{\mathbf{y}}, \mathbf{y}, m) = \begin{cases} 1 - \cos(\hat{\mathbf{y}}, \mathbf{y}) & \text{if } m = 1 \\ \max(0, \cos(\hat{\mathbf{y}}, \mathbf{y}) - \gamma), & \text{otherwise} \end{cases} \tag{3}$$

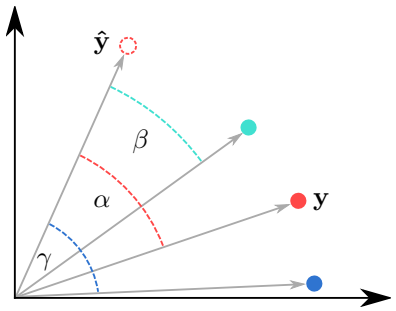

Fig. 1: Toy example showing a possible configuration where learning with the Cosine Embedding Loss is hampered due to disadvantageous weight initialization.

Here, $\hat{\mathbf{y}}$ is the output of the CNN to be trained, $\mathbf{y}$ an embedding vector and m an indicator function evaluating to 1 if the embedding corresponds to the input $\mathbf{x}$ responsible for obtaining $\hat{\mathbf{y}}$. Figuratively speaking, the loss encourages the CNN to minimize the angle between $\hat{\mathbf{y}}$ and $\mathbf{y}$ if the input and the embedding match. Otherwise the angle is maximized. In [3] the authors randomly sample word images and embeddings in order to create pairs to train the CNN with. For this they generate both corresponding (word image and string embedding match) and non-corresponding (word image and string embedding do not match) samples.

While this approach does allow for learning real-valued embeddings, we believe that the loss imposes a certain restriction on the training process: While the actual goal is to have the CNN output the desired embedding $\mathbf{y}$, it is additionally asked to maximize the distance of the predicted embedding to all other embeddings. Depending on the weight initialization this might cause the CNN to exhibit a slow learning behavior or even get stuck during training. A simple example for this is given in figure 1. Here, $\hat{\mathbf{y}}$ is the prediction of a CNN for a given point after initialization and $\mathbf{y}$ the desired, i.e, matching target. All other points are non-matching targets. During training, the Cosine Embedding Loss tries to simultaneously minimize α while maximizing β and γ. Thus training stalls as the two objectives cancel each other out. While this is only a toy example and the shown "blocking" behavior can be expected to be not as severe in higher dimensional spaces, it might still lead to slower learning and inferior results.

Thus, instead of the Cosine Embedding Loss, we propose to use the Cosine Loss

$$l_{\cos}(\hat{\mathbf{y}}, \mathbf{y}) = 1 - \cos(\hat{\mathbf{y}}, \mathbf{y}) \tag{4}$$

which is essentially the part of l_{CE} for matching pairs. Again, $\hat{\mathbf{y}}$ is the vector predicted from the CNN and $\mathbf{y}$ the desired embedding. This loss was previously used in [13] for learning an information-theoretical label embedding, which is also real-valued. As in other supervised tasks, the loss only minimizes the error between the label vector and the CNN's output (in this case the error is the angle between the two vectors), thus circumventing the problem shown in figure 1.

For training a CNN with the Cosine Loss, the partial derivative $\frac{\partial l_{\cos}}{\partial \hat{\mathbf{y}}}$ has to be computed. In [13] the authors make use of a computational graph to calculate the gradients for backpropagation. Thus they do not have to compute an analytical derivative for $l_{\cos}$. In order to allow the use of the Cosine Loss in frameworks not relying on a computational graph, we give an analytical expression for the partial derivative of the Cosine Loss w.r.t. the output of the CNN:

$$\frac{\partial l_{\cos}(\hat{\mathbf{y}}, \mathbf{y})}{\partial \hat{\mathbf{y}}} = \frac{1}{||\hat{\mathbf{y}}||\,||\mathbf{y}||}\left(\frac{\hat{\mathbf{y}}^T\mathbf{y}\cdot\hat{\mathbf{y}}}{||\hat{\mathbf{y}}||^2} - \mathbf{y}\right). \tag{5}$$

Using the partial derivative, we can backpropagate the error in the usual stochastic gradient descent (SGD) framework to train a CNN. When using the Cosine Loss, we also remove the sigmoid activation from the last layer of the PHOCNet as the desired output is in $\mathbb{R}^D$ instead of $\{0,1\}^D$.

In all of our experiments, we do not make use of the Euclidean loss. We argue that this loss is ill suited, as the Euclidean distance is not a suitable distance measure for high-dimensional data such as the presented attribute representations. This is due to the fact that all data points in high dimensional spaces have roughly equal pairwise Euclidean distances [14], [15]. Thus we exclude the Euclidean loss from consideration.

C. Temporal Pyramid Pooling Layer

In our experiments, we make a modification to the PHOCNet architecture by introducing a new kind of pyramidal pooling layer. This layer is independent of the loss or embedding and can be plugged into any CNN architecture.

The Spatial Pyramid Pooling layer (SPP) [16] in the PHOCNet is used in order to enable the CNN to process arbitrarily sized input images. The SPP layer subdivides its input feature maps into an equal amount of horizontal and vertical bins, similar to the original spatial pyramid [17]. When dealing with spatial pyramids built on top of local image descriptors, it was shown that subdividing a word image along the horizontal axis is more important than along the vertical axis [11]. Common descriptor-based approaches thus feature a larger number of bins along the horizontal axis than the vertical axis [9], [18], sometimes even ignoring these subdivisions altogether [10], [19].

Going back to the SPP layer, we expect a similar principle, namely that subdivision of feature maps is more important along the horizontal than vertical axis when dealing with word images. Hence, we propose a new layer type based on the principles of the SPP layer which ignores vertical bins. This type of layer is especially suitable when dealing with images of varying size where the content is of a sequential nature. As this layer is designed to pick up features appearing at different temporal positions of such sequences, we term this layer *Temporal Pyramid Pooling Layer (TPP)*. Figure 2 exemplarily visualizes the TPP layer. Similar to the SPP layer, we pool each feature map of the last convolutional layer in order to create a fixed size representation from a number of arbitrarily sized feature maps. However, the bins are only partitioned

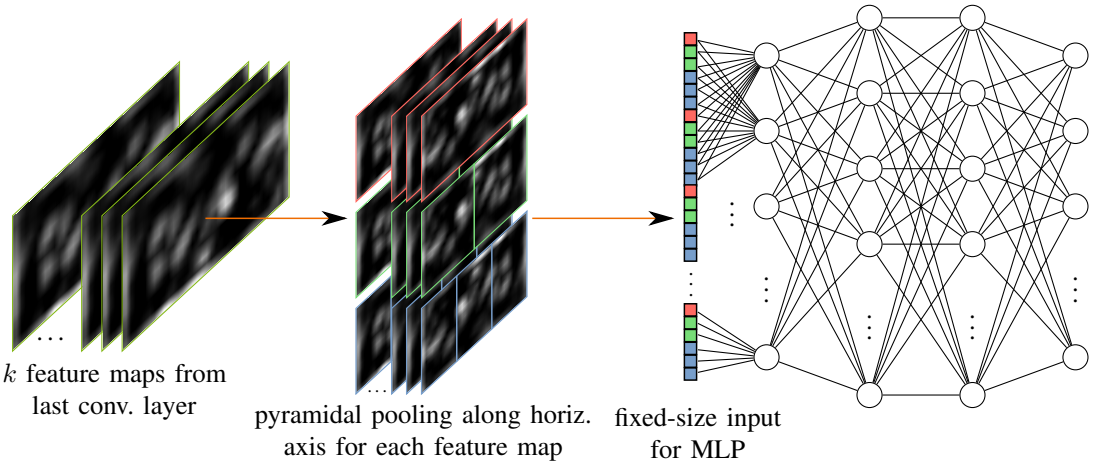

Fig. 2: The figure exemplarily visualizes the principle behind the Temporal Pyramidal Pooling layer. For each of the k feature maps of the last convolution layer the TPP layer generates pooling regions along the horizontal axis. Max pooling is applied for each region and the results are concatenated. The resulting vector has a fixed size and can be processed by an MLP.

along the horizontal axis. The figure shows the example of a 3-level TPP layer. The fixed output dimensionality of this layer is the sum over all bins for a single feature map times the number of input feature maps. In general, a number of pooling strategies may be used in the TPP layer, e.g. max, average or stochastic. In practice, however, we use max pooling as it constantly achieves good results. In the following, we refer to the PHOCNet using our new TPP layer as TPP-PHOCNet.

IV. Experiments

A. Data Sets

We evaluate our word spotting systems on five publicly available data sets. The **George Washington (GW)** data set has become a standard benchmarks for word spotting. It consists of 20 pages from a letter book by George Washington. The corresponding annotation contains word level bounding boxes and transcriptions for 4860 words. As there exists no official training and test splits, we follow the common approach by performing a fourfold cross validation, using the splits proposed in [1].

Originally proposed as a handwriting recognition benchmark, the **IAM-DB** [20] data set has recently enjoyed an increased use as word spotting benchmark as well. It contains a total of 115 320 words from 657 different writers. We use the official writer independent text line recognition partition to split the data set into training and test sets. As is common for this benchmark, we exclude the official stop words as queries but keep them as distractors [1].

The third data set used is the **Esposalles database** [21], an ancient marriage license register containing documents from the 15th to 20th century. Again, we use the official partition which splits the database into 32 052 training and 12 048 test word images.

The last two data sets, **Botany** and **Konzilsprotokolle**, were used as benchmark in the 2016 Keyword Spotting Competition [22]. They come with a dedicated set of queries for QbE and QbS. For the competition, the training data was partitioned into a small, medium and large set in order to investigate the effects of training data size on the proposed systems. For our experiments, we use the largest training partition for both data sets (`Train III`). This gives us 16 686 training images for Botany and 9 102 for Konzilsprotokolle. The test sets contain 3 318 word images for Botany and 3 891 for Konzilsprotokolle.

B. Evaluation Protocol

Our experiments for the data sets GW, IAM-DB and Esposalles follow the de-facto standard protocol for segmentation-based word spotting originally proposed in [1]: The training partition of each data set is used to train a single TPP-PHOCNet. For QbE, each word in the test partition is used once as query to rank all remaining word images. For this, a word string embedding is predicted from the TPP-PHOCNet for the query as well as all other test words. Retrieval is then performed by running a nearest neighbor search in the word string embedding space. The distance metric used here is the cosine distance which we found to perform better than the Bray-Curtis dissimilarity used in [4]. For QbS, each unique transcription from the test set is used to serve as query. The string embedding is directly computed from the transcription and retrieval is run as for QbE.

As the data sets Botany and Konzilsprotokolle come with a dedicated query set for both QbE and QbS, we follow the ICFHR 2016 Keyword Spotting Competition protocol [22] for these two data sets. This protocol only slightly deviates from the protocol defined above in that no word from the test set is used as query but rather the dedicated query images or strings.

As performance metric for a single query we use the interpolated *Average Precision (AP)*

$$AP = \frac{\sum_{i=1}^{n} p(i) \cdot r(i)}{t} \tag{6}$$

where n is the length of the retrieval list, $p(i)$ is the precision of the retrieval list if cut off after i elements and $r(i)$ is

TABLE I: Results for the QbE and QbS experiments in mAP [%]

Method	GW		IAM-DB		Esposalles		Botany		Konzilsprotokolle	
	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS	QbE	QbS
Binary Log. Loss + PHOC	97.75	97.50	83.38	92.59	96.93	**94.33**	**91.23**	**95.06**	**97.70**	**97.28**
Cosine Loss + PHOC	97.96	97.92	82.74	**93.42**	97.10	94.32	80.81	90.15	96.42	94.63
Cosine Loss + SPOC	97.78	**98.02**	82.17	92.64	97.05	94.07	80.37	89.58	96.43	95.88
Cosine Loss + DCToW	97.98	97.65	70.44	83.02	97.11	93.75	79.36	88.68	96.61	94.83
PHOCNet [4]	96.71	92.64	72.51	82.97	**97.24**	93.29	89.69[1]	74.47[1]	96.05[1]	94.20[1]
Attribute SVM [1]	93.04	91.29	55.73	73.72	–	–	75.77[1]	65.69[1]	77.91[1]	82.91[1]
Deep Feat. Embedding [2]	94.41	92.84	**84.24**	91.58	–	–	–	–	–	–
Triplet-CNN [3]	**98.00**[2]	93.69[2]	81.58[2]	89.49[2]	–	–	54.95[1]	3.40[1]	82.15[1]	12.19[1]

an indicator function computing to 1 if the i-th element of the retrieval list is relevant with respect to the query and 0 otherwise. Finally, t is the total amount of relevant elements in the database. Please note that the recall for each query is always 100% as every element in the database is returned in the retrieval list. The performance for an entire data set is determined by computing the *mean Average Precision (mAP)* over all queries.

C. Training Setup

For creating the different word string embeddings (cf. section III-A) we always use all characters present in the training set as alphabet. Thus the PHOC and SPOC exhibit different dimensionalities, depending on the data set and supplied annotation. All TPP-PHOCNets make use of a 5-level TPP layer and are trained with standard stochastic gradient descent (SGD) using momentum and weight decay. The meta-parameters used for SGD in large parts follow the parameters used in [4]. We use a batch of 10 images, momentum of 0.9 and weight decay of $5{\cdot}10^{-5}$. We train all PHOCNets for 80 000 iterations except for the IAM-DB where we train for 240 000 iterations. Here, an iteration means computing the gradients for a single batch and updating the weights accordingly. The initial learning rate is determined by choosing the maximum value at which the loss starts to converge. This approach yields a learning rate of 10^{-4} for all systems using the Binary Logistic Loss. For CNNs using the Cosine Loss, the initial learning rate is 0.01 for all experiments but the ones on IAM-DB. Here, the initial learning rate is 0.1. During training, the learning rate is divided by 10 after 70 000 iterations for each experiment.

D. Results & Discussion

Table I lists the results for the QbE and QbS experiments on the five benchmarks. In addition, figure 3 displays the evolution of the mAP for the QbE experiments over the course of training. As can be seen in the table and the plots, the different embeddings and loss functions achieve very similar results. Especially for the benchmarks yielding close to 100% mAP (GW, Esposalles, Konzilsprotokolle) the results are almost identical. The time at which learning starts to really achieve good results can vary for these benchmarks but they eventually converge to roughly the same result. For Botany, the combination of PHOC and Binary Logistic Loss outperforms the other systems. However, increasing the amount of training iterations to a maximum of 150 000, we achieved similar results using the combination of Cosine Loss and SPOC (QbE: 87.67, QbS: 94.87). We expect a similar behavior with the other Cosine Loss-based systems as well. For IAM-DB, the DCToW embedding performs worse than the others. As the training loss is already quite low here, we expect that the result achieved with the DCToW embedding won't be improved by allowing more training iterations.

Overall it can be seen, that integrating the TPP layer into the PHOCNet architecture leads to a performance gain over the original architecture. We are able to either achieve comparable or better results on all benchmarks with respect to the current state-of-the-art.

As almost all embeddings and loss functions achieve the same results on the five different benchmarks, we conclude that no combination of embedding and loss function is superior to another. Likewise, none of the evaluated word string embeddings is easier or harder to learn given an appropriate loss function.

The reason for the SPOC descriptor not performing better than the PHOC is that it does not carry a large amount of additional information compared to the PHOC. For example, taking all SPOCs for the GW data set, only 3% of the non-zero entries have a value greater than one, i.e. differ from the PHOC.

V. Conclusion

In this work, we presented an extensive evaluation of word string embeddings and loss functions for Query-by-Example and Query-by-String word spotting. The experiments conducted show that all embeddings and loss functions evaluated perform almost equally well on five standard word spotting benchmarks with minor outliers. This leads us to the hypothesis that the current word string embeddings in combination with CNNs have reached their performance limits. Future embeddings should incorporate more information than only character occurrence and position.

Additionally, we proposed a new pyramidal pooling layer called Temporal Pyramidal Pooling layer. Using this new layer as a connection between convolutional part and MLP in the

[1] results obtained from [22]

[2] results obtained with additional annotated training data [3]

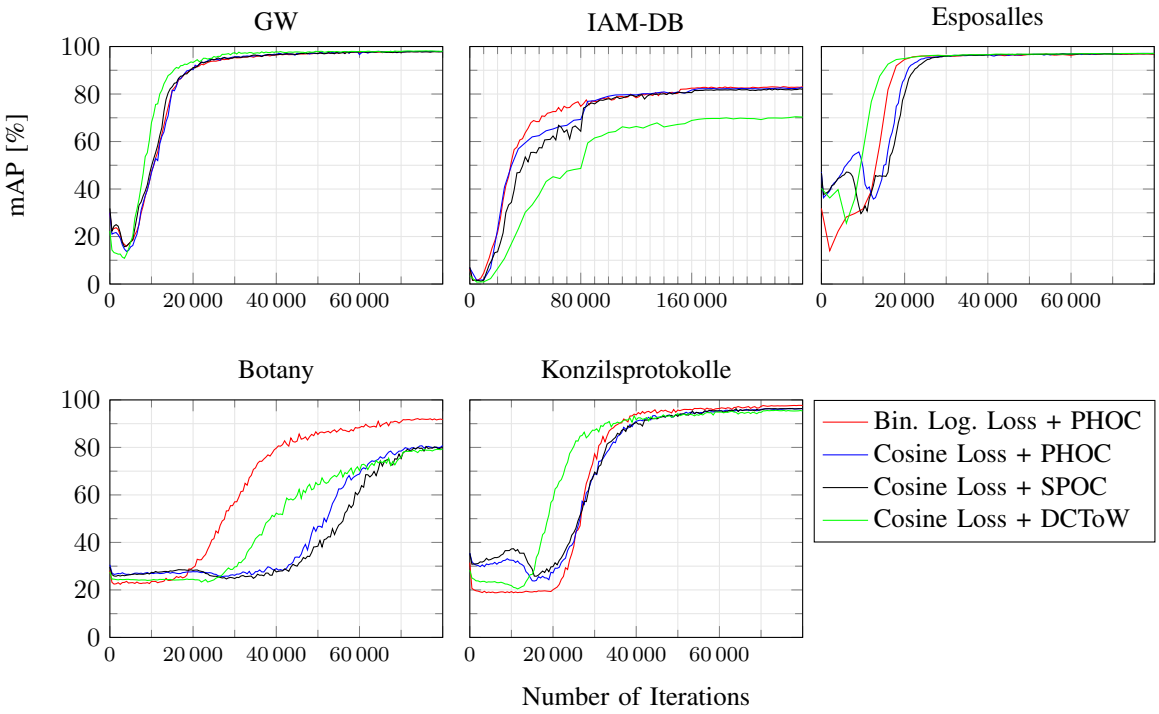

Fig. 3: Evolution of mAP for the five QbE experiments over the course of training using the different loss functions and embeddings.

PHOCNet architecture, we are able to achieve comparable results to or even beat the state-of-the-art.

The source code used for our evaluations is made publicly available at https://github.com/ssudholt/phocnet.git.

References

[1] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word Spotting and Recognition with Embedded Attributes," *Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[2] P. Krishnan, K. Dutta, and C. Jawahar, "Deep Feature Embedding for Accurate Recognition and Retrieval of Handwritten Text," in *International Conference on Frontiers in Handwriting Recognition*, 2016, pp. 289–294.

[3] T. Wilkinson and A. Brun, "Semantic and Verbatim Word Spotting using Deep Neural Networks," in *International Conference on Frontiers in Handwriting Recognition*, 2016, pp. 307–312.

[4] S. Sudholt and G. A. Fink, "PHOCNet : A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents," in *International Conference on Frontiers in Handwriting Recognition*, 2016.

[5] J. A. Rodríguez-Serrano and F. Perronnin, "A Model-Based Sequence Similarity with Application to Handwritten Word Spotting," *Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2108–2120, 2012.

[6] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A Novel Word Spotting Method Based on Recurrent Neural Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 211–224, 2012.

[7] L. Rothacker, M. Rusinol, and G. A. Fink, "Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents," in *International Conference on Document Analysis and Recognition*, 2013, pp. 1305–1309.

[8] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós, "Towards Query-by-Speech Handwritten Keyword Spotting," in *International Conference on Document Image Analysis*, 2015, pp. 501–505.

[9] ——, "Efficient segmentation-free keyword spotting in historical document collections," *Pattern Recognition*, vol. 48, no. 2, pp. 545–555, 2015.

[10] ——, "Browsing Heterogeneous Document Collections by a Segmentation-Free Word Spotting Method," in *International Conference on Document Analysis and Recognition*, 2011, pp. 63–67.

[11] D. Aldavert, M. Rusinol, R. Toledo, and J. Llados, "Integrating Visual and Textual Cues for Query-by-String Word Spotting," in *International Conference on Document Analysis and Recognition*, 2013, pp. 511–515.

[12] J. A. Rodriguez-Serrano and F. Perronnin, "Label Embedding for Text Recognition," in *British Machine Vision Conference*, 2013.

[13] F. Chollet, "Information-theoretical Label Embeddings for Large-scale Image Classification," *arXiv*, 2016.

[14] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the Surprising Behavior of Distance Metrics in High Dimensional Spaces," in *International Conference on Database Theory*, 2001, pp. 420–434.

[15] P. Domingos, "A Few Useful Things to Know about Machine Learning," *Communications of the ACM*, vol. 55, no. 10, p. 78, 2012.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *European Conference on Computer Vision*, pp. 346–361, 2014.

[17] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2169–2178.

[18] S. Sudholt and G. A. Fink, "A Modified Isomap Approach to Manifold Learning in Word Spotting," in *German Conference on Pattern Recognition*, 2015, pp. 529–539.

[19] C. Wieprecht, L. Rothacker, and G. A. Fink, "Word Spotting in Historical Document Collections with Online-Handwritten Queries," in *International Workshop on Document Analysis Systems*, 2016.

[20] U. V. Marti and H. Bunke, "The IAM-database: An English Sentence Database for Offline Handwriting Recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002.

[21] V. Romero, A. Fornés, N. Serrano, J. A. Sánchez, A. H. Toselli, V. Frinken, E. Vidal, and J. Lladós, "The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition," *Pattern Recognition*, vol. 46, no. 6, pp. 1658–1669, 2013.

[22] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A. H. Toselli, and E. Vidal, "ICFHR2016 Handwritten Keyword Spotting Competition (H-KWS 2016)," in *International Conference on Frontiers in Handwriting Recognition*, 2016, pp. 613–618.