Exploring critical aspects of CNN-based Keyword Spotting. A PHOCNet study.

George Retsinas[1,2], Giorgos Sfikas[1], Nikolaos Stamatopoulos[1], Georgios Louloudis[1] and Basilis Gatos[1]

[1] *Computational Intelligence Laboratory, National Center for Scientific Research "Demokritos", GR-15310 Athens, Greece*
Email: {georgeretsi,sfikas,nstam,louloud,bgat}@iit.demokritos.gr
[2] *School of Electrical and Computer Engineering, National Technical University of Athens, GR-15773 Athens, Greece*

***Abstract*—Deep convolutional neural networks are today the new baseline for a wide range of machine vision tasks. The problem of keyword spotting is no exception to this rule. Many successful network architectures and learning strategies have been adapted from other vision tasks to create successful keyword spotting systems. In this paper, we argue that various details concerning this adaptation could be re-examined, to the end of building stronger spotting models. In particular, we examine the usefulness of a pyramidal spatial pooling layer versus a simpler approach, and show that a zoning strategy combined with fixed-size inputs can be just as effective while less computationally expensive. We also examine the usefulness of augmentation, class balancing and ensemble learning strategies and propose an improved network. Our hypotheses are tested with numerical experiments on the IAM document collection, where the proposed network outperforms all other existing models.**

I. Introduction and Related work

In the recent years, convolutional neural networks (convnets, CNNs) have been established as the state-of-the-art models in a wide range of vision tasks. The task of searching for relevant word images given an image or text query, known in the related literature as keyword spotting or word spotting (KWS) [1], is no exception to this rule. Several variants of convolutional networks for keyword spotting have been proposed [2], [3] and employed with success.

In [2], the PHOCNet architecture has been proposed. PHOCNet is a typical feed-forward convolutional network, comprising pairs of convolutional and max-pooling layers, topped by fully connected layers leading to a stack of sigmoid outputs. The latter resembles the structure of the Pyramidal Histogram of Character (PHOC) vectors [1], [4]. PHOC vectors have been originally proposed in [4] as an attribute-based, pyramidal descriptor of the content of either a word image or a text string. PHOCNet inputs are segmented word images. A spatial pyramidal pooling layer (SPP) [5] bridges the convolutional layers to the fully connected layers and the output. In effect, this allows for accepting inputs of any size, with fixed-size output vectors. SPP can be seen as a pyramidal extension of zoning, used in various contexts in document image processing [6]. Temporal pyramidal pooling (TPP) has also been proposed, being effectively a special case of SPP where zones are partitioned over the horizontal axis only. A convnet employing a TPP layer in place of the SPP, yields slightly improved results [7].

Training neural networks can lead to the undesirable event of overfitting to their training set. Neural networks are notorious for supposedly being "data-hungry", in the sense of requiring vast amounts of data during training. Dropout is one standard technique that can improve network generalization. Another standard practice in the literature is data augmentation, used to artificially create a bigger training set. Augmentation has been used with its class balancing variant. All of these techniques have been employed in the PHOC-based convnets of [2], [7]. To the same end, alternative techniques using aggregations of intermediate layer activations as features have been proposed [6], [8].

In this paper, we aim to review some of the most crucial architectural and training choices made when training CNNs, assuming word-level segmented, keyword spotting context. We examine the pros and cons of each choice, and whether employing a particular technique outweighs the drawbacks, if any. In particular, we explore different strategies of input image resizing and combining with different strategies of spatial pooling, and we argue that a non-pyramidal, zoning approach can be equally if not more useful while being less computationally expensive. We check the extent of usefulness of augmentation with and without class balancing. Finally, we examine the usefulness of ensemble learning in this context [9]. All of the considered network parameters are evaluated with numerical experiments on the well-known IAM document dataset [1].

The remainder of this paper is organized as follows. In section II we review the baseline convolutional network that we shall assume for our experiments. In sections III and IV respectively, we define what architectural and training aspects of the model we choose to further examine, our motivation for choosing them and related implications. In section V we present our motivation for using an ensemble strategy and define ensemble learning in the context of KWS. We present numerical results in section VI, and review our conclusion and future work in section VII.

II. Baseline convolutional network

The core model of the considered keyword spotting scheme is a convolutional neural network, designed to produce a fixed-length word representation given an input word image. Representations are created for the query word image and the database images, then the database

978-1-5386-3346-5/18 $31.00 © 2018 IEEE
DOI 10.1109/DAS.2018.49

representations closest to the query representation are returned as the keyword spotting result. In this section we consider the *PHOCNet* model architecture proposed in [2] as our baseline system. Subsequently, we examine a number of ways to tweak the baseline model's architecture and review their impact on model performance.

The PHOCNet baseline architecture has been inspired by the "very deep" VGG architecture [10], proposed for image recognition. PHOCNet borrows from VGG the ideas of having convolutional layers with small receptive filters (3×3) and incrementally increasing the depth of successive convolutional layers, resulting to a relatively large number of layers.

From input to output, we can divide the PHOCNet model into three basic components (following the nomenclature of [11]): a) The network convolutional backbone. This is made up of stacks of convolutional layers paired with max-pooling layers. There are 2 convolutional layers followed by a 2×2 max-pooling layer, followed again by 2 more convolutional layers and a 2×2 max-pooling layer. These are followed by 9 convolutional layers. b) A spatial pyramidal pooling (SPP) layer [5]. c) The network head. This is a stack of fully-connected layers. There are in total 3 fully-connected layers in PHOCNet, trained with dropout.

All non-pooling layers are topped by ReLU non-linearities, except the output layer. The final fully-connected layer on the output is topped by per-unit sigmoid activations. Each sigmoid output corresponds to the probability estimate related to a particular word image attribute. This layout effectively reflects the Pyramidal Histogram of Characters (PHOC) representation [4], where attributes correspond to character unigrams or bigrams in different relative positions in the word. This representation has proven very effective, lending easily to definition extensions and enabling query-by-string keyword spotting.

The PHOCNet model achieved state-of-the-art keyword spotting results in tests with various document collections. While the model as a whole has proven to work very well, we argue that several choices concerning its architecture, the way it is trained, as well as the way the trained model is used for keyword spotting, are worth examining further. We present the considered choices in more detail in the three following sections (sections III, IV, V).

III. Considered Network Architecture Choices

Input size: One difference of the PHOCNet architecture versus the "very deep" VGG model [10] is the addition of the SPP layer (cf. discussion in sec. II) between the convolutional backbone and the network head. The SPP layer bears the special trait of transforming variable-size inputs into fixed-size outputs. In a sense, the whole model inherits this trait: the PHOCNet architecture indeed accepts variable-size word image inputs and produces outputs of the same size. This trait leads to the seemingly important advantage of not resizing the input image to a fixed size (as done in [10] for example, where all images are resized to 224×224 pixels). One could argue that not resizing the input is an obvious advantage, as resizing will change the aspect ratio and scale of the input. However, this advantage is not as clear as one may expect. We propose two different strategies concerning the input image, besides keeping the initial input size the same: a) resizing to a fixed height while keeping the aspect ratio of the original image, b) resizing to a fixed image size. It should be noted that a sufficiently deep network with a vast number of parameters and filters can model specific word text image structures (e.g. characters) at various aspect ratios in one class.

Spatial pooling strategy: The original SPP layer considers a pyramidal hierarchy of layers, where each layer pools from progressively less divisions of the input (in our case, the output of the convolutional backbone). An alternative layout of pooling has been proposed in [7], where the input is partitioned only across the horizontal direction. This layer has been dubbed Temporal Pyramidal Pooling (TPP) layer.

Since the standard layer operation at all levels of the hierarchy is max-pooling, the output of the SPP will contain largely redundant information. For example, consider an input that is split into 4 equal-size horizontal zones on one hierarchical level and 2 equal-size horizontal zones on the next level. All the information on a coarser level already exists on a finer level. The redundancy of the pyramidal pooling output is our motivation to consider only a single finer level of pooling. Hence, this strategy is akin to zoning [6], where a pooling operation is applied on a non-hierarchical partition of the input into spatial zones. Replacing pyramidal pooling with single-layer pooling / zoning results in a significant reduction of the number of learning parameters. This leads to a smaller parameter search space and in principle to an easier optimization. In the current model, replacing pyramidal pooling translates to a reduction of up to 20 million parameters.

Nevertheless, we have to note that pyramidal descriptions can be very useful, under a specific context. Bag of Visual Words (BoVW) approaches are one well-known example [1]. In that case, a pyramidal / hierarchical scheme is used to construct a descriptor; the redundancy trait of this descriptor helps in enabling matching in different levels of coarseness, which in turn lends to the robustness of the descriptor. However, in the context of convnets, where they are used as a non-output, intermediate layer, the picture is slightly different. The linear transformation of the pyramidal output before the non-linearity of the first fully connected layer can be trivially transformed to an equivalent linear transformation where we consider a single layer / zoning output. Hence, we can argue that in this case, pyramidal pooling does not have as much usefulness, if any.

We also consider a model where we dispense altogether with zoning and pyramidal pooling techniques. The alternative would be to simply flatten the output of the convolutional backbone and use that as input to the fully-connected layers. However, since these are necessary when working with variable-size input, this is applicable only

when we choose to resize to a fixed-size image input.

IV. Considered Network Training Choices

Augmentation vs no augmentation: Data augmentation (otherwise known as *data jittering*) [12] is a standard strategy employed for training neural networks when training examples are relatively scarce. The general idea is the application of a random transformation on each of the original data to produce new, artificial data. Typically a family of spatial transformations is employed to produce new data, with the choice of the family depending on the nature of the problem. In the work introducing PHOCNet [2], affine transformations have been used to produce warped word images. The affine parameters are constrained in order to have meaningful transformations considering text images.

Class balancing vs no class balancing: In [2], a class balancing strategy is used to mitigate the negative effect of under-represented classes (meaning in this context, words) in the training data. This is originally combined with the affine-based augmentation scheme. In particular, under-represented classes are used to produce augmented examples much more often than over-represented ones, so that eventually all training classes are equal with respect the number of their respective data.

The class balancing scheme can be easily decoupled from augmentation procedure as follows. When choosing the next datum to be used with the net training algorithm, we select an image at random. Instead of considering uniform probability for this random selection, we adjust the probability distribution so that asymptotically all classes are equally represented during training.

The problem with this type of class balancing is that the model, due to its structure, does not learn the target classes directly. Instead, classes are only learned indirectly as a result of learning to estimate sets of attributes [13] (i.e. unigrams/bigrams per word image position). We therefore argue that class balancing, at least in the sense previously described and employed in [2], is inadequate in terms of mitigating class under-representation.

V. Considered ensemble strategies

As solving for a convnet's optimal weights is treated as a local optimization task, different (random) initializations will in general lead to different local minima, and therefore different models. Combining a set of different models to obtain increased performance, known as *ensemble learning* is a strategy that has been employed for various pattern recognition tasks, including document image processing tasks [9], [14], [15]. The rationale behind ensembles is that different models will excel with respect to performance in different parts of the input space. Therefore, combining the models in some way, should lead to a better overall model. In the current work, we experiment with two different strategies for combining different trained convnets. The first strategy (*feature fusion*) consists of averaging over L convnet outputs given the same input word image. Concerning the second strategy (*late fusion*), we begin by computing L word spotting retrieval lists that correspond to the L convnets. Each retrieval list consists of the distance of the query example to the database instances, ordered in terms of increasing distance. Subsequently, we construct a single retrieval list, where we determine the position of each retrieved instance in the list by taking its minimum over the corresponding distances found on the L separate retrieval lists. We show that ensemble strategies, even when using a relatively low number of models L, can lead to significant accuracy improvement.

VI. Experimental Results

A. Experimental Setup

The keyword spotting experiments are evaluated on the challenging IAM dataset [1]. Other datasets (for example, the George Washington dataset [1]) may pose challenges on training due to their limited training sets, but the close to perfect existing results prove otherwise [7]. The IAM dataset contains a total of $115,320$ words written by 657 different writers. IAM dataset has been initially proposed for evaluating handwriting recognition methods. Lately, this dataset has become perhaps the most popular as well as reliable testbed for keyword spotting techniques. The large number of comprised words, as well as their diversity in writing style, make it ideal for training and testing deep neural networks. Generalization of the trained methods can be evaluated by selecting a test set that consists of writers unseen in the training set, as in [4]. Training and testing partitions are selected according to [4]. We focus on QbE KWS scenario and therefore images in the testing set that correspond to stop words or appearing only once are excluded from the query set but are kept as distractors.

Given the outputs of the PHOCNet architecture for the images consisting the testing set, i.e. the PHOC estimations, the retrieval list is computed by nearest neighbor search using the cosine distance [7]. As performance metric for a single query we use the interpolated Average Precision (AP). The performance on the whole test set is evaluated in terms of mean Average Precision (MAP) by computing the mean AP value for all the queries.

B. Training Setup

Training is performed assuming a binary logistic loss. Following [7], we use a batch size of 10 images. At this point, it is important to note that using batches of images of different sizes comes with the drawback of not fully utilizing GPU capabilities, as the batch cannot be described as a single 4-dimensional tensor. This results in slower training time compared to using same-sized images.

In contrast to previous works, we do not normalize the resulting loss of an image with respect to the batch size. Such normalization corresponds to updating the weights based on one image in terms of gradient magnitude. By avoiding this, we can achieve faster training convergence.

All networks are trained for $100,000$ iterations using Adam [16] with optimizer hyperparameters set to $\beta_1 = 0.9$, $\beta_2 = 0.99$ and weight decay $= 5 \cdot 10^{-5}$. An iteration corresponds to computing the gradients for a single batch

and updating the weights accordingly. The initial learning rate is 10^{-4} and is divided by 10 after $60,000$ training iterations.

Training of a deep architecture will converge to, in general different, local optima that may differ slightly in terms of KWS performance. In order to better compare the architecture and training choices that we explore in this work, our networks are re-trained 5 times from different random initializations and their results are averaged.

The PHOCNet architecture and training procedure have been implemented using the Pytorch [1] deep learning framework and are publicly available [2].

C. Results & Discussion

Input Size and Spatial Pooling Strategy

First, we aim to explore the impact of both the input size and the spatial pooling strategy. We consider three different scenarios with respect to the input size: 1) initial image size 2) fixed height while keeping aspect ratio and 3) fixed-sized images. We use fixed sized images or even fixed height in order to speed-up training and inference, since resizing usually reduces the image size. The fixed height was selected as 50, whereas the fixed width was selected as 100. This choice was not further explored. We also evaluate two different spatial polling strategies: 1) 5-level Temporal Pyramidal Pooling (TPP): Uniformly segment the output of the last convolutional layer along the x-axis into i zones at perform max pooling at each zone. This is repeated for 5 pyramid levels ($i = 1, \ldots, 5$) from fine to coarse level. 2) Zoning: use only the finer level of TPP (5th level, i.e. 5 zones), discarding the pyramidal scheme.

We trained a network for each combination of the aforementioned scenarios and the KWS evaluation results are summarized on Figure 1. Two main observations are derived: *1)* Changing the aspect ratio of an image does not affect the result. The seemingly preferable choice of keeping the initial image size has no performance impact, even though it has greater training and inference time requirements. *2)* Pyramidal approach of spatial pooling has no advantage in performance over a simple zoning approach. However, TPP generates $\times 3$ more features than zoning to be fed to the first fully connected layer, which results to an excessive number of redundant parameters (over 20 million parameters in our case).

These two observations can notably reduce the time of the training procedure. The impact on training time is presented in Table I. It is evident that using fixed-sized images along with zoning drastically reduces training time.

For each spatial pooling scenario, we also report the evolution of MAP while training the network. These results are depicted in Figure 2 for both TPP and zoning scenarios. Both figures share a similar behavior on training convergence over the different input size cases: resizing to a fixed-size provides a faster convergence compared to

[1] http://pytorch.org

[2] https://github.com/georgeretsi/pytorch-phocnet

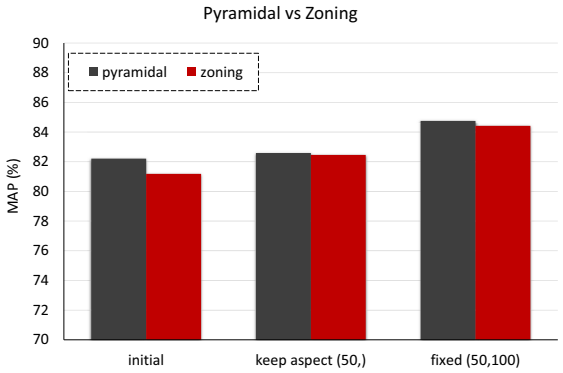

Figure 1. Performance of the trained models while different input size and spatial pooling strategies are considered.

Table I
TRAINING TIME PER BATCH

approach	time (sec)
TPP, initial-size	0.295
TPP, fixed aspect ratio	0.176
TPP, fixed-size	0.064
Zoning, initial-size	0.274
Zoning, fixed aspect ratio	0.163
Zoning, fixed-size	0.051

using the initial images. Furthermore, concerning convergence, we depict the MAP evolution for both spatial pooling strategies assuming fixed-sized images at Figure 3. The resulting evaluation curves are almost identical, supporting our claim that a pyramidal structure is redundant.

Up to this point, we have examined the effectiveness of the different spatial pooling schemes and have claimed that using fixed-size images does not affect the performance. However, the spatial pooling layer was introduced in order to handle images of arbitrary size. Therefore, a follow-up question should be: is an adaptive spatial pooling layer necessary when the input images are of the same size? To answer this question we will compare the performance of a PHOCNet architecture with zoning layer against a PHOCNet architecture where the convolutional output is simply flattened. The second architecture is infeasible on our GPU due to memory limitations, since the output of the convolutional part has almost $150,000$ dimensions. To overcome this problem we add a typical max pooling layer before the flattening operation. The aforementioned bare architecture gives 80.48% MAP, whereas using a zoning scheme gives 84.42% MAP. The difference in performance is significant, even though the bare network performs sufficiently well on this task. We speculate that this difference stems from the fact that using all the convolutional responses adds distractions, perhaps, unnecessarily enlarging the optimization search space. On the other hand, choosing a single response per filter and zone seems to suffice to describe the text structure, provided an adequate deep network.

Augmentation and Class-Balancing

Another important factor for training a deep CNN, such as PHOCNet, is the strategy of selecting and transforming the training images in order to be fed into the network.

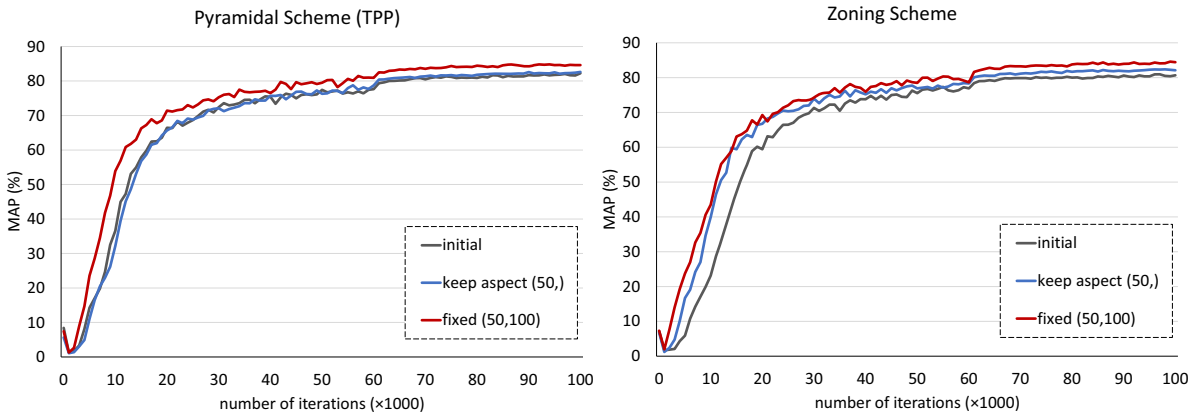

Figure 2. Evaluation curves of MAP performance during training under different input size

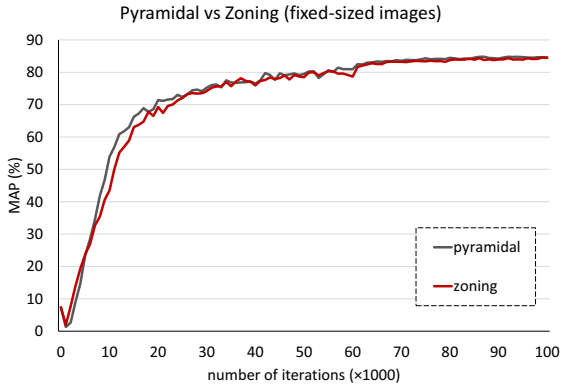

Figure 3. Comparison of evaluation curves during training for pyramidal and zoning schemes.

In the PHOCNet paper [2], there are two distinct choices regarding the training images: 1) data augmentation by (small) affine transformations and 2) class-balancing with respect to the unique word classes. We have trained models with or without utilizing these two strategies. For this experiment we assume zoning pooling at the output of convolution layers and fixed-sized input images. Figure 4 shows the resulting evaluation curves during the training process.

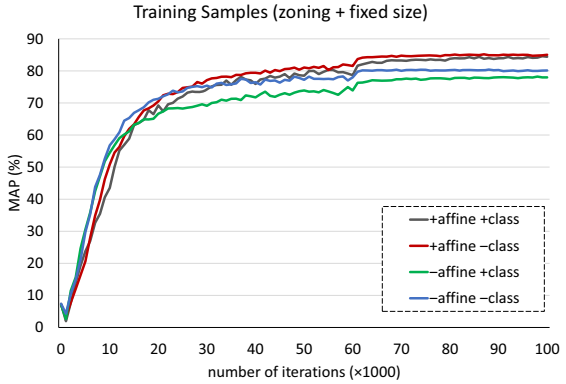

Figure 4. Evaluation curves under different training strategies with respect to affine augmentation and class balancing.

The results indicate that the PHOCNet architecture performs considerable well, even without any of these strategies. However, affine augmentation helps the network to generalize and produces a notable gain of 4% MAP. This was expected since small affine transformations appear frequently in text images are thus are a suitable for augmentation concerning KWS task. On the contrary, the class-balancing with respect to word classes has no impact when considering augmented images, while deteriorates the performance when no augmentation is involved. This behavior is seemingly surprising, but we should keep in mind that the network is trained with respect to PHOC attributes and not word classes.

Table II
MAP EVALUATION ON IAM DATASET

Method	MAP(%)
PHOCNet [2]	72.51
Attribute SVM [4]	55.73
Krishnan et al. [3]	84.24
Wilkinson et al. [17]	81.58
Deep PHOCNet features [8]	81.50
PHOCNet-TPP [7]	83.38
PHOC + bare-architecture	80.48
PHOC + zoning	84.42
5-ensemble, feature fusion	86.94
5-ensemble, late fusion	**87.48**

Using Ensembles of Convnets

We train $L = 5$ diffent models of the same architecture starting from different initializations and use them in order to evaluate two different ensemble strategies: *feature fusion* and *late fusion*.

The evaluation of the ensemble variants are presented in Table II, along with the evaluation of some baseline models from this work and the existing state-of-the-art approaches on IAM dataset. Baseline models *PHOC+bare-architecture* and *PHOC+zoning* are used as an indication of the capabilities of the PHOCNet architecture. Bare-architecture corresponds to discarding the spatial pooling layer and resembles the common CNN architecture. The *PHOC+zoning* is already on par with the best performing network so far [3] and is the network that we use for the ensemble approaches. The ensemble strategy utilizes

the generated models and produces results that clearly outperform all state-of-the-art approaches. Even though both fusion techniques provide a significant boost in performance, late fusion performs better than feature fusion, as the former tries to select the best output out of the five while the latter combines the generated outputs into a single PHOC estimation.

VII. Conclusion and Future work

We have studied a number of important aspects concerning the architecture and training of a class of convnets used for keyword spotting. The presented experiments help us draw some insightful conclusions. First, we conclude that using images of arbitrary size does not have any positive impact in performance compared to using images of fixed-size, while fixed-sized images greatly benefit the training time of the network. We have noted that the pyramidal pooling version of the spatial pooling layer generates a highly redundant output. Based on this observation, we have proposed that spatial pyramidal pooling can be replaced by a simple zoning pooling strategy, which has the effect of drastically reducing the number of network parameters. The zoning pooling approach appears to be more effective compared to simply flattening the convolutional output.

Concerning network training, we have validated that augmenting the training set with extra, affine-transformed versions of the training words improves performance. Class balancing, in the sense discussed here, has proven to be ineffective. We can perhaps envisage an improved version of class balancing as future work. Such a version of class balancing should ideally take into account the fact that the network does not learn word classes directly, but instead learns word attributes.

We have proposed a model where we integrate our conclusions on the better architecture and training strategies. Furthermore, we enrich the propose model by employing ensemble learning to construct an improved KWS system. The end model outperforms all existing state-of-the-art methods, forming a new baseline on the IAM dataset. As future work, we would like to examine the performance of more advanced ensemble strategies [9], [14].

Acknowledgment

This work has been supported by the EU project READ (Horizon-2020 programme, grant Ref. 674943).

References

[1] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou, "A survey of document image word spotting techniques," *Pattern Recognition*, vol. 68, pp. 310 – 332, 2017.

[2] S. Sudholt and G. A. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Proceedings of the* 15^{th} *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 277–282.

[3] P. Krishnan, K. Dutta, and C. V. Jawahar, "Deep feature embedding for accurate recognition and retrieval of handwritten text," in *Proceedings of the* 15^{th} *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 289–294.

[4] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[6] G. Sfikas, G. Retsinas, and B. Gatos, "Zoning aggregated hypercolumns for keyword spotting," in *Proceedings of the* 15^{th} *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 283–288.

[7] S. Sudholt and G. Fink, "Evaluating word string embeddings and loss functions for cnn-based word spotting," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[8] G. Retsinas, G. Sfikas, and B. Gatos, "Transferable deep features for keyword spotting," in *IWCIM, in conjuction with EUSIPCO*, 2017.

[9] G. Brown, "Ensemble learning," in *Encyclopedia of Machine Learning*. Springer, 2011, pp. 312–320.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv preprint arXiv:1703.06870*, 2017.

[12] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *CoRR*, vol. abs/1405.3531, 2014.

[13] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for attribute-based classification," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 819–826.

[14] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] C. Tensmeyer and T. Martinez, "Document image binarization with fully convolutional neural networks," *International Conference on Document Analysis and Recognition (ICDAR)*, 2017.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] T. Wilkinson and A. Brun, "Semantic and verbatim word spotting using deep neural networks," in *Proceedings of the* 15^{th} *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 307–312.