Compact Deep Descriptors for Keyword Spotting

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

[1] *Computational Intelligence Laboratory, Institute of Informatics and Telecommunications*
National Center for Scientific Research "Demokritos", GR-15310 Athens, Greece
Email: {georgeretsi,sfikas,louloud,nstam,bgat}@iit.demokritos.gr

[2] *School of Electrical and Computer Engineering, National Technical University of Athens, GR-15773 Athens, Greece*

***Abstract*—In this work, we present a novel approach for the extraction of deep features from a Convolutional Neural Network (CNN), designed for the task of Keyword Spotting (KWS). The main novelty of our work concerns the generation of a compact descriptor able to simulate the existence/absence of unigrams or bigrams. This is accomplished using a binary, attribute-based representation of a word string together with an appropriate training procedure. Deep features are extracted from the output of the last convolutional layer and are organized into zones in order to incorporate spatial information of the detected attributes. In addition, a novel optimization scheme is proposed which relies on a very effective initialization of the network generating the compact descriptors. Experiments conducted on the IAM dataset prove the efficiency of the novel compact descriptor since the proposed system's performance in on par with the state-of-the-art.**

I. Introduction and related work

Word or keyword spotting (KWS) is defined as the problem of searching for a specific textual query in a corpus of digitized documents [1]. The query can be chosen by the user either in the form of a cropped image containing an instance of the word to be searched, or in the form of a text string. These two spotting scenarios are usually referred to as query-by-example (QbS) and query-by-string (QbS) keyword spotting. QbE keyword spotting can hence be viewed as a special form of content-based image retrieval (CBIR), where the query is an image containing a digitized word instance ("word image"). While most KWS systems implement either one of the QbE or QbS scenarios, recently there have been proposed systems implementing both QbE and QbS under the same underlying model. This is the case for example in [2], where available training word images and text strings are used to learn a projection onto a common, latent subspace characterizing both word image and word string instances. The learned projection parameters are used to project the user query and the database word instances on the latent subspace. As projections are defined for either images or text strings, QbE and QbS are performed simply as nearest neighbor searches on the latent subspace. The same model can also be used for word recognition under a similar consideration.

Another taxonomy of KWS systems is based on the exact type of the elements that are to be retrieved, as well as the level of image segmentation that is assumed before the application of the KWS system *per se*. Hence, depending on whether a word-level or line-level segmentation is assumed, different KWS systems have been proposed (e.g. [3], [4]). Segmentation-free systems are also possible, where document pages are provided with no assumed segmentation on any level, hence casting KWS as a special form of object detection [5].

Supervised learning-based methods are a definite trend in the KWS state-of-the-art. In particular, convolutional networks [6] and recurrent neural networks [7] are prominent neural network architectures, used successfully for the KWS task.

In this paper, we propose a KWS system that is based on a new deep, compact word descriptor, that is computed using a set of appropriately trained neural networks. The proposed descriptor is also compact, in the sense that word instances can be represented as low-dimensional vectors following our technique; this result can be very advantageous in a large-scale KWS setting, where large document collections need to be efficiently indexed and subsequently searched.

Existing approaches either assume lexicon classes (each unique word corresponds to a different class) or an embedding of the spatial distribution of the existing characters (e.g. [6], [8]). The most popular string embedding is the Pyramidal Histogram of Characters (PHOC), which encodes the spatial information of the characters by using a pyramidal strategy [2]. The recent PHOCNet model [6] uses this attribute-based PHOC representation as a target to a CNN, which transforms images to a string embedding space and therefore both QbE and QbS scenarios are enabled.

In contrast to the aforementioned approaches, for which each word is represented by a discriminative target class, we assume that our target contains only the existence or not of unigrams and bigrams, resulting to a simple attribute-based representation. This approach cannot discriminate different words at target level, since many words contain the same characters, e.g. "dog" and "god". To address this, the spatial information of each character should be

978-1-5386-5875-8/18/$31.00 ©2018 IEEE
DOI 10.1109/ICFHR-2018.2018.00062

added to the generated descriptor. The word descriptor is therefore extracted from the convolutional output of the final convolutional layer, which can be viewed as an intuitive discriminative feature map (note that conceptually related maps, though following quite different considerations, were recently presented in [9] extending convnet classifiers and in [10] extending Generative Adversarial Nets (GANs)). Specifically, the word descriptor is the concatenation of the max-pooled outputs after applying a zoning scheme on the convolutional output, i.e. splitting of the output into horizontal segments.

Following the aforementioned feature extraction procedure, the generated descriptor is dependent to the number of filters of the last convolutional layer, which is typically large; this shortcoming is addressed by the use of a compact descriptor. To the end of computing the proposed compact descriptor, we present a method that entails a novel, two-stage optimization scheme. According to the proposed optimization scheme, two separate neural network architectures are defined. This couple of neural networks, dubbed here the *extended* and *compact* network are trained sequentially, in the sense of using the former network's output as a pre-training initialization for the latter. We demonstrate that this scheme leads to superior results compared to obtaining deep features separately from either of the two networks.

Another contribution of this work, is a novel training strategy, which relies on the application of the fully connected layers on parts of the convolutional output and on the combination of different responses into a single output (the attribute-based target). The idea behind this approach is to train a model that can accurately detect unigrams (or bigrams) independently of the context, i.e. the word image, as it will explained in detail at the corresponding section.

The rest of the paper is organized as follows: In section II, the processing pipeline of the proposed method is outlined, focusing mainly on the description of the model's components as well as on the analysis of the proposed optimization scheme. In section III we evaluate the proposed algorithm with keyword spotting trials on the IAM dataset. We conclude the paper in section IV, where we discuss the paper contribution together with directions for future research.

II. Proposed networks and optimization scheme

The proposed keyword spotting method assumes a set of annotated, word-level segmented images to be available for training. The case of interest is the QbE paradigm; the query, as well as the elements to be retrieved are segmented word images. Concerning the method itself, the backbone of the proposed processing pipeline is a pair of convolutional neural networks. These two networks are distinct, yet they are architecturally similar to a great extent. We shall dub the two networks, *extended network* and *compact network*. The architecture of both networks follows the template set by the PHOCNet model [6], [11], where a pyramidal pooling approach was employed for the flattening layer. However, the application of the network layers is different from the typical approach of [11]. One of the main contributions of this work is to apply the fully connected layers on multiple max-pooled outputs from the convolutional part, rather than creating a unique, fixed-size, feature vector from the flattening operation and fed it to the fully connected part. This different approach is adopted in order to simulate the existence or absence of characters, which is the target of our network. Contrary to the proposed approach, the concatenation of pyramidal pooling outputs, a procedure which is followed by [6], [11], contains spatial information being propagated to the fully connected part.

The final word image descriptor is evaluated as a function of convolutional layer activations, following the paradigm of deep features [12]. In this work, we use a word label that relies on encoding merely the existence of unigrams/bigrams, without directly taking into account their relative positions on the word. This is in contrast to what is done by recent word embeddings ([2], [13], [8]). Given the aforementioned targets, we expect that the final convolutional layer will generate features closely related to the existence of the respective unigrams or bigrams. Concerning the testing procedure, we integrate the description with a spatial reference of attributes indirectly, by applying a zoning scheme. The final, fixed-sized feature vector is generated by a max-pooling operation over the segmented feature map into zones. An overview of the base neural network architecture can be examined in Figure 1.

A. Word labels

According to the PHOCNet architecture, the network should learn the characters' representations (i.e. if a specific character exists in the image) along with their relative position (e.g. the character is in the first half of the image). This is imposed by the PHOC embedding, which encodes characters and their relative position. The PHOC formulation of relative positions requires filter responses that distinguish characters as well as their position and therefore a great number of such (convolutional) filters are required.

One straightforward simplification is to train the network in order to decide upon the existence or not of the possible characters. The word label for this approach would be a binary embedding (denoting the existence or not) of possible unigrams (lowercase letters and digits: 36 unigrams in total). An extension to the aforementioned embedding is to also include all possible bigrams to the word target (36 unigrams + 36×36 bigrams). Hence, the length of the last, sigmoid-activated layer is $36 \times 37 = 1337$ neurons long. This representation is expected to be sparse, since several bigrams do not exist for the English language (or the selected script language). The reasoning of such an

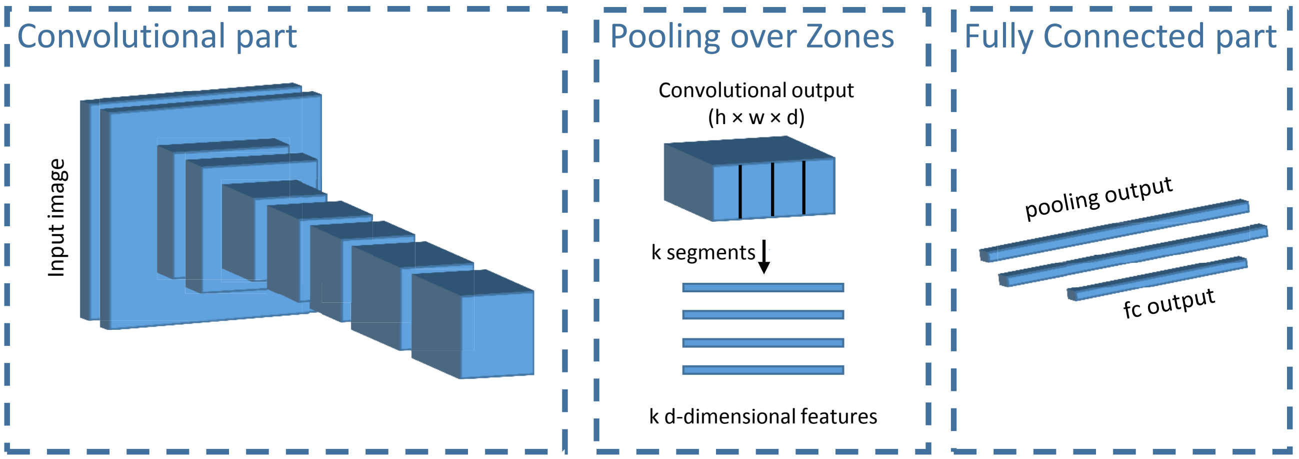

Figure 1. Neural network architecture overview. The base architecture, used for both employed neural networks in this work ("compact" and "extended") follows the template of a convolutional neural network comprising three components: a convolutional backbone (left), a pooling layer (center) and a fully-connected head (right). The pooling layer, applied on the last convolutional layer, is used as a deep feature extractor.

alternative representation is that usually isolated characters differ visually when written alongside specific characters.

Both representations are useful for training filters that correspond to specific characters (or couples of characters). However, they cannot constitute a descriptor that could be used for effective word comparison, which would in turn be necessary for a KWS system. This holds especially when taking into account a description comprising only unigram attributes with no spatial reference. To this end, we use the response of the final convolutional layer as a deep feature. This will ideally encode non-spatial, character attribute information. Concerning dealing with encoding spatial information, we integrate the obtained feature with a zoning scheme [14].

B. Network Architecture

We adopt a network consisting of three main components, namely:

- **Convolutional Part:** 2 convolutional layers $\rightarrow$ 2×2 max-pooling layer $\rightarrow$ 2 convolutional layers $\rightarrow$ 2×2 max-pooling layer $\rightarrow$ 9 convolutional layers.
- **Pooling/Flattening Part:** Use of a Temporal Pyramidal Pooling (TPP) [11] scheme on the convolutional output, without concatenating the resulting max-pooled outputs. This is explained in detail in the following subsections.
- **Fully Connected Part:** 3 fully connected layers resulting to a vector whose size is equal to the word's representation attributes (36 or 1337).

The size of all convolutional filters is set to 3×3. All non-pooling layers are topped by ReLU non-linearities, except for the output layer. The network is topped by sigmoid activation functions, with each layer variate representing a word image attribute class.

Each level l of the previous convolutional layer output is segmented into l horizontal zones. As a result, for n_l levels and a convolutional output of depth d, we get a set of $N = \sum_{l=1,\dots,n_l} l = n_l(n_l+1)/2$ different d-sized vectors. Compared to the standard Temporal Pyramidal Pooling (TPP) layer, an important tweak at this point is that we do not concatenate the N vectors into a single vector. Instead, each one of the N max-pooled, d-sized outputs is fed to the fully connected stack of layers separately.

An alternative way to understand the proposed approach is to view the fully connected stack in our model as a stack of convolutionalized layers. Each fully connected layer can then be seen as a deep 1×1 trivial convolutional filter, where the (single) filter weight for each depth value would correspond to a fully connected synapse weight. In the proposed architecture, this paradigm is employed with the additional tweak that the (trivial) convolutional filter weights are shared among each one the N vectors that are produced as the output of temporal pooling.

The N vectors that are produced as the output of the modified TPP layer are fed to the fully connected layer and N output vectors are generated. On the output layer, the variates of each of these N sigmoid-activated vectors correspond to the word attributes previously described in subsection II-A. Each one of the separate N vectors can hence be interpreted as an attribute detector and need to be combined into a single output. The outputs of each zone segment, after the fully connected part, corresponding to a specific pyramid level should be combined in order to produce a valid attribute representation, since each zone contains a different set of unigrams/bigrams. Therefore, they can be combined using a union operation of the separate attribute representations, i.e. an element-wise max operation. Each pyramid level, after the aforementioned max-pooling, should provide a valid attribute representation of the word and under this assumption we employ an element-wise min operation over the different attribute representations, which relates to optimizing the precision of the representations (in other words, the min operation assists the correction of true negatives over the pyramid levels). Figure 2 visually depicts the training strategy, consisting of union/intersection operations, that we described above.

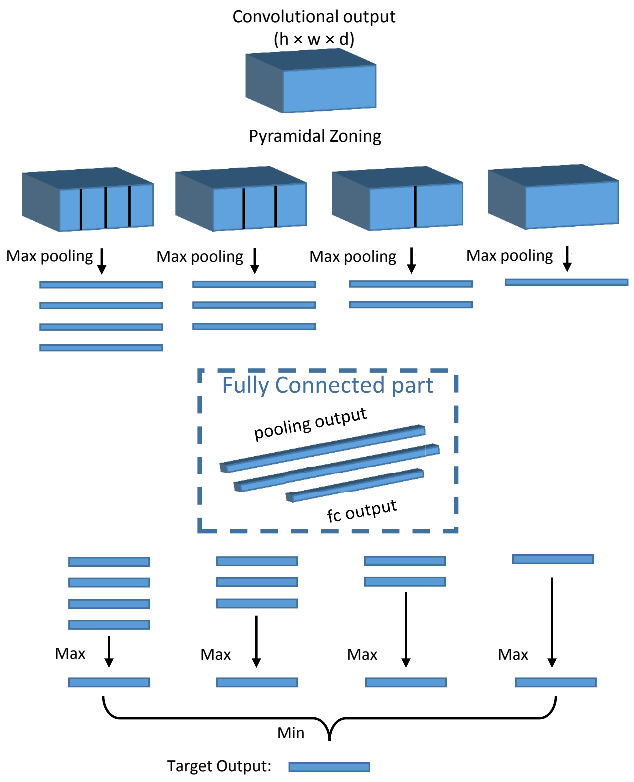

Figure 2. Proposed training strategy, which uses a modified TPP layer and generates multiple outputs for each image. Convolutional outputs are first pooled to N feature vectors (top, in this figure $N = 4+3+2+1 = 10$) using temporal max pooling. These are transformed to N attribute detector outputs (bottom), which are then combined with max and min operations to form a single attribute detector output.

C. Extended and compact network

The two networks, extended and compact, differ in the layout of exactly one layer, which however makes for a considerable difference in practice. The layer that differs is the last convolutional layer, i.e. the layer that is used as input to the pyramidal pooling layer. As mentioned before, the word image representation is linear to the depth of the convolutional output which means that reducing the number of filters of the last convolutional layer results to compact image representations. Nevertheless, such reduction may significantly affect the network performance.

Our goal is to examine if we can accomplish acceptable performance on the QbE KWS scenario even if we significantly reduce the depth of the convolutional output, i.e. the number of filters. A compact representation is supported by the idea of learning filters that only detect characters and not their spatial information, since much less filter depth is required for this goal. It should be noted that the compact architecture not only provides compact feature representations, but also results to a significant reduction of the inference time.

D. Optimization scheme

An important observation is that the training of the compact model, assuming random initialization, converges to suboptimal solutions. This is not surprising, since the reduced last convolutional layer operates as bottleneck to the entire architecture and it is highly probable to have a unique solution with respect to its parameters. Such a unique solution is difficult to approximate while using a stochastic gradient approach to simultaneously optimize the whole set of parameters. Nevertheless it is safe to assume that the extended and the compact network are closely related regarding their weights, since the task remains the same. Compared to the compact network, the extended network can be trained effectively without a problem from randomly initialized weights.

Therefore, for training, the networks are used in the following manner. We use the available word images to first train the extended network. After the training of the extended network finishes, we keep all of the convolutional weights except for those of the last layer, and re-use them as pretraining weights for the compact network. Note that this is possible since the two networks share very similar weights with regard to all convolutional layers, save for the last one.

In detail, we apply a two-step training procedure, which ensures a fast convergence of the compact network:

1) **Initialization:** Use the pre-trained model of the extended network to initialize the convolutional layers except the last one, since it has a different number of filters. Train only the remaining layers/parameters, i.e. the last convolutional layer and the fully connected layers, assuming the other convolutional layers fixed. This approach will converge fast to a valid solution, which corresponds to a decent performing model.
2) **Fine-tuning:** Re-train the whole network, using the initialization of the first step. All convolutional layers are also trained in this step in order to better adapt to the problem.

The intuition behind the reason the proposed scheme works is summarized as follows. The extended network is easier to optimize, in the sense of reaching lower loss and higher accuracy figures comparatively fast. At the same time, the compact network is harder to optimize. Since the architectural difference between compact and extended network is the depth of the last convolutional layer, we can conclude that this feature forms an apparent optimization bottleneck. However, having two similar models comes with the advantage that the weight search space of the two networks is "similar". This is said in the sense that the compact and extended network differ only to a minor percent of weights, and practically all their convolutional backbone is identical. In practice, we have validated that the compact network trains better once a good initialization is used; this is provided by the (comparatively) easy-to-train extended network.

III. Experimental results

We have run keyword spotting trials on the well-known IAM dataset [15]. The IAM dataset contains a total of $115,320$ words written by 657 different writers. The large number of comprised words, as well as their diversity in writing style, make it ideal for training and testing deep neural networks. 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, as done in [2].

Given the extracted descriptors corresponding to the images consisting the testing set, the retrieval list is computed by nearest neighbor search using the cosine distance [11]. 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.

Concerning our method, we assumed 5 levels of the modified TPP layer and 5 zones on the extraction of the deep descriptor. In Table I, the numerical results for both the extended and the compact network are presented. There are three main observations: 1) The attributed-based output used in this work is not capable of successfully discriminating words, as expected. On the contrary, the zoning scheme achieves very good results. 2) Including the bigram information on the target vector provides a notable boost in performance. 3) Compact network retains a good performance, even though it generates $32\times$ smaller descriptors. For the rest of the evaluation section, we consider unigrams+bigrams representation and the zoning scheme as the default parameters for our method.

It should be noted that if we train the compact network with randomly initialized weights, the model seems to get stuck to a local optima which performs at least 5% lower compared to using the proposed optimization scheme.

Table II reports the numerical results for the proposed method compared to several state-of-the-art methods. Both networks, extended and compact, achieve notable results that are on par with state-of-the-art results (extended network outperforms the reported methods). Concerning the compact descriptor, which is our case of interest, not only it achieves state-of-the-art results but also it has other merits as well.

Table I
MAP (%) performance comparison of the extended and the compact network.

Approaches	Unigrams	Unigrams+Bigrams
Extended		
Zoning ($5 \times 512\,d$)	80.79	84.68
Output	60.43	73.77
Compact		
Zoning ($5 \times 16\,d$)	74.72	81.65
Output	52.06	56.95

First, even though the proposed optimization scheme is a two-step process, integrating the training of two separate networks, word descriptor evaluation can be performed by a simple feed-forward on the compact network only. Hence, descriptor evaluation is comparably fast. Second, the resulting word image descriptor is compact, represented by a 80-dimension vector, which is almost 10 times smaller than the PHOC representation of [6]. This can be seen as a considerable advantage, especially in a large-scale spotting context.

Table II
MAP comparison on IAM Dataset using the extended and the compact networks trained on the unigrams+bigrams attribute representation.

Method	MAP(%)
PHOCNet [6]	72.51
Attribute SVM [2]	55.73
Krishnan et al. [16]	84.24
Wilkinson et al. [8]	81.58
Deep PHOCNet features [12]	81.50
PHOCNet-TPP [11]	83.38
Extended Network	84.68
Compact Network	81.65

Finally, the network formulation and training scheme that we presented enables the detection of characters on the image, which is not explored on this work in the context of KWS. However, it is an advantage of our approach and could provide a cost-effective solution for segmentation-free OCR on document images. A preliminary visualization of the aforementioned idea is depicted on Figure 3, where the fully connected layers are used as convolutional layers with 1×1 filters and the initial images is transformed to a response map of the existing unigrams.

IV. Conclusion and Future work

In this paper, we have presented a new method for keyword spotting. Our method is based on producing a word image descriptor, extracted as a deep feature vector. The required feature vector is produced as a function of the layer activations of the employed convolutional network architecture. One key element of our approach is to train our model on a representation that encodes the existence or not of unigrams/ bigrams and support it with an appropriate

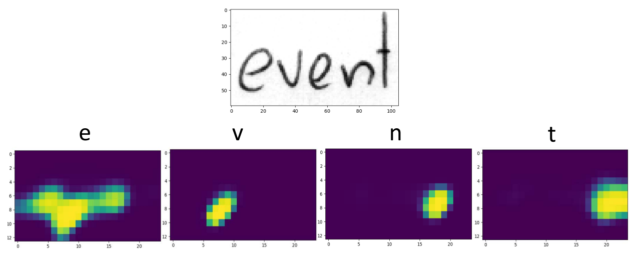

Figure 3. Response output maps of the word image “event”, after the use of the fully connected part as convolutional with 1×1 filters. The only characters that have high confidence (> 0.8) on the output are the existing ones, “e”, “v”, “n” and “t”. Only the responses of these unigram filters are depicted and we can clearly observe the spatial information about each unigram.

training scheme that learns features independent of the characters’ positions.

Our goal is to extract compact deep features vectors, avoiding the high-dimensional representations that are imposed from the existing architecture. In order to train a model that generates compact descriptors efficiently, we have proposed an optimization scheme that is based on the interplay of two related networks, dubbed here extended and compact network. The extended network trains more easily, but provides a non-compact word image descriptor. On the other hand, the compact network is harder to train, but provides a compact descriptor. The end-result of our method is a compact descriptor, produced as the output of training the two networks together. Numerical experiments on the IAM dataset have validated the usefulness of our method, with results that are on-par with the KWS state of the art.

One interesting research direction and extension of the current approach is the application of the trained network as character detector, an idea that was supported by preliminary experiments as shown at the experimental section. Such an approach, along with a decoding process, could provide a cost-effective solution for handwritten text recognition.

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*, 2017.

[2] 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.

[3] G. Retsinas, G. Louloudis, N. Stamatopoulos, and B. Gatos, “Keyword spotting in handwritten documents using projections of oriented gradients,” in *International Workshop on Document Analysis Systems (DAS)*. IAPR, 2016, pp. 411–416.

[4] A. H. Toselli, E. Vidal, V. Romero, and V. Frinken, “Hmm word graph based keyword spotting in handwritten document images,” *Information Sciences*, vol. 370, pp. 497–518, 2016.

[5] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, “Segmentation-free word spotting with exemplar svms,” *Pattern Recognition*, vol. 47, no. 12, pp. 3967–3978, 2014.

[6] S. Sudholt and G. A. Fink, “PHOCNet: A deep convolutional neural network for word spotting in handwritten documents,” *arXiv preprint arXiv:1604.00187*, 2016.

[7] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, “A novel word spotting method based on recurrent neural networks,” *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 34, no. 2, pp. 211–224, 2012.

[8] T. Wilkinson and A. Brun, “Semantic and verbatim word spotting using deep neural networks,” in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 307–312.

[9] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization,” in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2921–2929.

[10] D. Kastaniotis, I. Ntinou, D. Tsourounis, G. Economou, and S. Fotopoulos, “Attention-aware generative adversarial networks (ATA-GANs),” *arXiv preprint arXiv:1802.09070*, 2018.

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

[12] G. Retsinas, G. Sfikas, and B. Gatos, “Transferable deep features for keyword spotting,” in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 2, no. 2, 2018, p. 89.

[13] J. A. Rodriguez-Serrano, F. Perronnin, and F. Meylan, “Label embedding for text recognition,” in *British Machine Vision Conference (BMVC)*, 2013.

[14] G. Sfikas, G. Retsinas, and B. Gatos, “Zoning aggregated hypercolumns for keyword spotting,” in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 283–288.

[15] U. Marti and H. Bunke, “A full english sentence database for off-line handwriting recognition,” in *International Conference on Document Analysis and Recognition (ICDAR)*, 1999, pp. 705–708.

[16] 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 (ICFHR)*, 2016, pp. 289–294.