Semantic and Verbatim Word Spotting using Deep Neural Networks

Tomas Wilkinson and Anders Brun
Department of Information Technology
Uppsala University
Uppsala, Sweden
{*tomas.wilkinson, anders.brun*}*@it.uu.se*

***Abstract*—In the last few years, deep convolutional neural networks have become ubiquitous in computer vision, achieving state-of-the-art results on problems like object detection, semantic segmentation, and image captioning. However, they have not yet been widely investigated in the document analysis community. In this paper, we present a word spotting system based on convolutional neural networks. We train a network to extract a powerful image representation, which we then embed into a word embedding space. This allows us to perform word spotting using both query-by-string and query-by-example in a variety of word embedding spaces, both learned and handcrafted, for verbatim as well as semantic word spotting. Our novel approach is versatile and the evaluation shows that it outperforms the previous state-of-the-art for word spotting on standard datasets.**

***Keywords*-handwritten word spotting; convolutional neural networks; deep learning; word embeddings**

I. Introduction

Word spotting is the task of searching through a scanned document collection to find a given search query. One way of doing this is to use the query-by-example (QbE) paradigm, wherein an example of the search query is manually cropped from an image and used as a template to find more instances of the same word. This approach is solely based on comparing the visual appearances of the words. Another way is to search using a string of characters as a query, usually called query-by-string (QbS). QbS has an advantage that it does not require the user of a word spotting system to find an instance of a query word that prior to searching. However, QbS word spotting systems require a mapping from strings to images, which is typically learned using supervised learning, thus requiring annotated data. Historically, QbE has been the paradigm of choice for word spotting, because such systems are typically simpler to implement.

More recently, QbS word spotting methods have been getting very good performance [1], [2], [3]. The approaches are based on learning a mapping from the word images to a word embedding space, in which word image labels are embedded. In this work, we adopt the same paradigm and investigate the performance of four different embeddings, the Pyramidal Histogram of Characters (PHOC) [1] representation, a novel coding of word that we call DCToW, Discrete Cosine Transform of Words, an embedding that applies a discrete cosine transform to a one-hot string encoding. Finally, we evaluate two learned word representations from the model in [4]. Our approach can be seen as a combination between [1] and [2], since like [1], we can learn an embedding to any space, and like [2], we use neural networks to learn the image-to-word mapping.

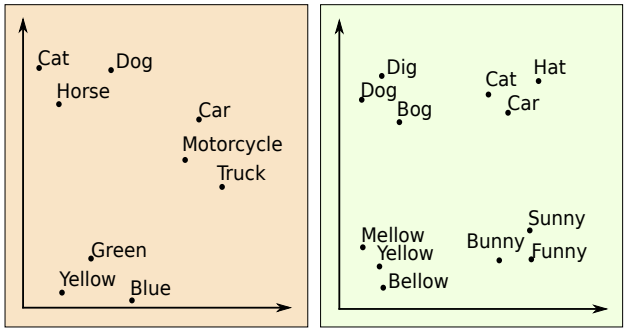

Figure 1. The two kinds of word embedding spaces we perform word spotting in. To left is an example of a semantic space, where words are grouped by semantic categories rather than string similarity. To the right is the typical space that we use for verbatim word spotting, where words are grouped on string similarity.

A word embedding space typically places similar words close to each other, given some similarity metric. For a verbatim word spotting, a suitable distance metric to approximate could be the edit distance between word strings. However, one of the word embedding spaces we investigate in this paper is a semantic embedding space, wherein the distances between word vectors correspond to semantic difference of words. We call this *semantic word spotting*. In this space, the goal is not only to focus on the first verbatim results that are retrieved, but also on the secondary retrievals. The use case we imagine is a user exploring a largely unknown document collection, not already knowing exactly what is interesting and which terms to search for. For example, a search for the query ”sunday” in this space would in the ideal case first return all instances of ”sunday” followed by perhaps other days in the week. For an illustration of the difference between a verbatim space and a semantic space, see Figure 1.

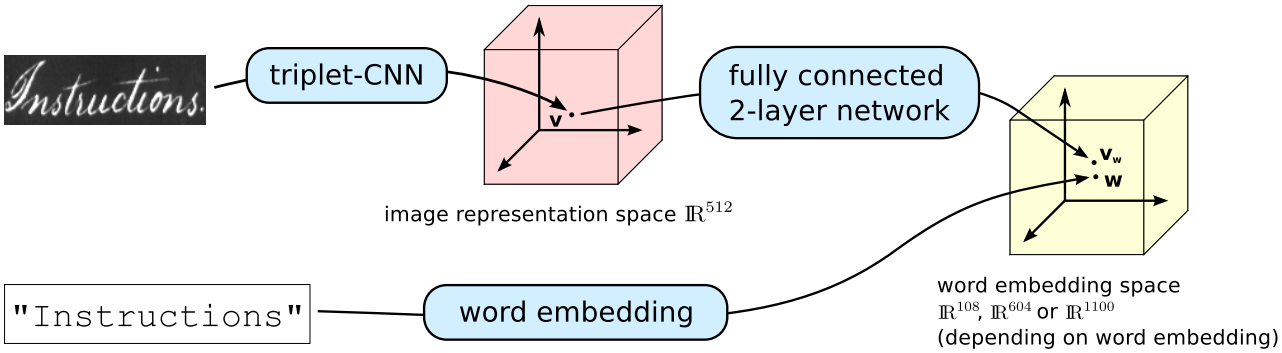

Figure 2. An overview of our system. An input image is fed through the triplet-CNN to get an image representation, $\mathbf{v}$. $\mathbf{v}$ is then fed through the fully-connected 2-layer embedding network to produce $\mathbf{v_w}$, the image embedded in the word embedding space. Similarly, we either feed the corresponding transcription through the neural network language model or calculate its PHOC or DCToW representation to produce the word embedding $\mathbf{w}$.

II. Related Work

Deep convolutional neural networks (CNN) have been popular in recent years, starting with the leap in performance compared to hand-crafted approaches in the 2012 large scale visual recognition challenge. In the following years, deep CNNs have become the foundation for many state-of-the-art systems in computer vision for tasks like object detection [5], semantic segmentation [6], and image captioning [7].

Recently, there has been a lot of work surrounding the connection between image representations and word embeddings, both with natural images [8], and word images [1], [2], [9], [10]. For the handwritten word spotting community, hand-crafted image representations, primarily Fisher vectors and other bag-of-visual-words models, have been the features of choice [1], [3], [10]. However, there have been some notable exceptions [2], [11]. A hand-crafted approach has also been used for the word embeddings, where the embedding of choice has been the PHOC representation and its variations.

In the last few years, word embeddings have become very popular in the natural language processing community, with one of the most widely known being word2vec [12]. The learned embeddings typically capture the semantic similarity between words, although many approaches do not explicitly take character information into account, seeing words as atomic rather than a composition of characters.

There has been some recent work on embedding word images into semantic spaces [13]. However, they use a different approach whereby they directly learn a mapping from word images to semantic categories, mined from Wordnet. For example, the semantic category for the word "astronomer" could be scientist. This is distinctly different from how the semantics of word embeddings in natural language processing are learned since they are based on the similar usage of words in text, and therefore different from our approach.

III. System

The word spotting system consists of three different modules that are trained in three stages. First, to learn a representation of the word images, we use a triplet network architecture [14]. Second, the system in [4] is adopted to learn a word representation. In addition, the hand-crafted PHOC and DCToW representations are also evaluated. Finally third, a small fully connected network is used to learn a mapping from the image representation to the word representation. The entire system architecture can be seen in Figure 2.

A. Image Representation

The network used to learn an image representation is adopted from [14]. They use the network to learn an image descriptor for an image patch by comparing whether or not patches are from the same class. We adopt the system to learn a descriptor for an image of a word. The architecture is a so-called triplet network, which takes as input a triplet of images and is trained to measure the distance between them.

The input is a triplet of word images, $\mathcal{T} = (p_1, p_2, n)$, consisting of two of the same class (positive) and one of a different class (negative). The network consists of a convolutional neural network that is replicated 3 times, one for each input in the triplet, while sharing the weights. For the CNN architecture we use the 34-layer ResNet described in [5], due to its recent state-of-the-art performance on visual tasks. Each image is propagated through their respective CNN, resulting in a descriptor for each image, and then the distance between each pair in the triplet is calculated. Using the specially designed SoftPN loss function [14] that selectively back propagates the similarity that is smallest between the two positive images and the negative image, as well as the distance between the positive pair.

$$L(\mathcal{T}) = \left(\frac{e^{d(p_1,p_2)}}{e^{min(d(p_1,n),d(p_2,n)} + e^{d(p_1,p_2)}}\right)^2 + \left(\frac{e^{min(d(p_1,n),d(p_2,n)}}{e^{min(d(p_1,n),d(p_2,n)} + e^{d(p_1,p_2)}} - 1\right)^2 \quad (1)$$

After having learned this model on triplets of images, we use one of the three identical CNNs to extract 512-dimensional feature vector, $\mathbf{v}$. We will refer to this network as the triplet-CNN from now on.

B. Word Representations

We investigate four different word representations in this paper. The first is the PHOC [1]. This is a hand-crafted, pyramidal representation that represents a word as a high dimensional binary histogram of the occurrences of characters in multiple splits of the word. For example, at level 2, a word is split into two, at level 3, it is split into 3 parts etc. The final histogram representation is concatenated for each level into a final histogram. This can be extended to looking for common n-grams present in different splits of the word. We use the PHOC settings from [1] and [2] which uses the pyramid levels [2, 3, 4, 5] and the 50 most common English bi-grams in level 2. The resulting word embedding, $\mathbf{w}_p$ has 604 dimensions.

In addition to PHOC, we formulate and investigate another hand-crafted embedding. We define a novel coding of word that we call DCToW, Discrete Cosine Transform of Words. Given a character string s with length $|s| = m$, we convert each character to a vector using one-hot (one-of-k) encoding. The result stored as a $K \times m$ matrix, where K is the size of the alphabet. Since this matrix has a variable width, depending on the length of the word, we first apply a discrete cosine transform (DCT-II) to each row. We then crop the matrix, selecting the R first DCT components. Or in case $R > m$, we instead pad with zeros to guarantee that a new $K \times R$ matrix is formed. This matrix is then flattened, resulting in $K \cdot R$ features that will represent the word s. For the experiments in this paper, we set $R = 3$, and $K = 36$, which gives us the embedding $\mathbf{w}_w \in \mathbb{R}^{108}$.

In this paper, we also investigate the use the LSTM-Char-Large model from [4] to learn a language model from separate text data. We can in fact extract two conceptually different representations from this trained model. The first of the learned representations is the output of a small neural network that performs convolutions with a set of filters over sequences of character embeddings followed by a max-pooling over time operation to achieve a fixed length representation. One can think of this representation as character n-grams. We will henceforth call this the n-gram embedding, $\mathbf{w}_n$.

The second representation is the first space propagated through two so-called highway layers [15]. The highway layer models interactions between the character n-grams that transforms the space it to a semantic space wherein the distance between two words measures how semantically similar they are in addition to string edit similarity, see Figure 1. We will call this the semantic embedding, $\mathbf{w}_s$, from now onwards. Both $\mathbf{w}_s$ and $\mathbf{w_n}$ are in $\mathbb{R}^{1100}$.

C. Learning the Embedding

Previous approaches use techniques like Canonical Correlation Analysis [1], Latent Semantic Analysis [9], or structured Support Vector Machines [9] to learn a mapping from image representation to a word embedding space. In order to allow for fine-tuning of the triplet-CNN, we opt for a small fully connected neural network, consisting of two hidden layers of size 4096. Input is a fully connected layer of size 512, the dimensionality of the image representation $\mathbf{v}$. The output is a fully connected layer with size equal to the dimensionality of the embedding. After the output layer, we perform l^2 normalization on the outputs. We use the hyperbolic tangent nonlinearity with batch normalization [16] before the activation. Similar to the triplet-CNN, this network is trained using randomly generated pairs of images (fed through the triplet-CNN) and embeddings $(\mathbf{v}, \mathbf{w})$. The cosine embedding loss is used to train the network, and it is defined as follows. Given $\mathbf{x} = (\mathbf{v}, \mathbf{w})$ are two vectors and a label $y \in \{-1, 1\}$ indicating whether or not $\mathbf{v}$ and $\mathbf{w}$ are part of the same class. We get

$$L(x, y) = \begin{cases} 1 - \cos(\mathbf{v}, \mathbf{w}) & \text{if } y = -1 \\ \max(0, \cos(\mathbf{v}, \mathbf{w}) - \gamma) & \text{if } y = 1 \end{cases} \quad (2)$$

where γ is a margin. The loss function requires that the pairs are sampled such that $\mathbf{v}, \mathbf{w}$ are from the same class (y=1) and from different classes (y=-1), and various ratios may be chosen. However, for simplicity, we sample half of the pairs from the same class. Let us call the output from the embedding network $\mathbf{v_w}$, which is the image representation $\mathbf{v}$ embedded in the word embedding space.

IV. Experiments

We evaluate our method on two datasets, the George Washington dataset (GW) [17] and the IAM Handwritten Database (IAM) [18]. The GW dataset consists of 20 pages from the George Washington papers at the Library of Congress. It was written by George Washington and his secretary. To keep our results comparable, we adopt the same evaluation procedure as [1], [2], and use the same four folds of the dataset to evaluate.

The second dataset is the IAM offline handwriting dataset. It consists of 1539 pages, 115320 words, of modern handwritten text written by 657 different writers. We use the official partition for the writer independent text line recognition. Like [1], [2], we exclude the official stop words as queries.

Similar to [2], we utilize data augmentation to a large degree while learning. We use the same strategy for the GW

database of first generating samples such that we get the same amount of examples in each class, then we continue generating until we get a total of 500000 examples in total. For the IAM database, we limit the amount of samples we generate per class to 50. Our approach, which we found to work slightly better, differs in how we generate our samples. Instead of sampling two sets of points and estimating an affine transform, we uniformly sample a small angle (in degrees) for rotation and shearing transforms from the intervals [-5, 30] and [-5, 5]. Then we embed the resulting image with vertical and horizontal padding, uniformly sampled from $\{3, 4, ..., 15\}$. Finally, we apply either a gray scale erosion or dilation with a square kernel, the size of which is randomly sampled from $\{1, 2, 3\}$ and $\{1, 2, 3, 4\}$ respectively, to thicken or thin the ink. Before feeding the word images to the network, we resize them to a size of 60×160 and invert them.

A. Training Procedure

First we pre-train the triplet-CNN on the CVL database [19] for 185000 iterations (parameter updates). We then use that model to initialize the models for the GW and IAM databases, thereby performing fine-tuning on them. We found that this helped to reduce overfitting.

We train the GW and IAM databases nearly identically, with the differences stated below. We make sure to generate enough randomly sampled triplets to ensure that the same triplet is never seen by the networks twice. The learning rate is initially set to 0.01, and is multiplied every 30000 (50000 for IAM) iterations by 0.1 and we train for 100000 iterations for the GW database, and 200000 iterations for IAM.

We use the default settings, described in [4], to train the language model. For the GW database, we train the model on all the 40 volumes of "The Writing of George Washington from the Original Manuscript Sources 1745 - 1799" [1] using a vocabulary size of 20000 of the most common words. We normalize the data by approximately separating the text into sentences using regular expressions and replace words not in the vocabulary and numerals with a special token. For the IAM dataset, we train a language model trained on the Penn Treebank using the default settings.

For the embedding network we train for 30000 iterations for GW database and 50000 for the IAM database. Similar to the triplet-CNN, we generate enough image-embedding pairs such that network never sees the same triplet twice. In addition to training the embedding network, we also fine-tune the CNN, which we found to greatly improve results. We set the margin in the loss function γ, to 0.1 for GW and 0.3 for IAM.

We train the triplet-CNN and embedding network using mini-batch stochastic gradient decent with Nesterov Momentum. We use a batch size of 128, a weight decay of 10^{-4}, a momentum of 0.9, and a learning rate of 0.01.

[1] https://archive.org/details/writingsofgeorge01wash

Table I
MAP COMPARISON (IN %) WITH STATE-OF-THE-ART METHODS

Methods	GW		IAM	
	QbE	QbS	QbE	QbS
triplet-CNN	93.61	-	70.81	-
PHOC	**98.00**	92.23	77.24	**89.49**
DCToW	97.98	**93.69**	76.98	85.33
n-gram	97.70	83.97	53.43	45.08
semantic	96.91	69.81	**81.58**	75.74
embedded attributes [1]	93.04	91.29	55.73	73.72
PHOCNet [2]	96.71	92.64	72.51	82.97
Finetuned CNN [11]	-	-	46.53	-
LSA [20]	-	56.54	-	-

B. Evaluation Procedure

To test a new query we need to embed it into the embedding space. For QbE, we first resize the image to 60×160 and invert it. Then we feed it through the triplet-CNN and then embed it in the word embedding space by feeding the output feature vector from the triplet-CNN through the embedding network. For QbS, we embed the query string in the embedding space by either feeding it through the trained language model or calculating its PHOC or DCToW representation. We then measure the distance to each instance in the database using the l^2-distance and return a list of retrieved results sorted by distance to the query.

We evaluate our different setups using both QbE and QbS and to be comparable, we use the exact same procedure as [1], [2]. For QbE, we use all the word images in the test set that occur more than once as queries. In addition, the query is removed from the retrieved list and not counted. For QbS, we limit the test set to only contain one of each unique transcription. Following the standard protocol, we remove stopwords from the list of queries for the IAM dataset. We report our results using the mean average precision (MAP) score. We compare the different embeddings to the state-of-the-art in segmentation-based word spotting. In addition, we report the result of the triplet-CNN for QbE word spotting.

C. Results

The performance of our different setups (top), compared to the state-of-the-art approaches (bottom), can be seen in Table I. First we note that our system achieves a very competitive performance compared to the state-of-the-art method, PHOCNet [2]. In fact, our various setups outperform PHOCNet in all four different evaluations.

Figure 3 shows the top ten *unique* (i.e., we keep only one instance for each different class that is retrieved) results for three queries using the semantic, n-gram and PHOC embeddings. For the query "captain", we observe that not only are "captain" and "captains" returned, but also other military ranks like "colonel", "lieutenant", and "sergeant"

Figure 3. Comparing the semantic capabilities between the semantic, and PHOC embedding. The figure depicts the top ten unique results for the queries "captain" (top), "orders" (middle), and "two" (bottom). For each query, the top row corresponds to the semantic embedding, while the bottom row corresponds to PHOC.

are also retrieved as secondary results. "Company" is also a semantically relevant retrieval as captain is typically the rank of the commanding officer of a company. However, the ranking is imperfect, as results like glad and circum are semantically different from captain".

Similarly for the query "orders", we first find "orders" and "order". But then we find the visually different but semantically close directions and further down we also find "instructions". For the final query "two", we see results that are related to quantities, not only other numbers like "twelve" and "twenty", but also more fuzzy quantities like "several" and "some".

V. Discussion

Our purely image based triplet-CNN learns a very powerful representation using only the relatively weak learning signal of knowing whether or not images are of the same word. This approach can be likened to the Softmax CNN from [2] or the fine-tuned CNN from [11] where they learn features by formulating a multi-class classification problem. But the triplet-CNN (70.81% MAP on IAM) significantly outperforms both the Softmax CNN (48.67% MAP) and the fine-tuned CNN (46.53% MAP). Although there are some architectural differences between the networks used, the discrepancies are too large for them to account for the performance gap. This would indicate that the triplet-CNN setup with the SoftPN loss works very well to learn a word image descriptor.

Another significant difference between the triplet-CNN and PHOCNet (and a possible source of improvement) is the use of a spatial pyramid pooling layer [21] which removes the need to resize images to a fixed size before inputing them to the network. They emphasize its use and claim it improves generalization, but they do not report the difference it makes to their MAP score. Nonetheless, adding a spatial pyramid pooling layer is a source of possible improvement for our system. Furthermore, the PHOCNet architecture is limited to embeddings that are binary, whereas our approach grants us the flexibility to embed images in any word embedding space, like [1].

For QbS on the GW dataset, the two hard-crafted embeddings perform the best. The n-gram embedding is a bit worse and even further down is the semantic embedding. This dip between the n-gram and semantic embeddings could be explained by the fact that words that might look completely different are being mapped to the same place in the embedding space, to facilitate the semantic retrieval. The n-gram embedding performs the worst by a large margin on the IAM dataset, both on QbE and QbS. This behavior is a bit odd since the semantic embedding works well, actually giving the best QbE result (which is in itself quite surprising). One possible explanation for this result is that we need to train the language model on more similar data. Perhaps the semantic embedding is more robust for transfer learning compared to the n-gram embedding. More experiments are needed to be sure.

Another result of note is that DCToW outperforms the PHOC embedding on the GW dataset, despite being nearly a sixth of the dimensionality. It is slightly worse on the IAM dataset. This could be due to a DCT resolution of 3 being a poor choice for IAM. The use of DCT-II coefficients in DCToW can be seen as a low-pass filtering of the spatial position of individual characters. The mapping is oblivious, i.e. independent of the mapping of other words. It is well-defined, given an alphabet and a DCT resolution parameter R. Contrary to PHOC, it does not need any kind of training

for a particular language. As long as the alphabet is the same, the exact same mapping can be used for e.g. old english and modern english.

Performing word spotting in a semantic space can be a situational application. We imagine that it can be desirable when you are unsure what you are looking for, e.g., when exploring a collection of documents. Ideally, a user would be able to easily switch between a semantic and verbatim search.

VI. Conclusion

We have shown the viability of using deep neural networks to create a flexible system for both query-by-example and query-by-string word spotting. Three of the embeddings that we evaluate aims for verbatim word spotting, returning words that have a similar string encoding. Of those, DCToW is a novel embedding of words into a vector space that is more compact than PHOC. Finally, we introduce a semantic embedding for word spotting and obtain novel results. Overall, our approach outperforms the previous state-of-the-art.

Acknowledgment

We would like to thank Kalyan Ram Ayyalasomayajula for his invaluable help with MATLAB, and Anders Hast. This project is a part of q2b, From quill to bytes, a framework program sponsored by the Swedish Research Council (Dnr 2012-5743), Riksbankens Jubileumsfond (Regnumber NHS14-2068:1) and Uppsala university.

References

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

[2] S. Sudholt and G. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," *arXiv preprint arXiv:1604.00187v2*, 2016.

[3] S. K. Ghosh and E. Valveny, "Query by string word spotting based on character bi-gram indexing," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 881–885.

[4] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[6] G. Lin, C. Shen, I. Reid *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation," *arXiv preprint arXiv:1504.01013*, 2015.

[7] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[8] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, T. Mikolov *et al.*, "Devise: A deep visual-semantic embedding model," in *Advances in Neural Information Processing Systems*, 2013, pp. 2121–2129.

[9] J. A. Rodriguez-Serrano, A. Gordo, and F. Perronnin, "Label embedding: A frugal baseline for text recognition," vol. 113, no. 3. Springer, 2015, pp. 193–207.

[10] S. Ghosh and E. Valveny, *A sliding window framework for word spotting based on word attributes*, ser. Lecture Notes in Computer Science. Springer Verlag, 2015, vol. 9117, pp. 652–661.

[11] A. Sharma and P. S. Kompalli., "Adapting off-the-shelf cnns for word spotting & recognition," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, Aug 2015, pp. 986–990.

[12] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[13] A. Gordo, J. Almazan, N. Murray, and F. Perronin, "Lewis: Latent embeddings for word images and their semantics," in *The IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[14] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk, "Pn-net: Conjoined triple deep network for learning local image descriptors," *arXiv preprint arXiv:1601.05030*, 2016.

[15] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2368–2376.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[17] V. Lavrenko, T. M. Rath, and R. Manmatha, "Holistic word recognition for handwritten historical documents," in *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*. IEEE, 2004, pp. 278–287.

[18] 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.

[19] F. Kleber, S. Fiel, M. Diem, and R. Sablatnig, "Cvl-database: An off-line database for writer retrieval, writer identification and word spotting," in *2013 12th International Conference on Document Analysis and Recognition*, Aug 2013, pp. 560–564.

[20] D. Aldavert, M. Rusiol, R. Toledo, and J. Llads, "Integrating visual and textual cues for query-by-string word spotting," in *2013 12th International Conference on Document Analysis and Recognition*, Aug 2013, pp. 511–515.

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