Pattern Recognition Letters
journal homepage: www.elsevier.com

Deep Neural Networks for Record Counting in Historical Handwritten Documents

Samuele Capobianco[a,**], Simone Marinai[a]

[a] *Universita di Firenze, Via S. Marta,3 , Firenze, Italy*

ABSTRACT

In this work, we study the use of Convolutional Neural Networks for counting the number of records in each page of an historical handwritten document. The initial network training is made on a large set of document images synthetically generated with a suitable tool implemented for this task. The trained network allows us to evaluate the number of records in the documents with a good accuracy that is subsequently improved with a fine-tuning performed with a limited number of real documents. In the experiments we compared three architectures on two datasets. On one benchmark dataset composed by marriage records we outperform previous results on a similar task.

1. Introduction

One important application area for the processing of handwritten documents is related to information extraction from historical documents coming from census, birth records, and other public or private collections. The pages in these documents are often semi-structured and contain a variable number of records where each record follows a general structure, but does not have a fixed number of lines or items. For instance, in a given document there might be records with more or less details according to the information available. By completely recognizing the content of these documents it is possible to reconstruct genealogies and perform demographic studies [3] [7] [9] [17].

When analyzing handwritten historical records it is possible to explicitly segment each record identifying its position in the page or to count the number of records in each page. It is clear that the former task provides more information, but is also more complex. On the other hand, often the solution of some subtasks can provide valuable information to the users before, or instead of, recognizing the whole content. In particular, when the recognition of handwriting is difficult, and the transcription is performed by human annotators, one accurate count of the number of records in one collection can provide useful information to foresee the amount of data available in the digitized documents and therefore give an estimate of the conversion costs. One example of page with records highlighted is shown in Figure 1.

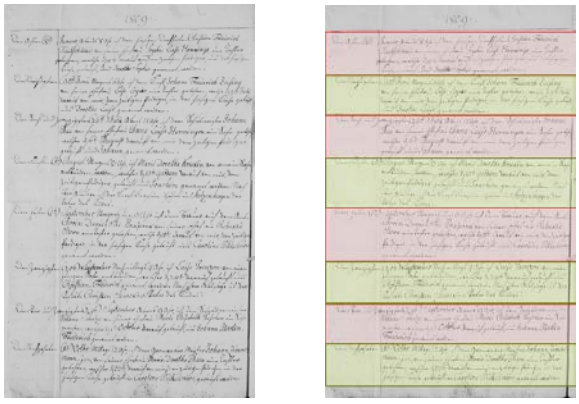


Fig. 1. The records in the document on the left are shown with different colors on the right.

One solution to the record segmentation task has been proposed in [2] where the authors perform structure detection and page segmentation applied to marriage license books. The latter books are handwritten documents where each page contains a variable number of records. Each record is composed by three main logical entities (body, name and tax) and the number of records in each page is variable. In [2] different approaches to perform record and cell segmentation (including 2D Stochastic Context Free Grammars) are compared. As an indirect result of this segmentation step it is possible to count the number of records in each page.

Object counting in an image is a relevant task in computer

**Corresponding author: Tel.: +39-055-275-8643; fax: +39-055-275-8570;
e-mail: samuele.capobianco@unifi.it (Samuele Capobianco)

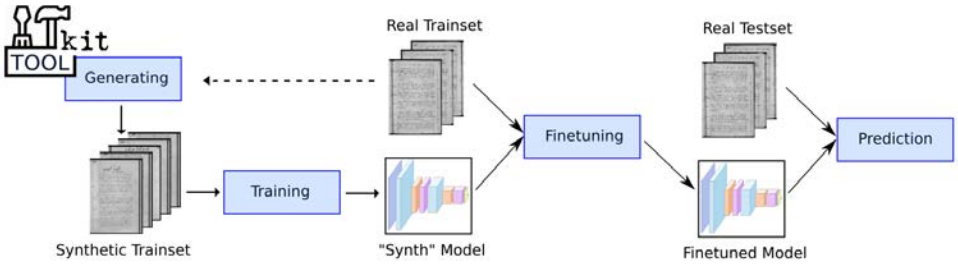


Fig. 2. An overview of the proposed system to train and predict the number of records in an handwritten collection.

vision, with several applications in real-world problems. One solution for object counting has been proposed in [13] where the authors aim to recover a density function related to one input image and enumerate the objects. The density function is a real-value functions over the pixel grid and the integral over this representation is expected to match the overall objects count.

The authors also highlight two approaches for object counting. *Counting by detection* is based on detecting the objects and then counting them. In *counting by regression* the objects are counted by using only input image features and object counting is casted as a classification (or regression) problem and to cover all the dataset variabilities a large training set is usually required.

In recent years, deep architectures have been proposed to address many computer vision tasks and object counting is not an exception. More specifically, several tasks in document image analysis applications have been often addressed with artificial neural networks [16]. In particular, convolutional neural networks have been initially used for handwritten digit recognition on the well known MNIST dataset [12] and more recently used to address other tasks (e.g. [1] [10]).

In the work described in this paper, we address the record counting problem by using deep Convolutional Neural Networks (CNNs) [12]. To the best of our knowledge, this is the first time that these techniques have been applied to this task. One related work [21] investigates the task of even digits counting in synthetic images generated from the MNIST dataset by means of CNNs. The authors apply the same approach also to count pedestrians by creating synthetic images to extend the initial dataset. People counting by using convolutional neural networks has been addressed also by [24] [25] [26]. In particular [24] and [26] compute the density function (like in [13]) by using CNNs and then the integral over the image domain of the density function is used to count the number of items. On the other hand in [25] there is no density map and one regression neuron is the network output whose value is expected to identify the number of people in the image patch. One similar approach (with one regression neuron in output) is proposed in [19] for the task of fruit counting. What is common to all these methods is the assumption that individual items to be counted (i.e. people, fruits or cells in microscopy images) cover a relatively small portion of the image. Moreover, single items can be often modeled as Gaussian kernels centered on the location of the objects (or sliglty more complex models, like in [26]). In several works it is also assumed, for training and testing purposes, that the position of each item is known and can be used for the network training.

On the other hand in the application addressed in this work the items (records) span large portions of the image, that is often nearly completely covered by records. In addition, the information about the position of each record is not available for the most challenging dataset used in our experiments. Therefore it was not possible to use the above methods for attaching the records counting task described in this work.

It is well known that to train CNNs it is important to use a large dataset of labeled instances. To the best of our knowledge there is no large data set annotated on the basis of the number of records in the pages and therefore we need to find a solution to this lack of data. This is not an uncommon problem and, when dealing with pattern recognition tasks by using learning system, common solutions to the scarcity of data are based on the generation of synthetic data and on the use of data augmentation. In the first case, synthetic data that emulate real ones are generated with one suitable application. In data augmentation, the number of real data is increased by adding distortions or noise to the existing ones. The best practice clearly requires to use the generated data only for training the system, while the performance should be always computed on real data. In the item counting task synthetic images of fruits are for instance generated in [19].

In the area of document image analysis and recognition synthetic data generation has been used for instance to model the character degradation [15] or to generate synthetic documents for performance evaluation of symbol recognition systems in the area of graphics recognition [8]. In the field of handwriting processing, handwritten documents have been generated by using standard cursive fonts with an approach that is important for our research [22].

We present in Figure 2 one overview of the proposed approach for counting records. As we can notice, one Toolkit [5] is used to generate a large dataset of pages used for the preliminary training of the CNN. Real data are then used to finetune the model and to asses the system performance in the test set. Concerning the architecture, we modified the *AlexNet* architecture [11], the *Network in Network* (*NIN*) architecture [14], and the *VGG*16 architecture [23]. In all the cases we modified the input size and the final and output layers in order to fit to the record counting problem.

To evaluate the proposed approach we used two collections: one benchmark dataset proposed in [2] for addressing the seg-

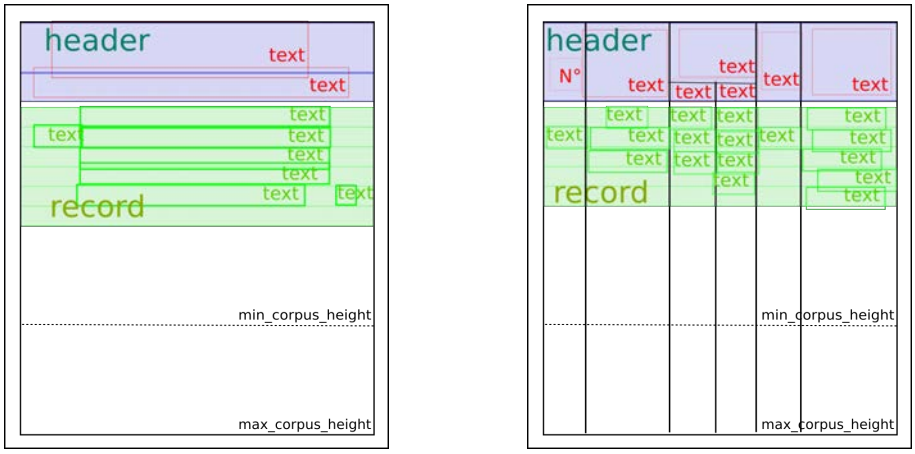


Fig. 3. Page structure defined in the XML configuration file. Left *Esposalles* dataset. Right *Brandenburg* dataset.

mentation of historical handwritten documents (we will refer to this collection as *Esposalles*); one collection composed by images provided by Ancestry (the global leader in family history and consumer genomics) through our research collaboration. The latter images were drawn from their collection *Brandenburg, Germany, Transcripts of Church Records, 1700-1874* (we will refer to this collection as *Brandenburg*). Additional details on both collections will be provided in Section 4.

There are three main contributions of this paper. First, we describe one tool for the semi-automatic generation of synthetic handwritten documents containing records. Second, we propose a simple algorithm based on the alignment of vertical ruling lines that allows us to group together similar pages from large collections of documents. Third, we demonstrate that by training a deep architecture (described in Section 3) with synthetic pages it is possible to estimate the number of records in real documents with very good results. These contributions are described in Sections 2, 4, and 5 respectively, while the conclusions are drawn in Section 6.

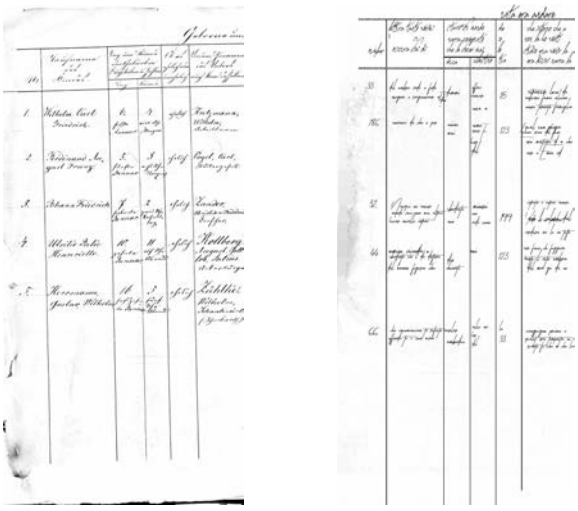


Fig. 4. Example of real (left) and synthetic (right) images.

2. Generation of Synthetic Data

To build a large training set we implemented one tool that can generate synthetic pages that look similar to real ones when considering their layout. We designed this application to be flexible and easy to configure (hiding to the user some technical details) so as to allow the generation of a broad range of different types of documents. We provide here a brief description of the tool, while additional details on it can be found in [4] and [6]. To allow further research in the field the toolkit is available as open source, together with some examples of synthetic images [5].

To infer the general record structure the user needs to analyze a few pages in the collection and describe the record structure in one XML configuration file that defines the rules needed to generate the documents. This is the only process that is performed by the user, while the other tasks for synthetic data generation are made by the application with a sequence of steps that somehow emulate the document generation. In particular, in the first step the background from some real pages is extracted and used to define the substrate where the synthetic text will be printed. In the next step the synthetic pages are generated taking into account the rules defined in the XML file.

The previous general procedure is often adopted when generating synthetic document images. The peculiarities of the tool described in this work is that the pages are generated considering the desired record structure. It is important to notice that in the record counting task there is no need to model the pages at a fine grained resolution. In particular, we assume that we can count the records in a page regardless of the actual textual content and of the actual font used to write the text. This assumption will be verified in the experimental part of this paper and is a consequence of the observation that humans can infer the location of records in pages (and their number) even without reading the actual text.

2.1. Background Extraction

The first step in the generation of synthetic pages is the extraction of the background from real documents. As we will see in the experiments (Section 5) the generation of pages with a white background produces worst results than those achieved

with a real background. To extract the page background from some training pages we first identify the pixels that surely belong to the foreground by using the Otsu [18] and the Sauvola [20] binarization algorithms. We subsequently "clean" the image by replacing the foreground pixels with the average value of background pixels in a 20×20 window centered over each foreground pixel.

In this way, we obtain a few empty pages that can be used, like ancient palimpsests, to write new synthetic documents as described in the following.

2.2. Page Generation

By means of the XML configuration file the user can define the page size, the zone of the page that will contain (when appropriate) the page header, the structure of records, and the position and range of the number of records that populate the page. It is also possible to define the dictionary used to write the text and the cursive font.

For the latter we used the "Scriptina", "A glitch in time", "Love letter tw", and "Vultures" fonts downloaded from `http://www.dafont.com` and Italian text as dictionary. For the dictionary we considered random words from one Italian text that obviously has no relationship with the real textual content of the page. Considering the resolution of the images fed to the classifier, the actual textual content has little relevance and this is in agreement with the observation that there is no need to read the document contents to infer the record location and number.

To increase the noise of the documents it is also possible to define in the configuration file the probability distribution of salt and pepper noise or other artifacts, like random white or black lines printed along the page. In some cases we need to model also ruling lines to design preprinted structured documents and it is therefore possible to include these objects that can be placed in fixed or variable locations in the page.

In Figure 3 we graphically illustrate the main areas in the page that are described in the configuration file. In particular, we show two examples corresponding to the collections used in our experiments. In both cases the banner of the page is shown in light blue and the record structure in green. We can notice that in the *Esposalles* dataset the banner and the records span the whole page width. On the other hand in the *Brandenburg* collection the banner is more complex since it is composed by fixed lines with a preprinted structure that can change from document to document. Also the record structure is more complex since there are six columns which define each record cell to be filled with text lines.

The record of the header, like other records, can contain mandatory fields (e.g. the page number) and other fields that are added according to a specified probability distribution. The main corpus area, depicted in green, defines the part of the page where records are printed. Any page generated by the tool contains records in the area between the header and any vertical position comprised between the min_corpus_height and max_corpus_height delimiters. The generation of records is based on a random selection of words from the dictionary and subsequent "writing" until each record line (the green boxes) is filled with text. If the number of lines in each record is not fixed we randomly decide their number. When the printed records reach the min_corpus_height position, to increase the variability of pages, the tool adds a random number of records to fill the document until the max_corpus_height delimiter is reached.

To illustrate the output of the synthetic generator, in Figure 4 we compare one real and one synthetic image from the *Brandburg* collection.

3. Convolutional models

The first model used to count the number of records is based on the well-known Alexnet architecture [11], where we changed the input size and replaced the output classification layer with one regression neuron, trained to count the number of records in the page. According to [21] we therefore casted the counting problem as a regression one. In Figure 5 (top) we graphically depict the corresponding architecture.

In the second model we extended the *Network in Network* (*NIN*) model proposed by Lin et al. [14]. This architecture consists of stacked blocks. Each block computes a transformation composed by a convolution operator followed by a Multi Layer Percepron (MLP). The resulting structure is called *mlpconv* layer. The *mlpconv* maps the input patch to the output feature vector. The MLP is shared among all local receptive fields and the feature maps are computed by sliding the MLP over the input. The overall model consists of four *mlpconv* transformation layers. The output of the last layer are a number of feature maps corresponding to the number of classes. Each feature map is followed by one global average pooling.

In [14] the output of the global average pooling is fed into one softmax layer. In our work we have one single feature map as output map that is followed by one global average pooling to compute the prediction about the number of records. In Figure 5 (bottom) we graphically depict the corresponding architecture.

The third model used to count the number of records is based on the VGG16 architecture [23], where we changed the input size and replaced the output classification layer with one regression neuron, trained to count the number of records in the page.

The input dimensions slightly change according to the dataset. In all the cases we scale the images preserving the original proportions to reduce the network input size and improve generalization. With the *Esposalles* dataset the input images have been scaled to 366×256, while with the *Brandenburg* dataset the images have been scaled to 450×190. In both cases we used binarised images (obtained with the Sauvola algorithm [20]) as input for both the training and test.

4. Datasets

As previously mentioned, to evaluate the proposed techniques we used the *Esposalles* collection (containing 200 pages) and the *Brandenburg* one (containing $4,956$ pages). In both cases we generated synthetic data as described in Section 2.

For the *Esposalles* collection we generated $81,060$ synthetic pages with associated information about the number of records

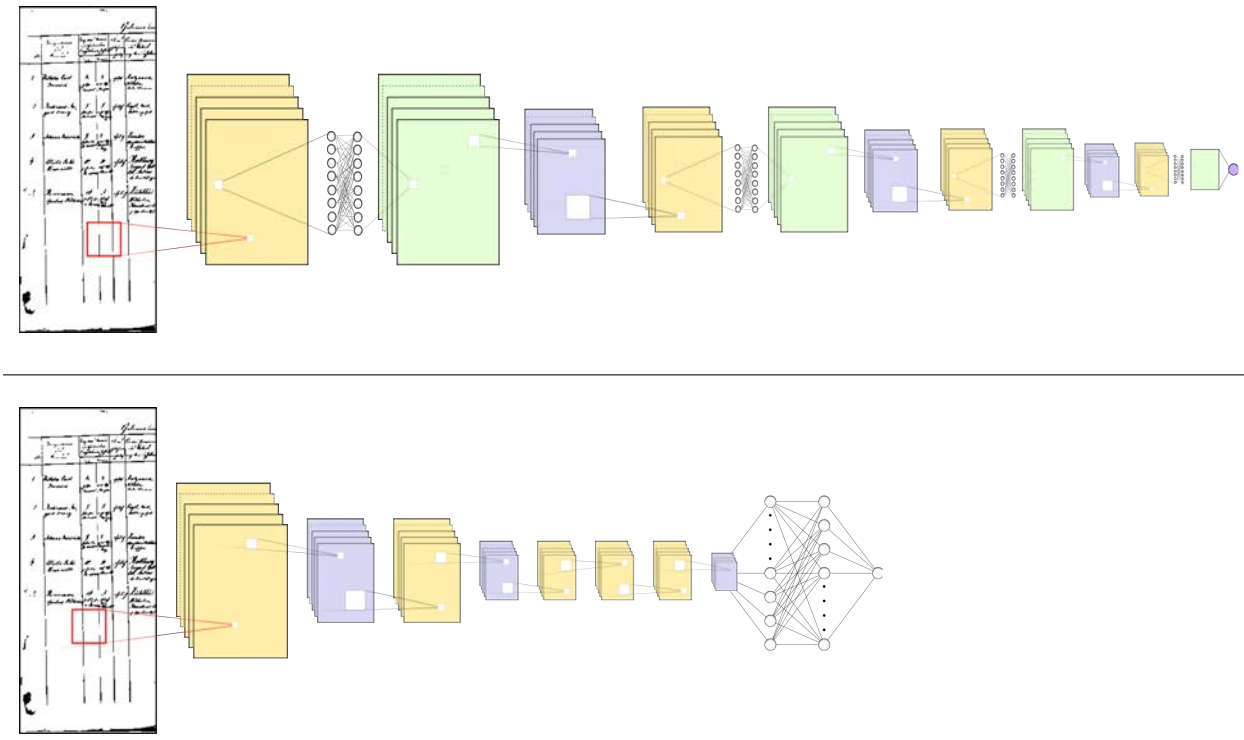


Fig. 5. Architectures used in the experiments. Top: Regression layer added to the NIN model. Bottom: regression layer added to the AlexNet model. A similar change is considered for the VGG16 architecture.

in each page. The synthetic dataset contains pages with a number of records comprised between 3 and 9 even if the benchmark collection in [2] contains only pages with 5, 6, or 7 records each.

The distribution of the 4,956 pages in the *Brandenburg* collection is as follows (where (X, Y) means X pages with Y records): $(127, 1)$, $(273, 2)$, $(356, 3)$, $(485, 4)$, $(574, 5)$, $(806, 6)$, $(813, 7)$, $(833, 8)$, $(689, 9)$. The distribution is not balanced, but there is a large number of pages for each number of records. For this collections we generated 61,914 synthetic pages with a number of records comprised between 1 and 11.

The variability of the *Brandenburg* collection is significantly higher than the *Esposalles* one not only because the number of records in each page range from 1 to 9 but mostly because the pages come from different sources and the layout is variable (some examples are shown in Figure 9).

4.1. Dataset Selection for the Brandenburg *collection*

Through our research collaboration with Ancestry we had the chance to work with one large collection containing 78,781 images were only the number of records in each page was available. However, this collection contained pages with significantly different layouts, while one assumption of our task is to deal with relatively similar images (as will be clear in the experiments there is no need to deal with identical documents).

In order to build one homogeneous subset one visual scan of all the images is clearly infeasible. To obtain the 4,956 pages we therefore designed one technique to identify pages with similar layouts taking into account the presence of pre-defined ruling lines in all the pages. One example page is shown in Figure 6.

4.1.1. Page fingerprint

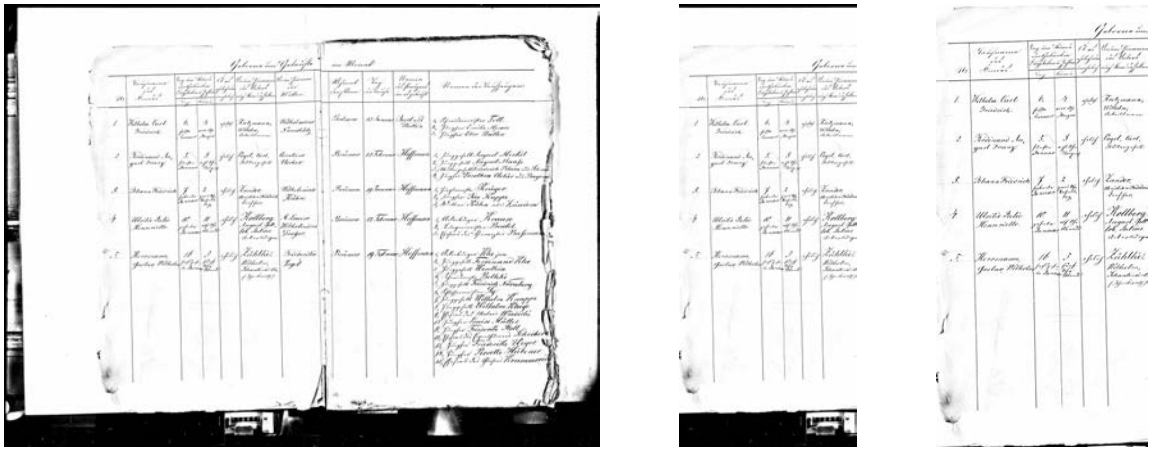
The selection of the sub-set of pages is based on a page description (like a fingerprint) that is based on the column structure that is defined by the spatial organization of the vertical ruling lines. To represent the pages we therefore first identify the vertical lines in the top half part of the page by using the Hough transform. The vertical lines identified in one page are shown in red in Figure 7. The relative line position computed in the center of the top half part of the page (along the blue dotted line) defines the page fingerprint. Along the blue line we compute the distances between neighboring lines that define the column width. Excluding lines too close to the page borders the fingerprint is made by considering the list of distances between neighboring vertical lines. In the example in Figure 7 the fingerprint is: $F = (d(1, 2), d(2, 3), \ldots, d(12, 13))$ where $d(i, j)$ is the distance between the i-th and j-th vertical lines.

4.1.2. Finding similar pages

Using the previous approach we can define one fingerprint for each page. As we can see from Figure 7 the scanned images are noisy and in particular the identification of vertical lines is not reliable in the center and at the borders of the image. This noise is incorporated in the fingerprint that is not unique for very similar pages.

In order to identify the pages most similar to one prototype page we represent this page with one subset of the page signature, one pattern, which represents only the columns that are correctly identified in most cases. For instance, the pattern that defines the example page is $P = (d(1, 2), d(2, 3), \ldots, d(5, 6))$.

The distance between one page fingerprint F and one pattern



(a) Input image (b) First segmentation (c) Second segmentation

Fig. 6. One example page from the *Brandenburg* dataset.

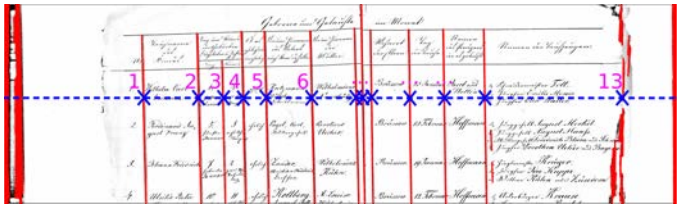


Fig. 7. Page fingerprint.

descriptor P can be computed (when $|P| < |F|$) by:

$$\mathcal{D}(F,P) = \min_{j \in 0,\ldots,|F|-|P|-1} \left(\sum_{i=0}^{|P|-1} |F_{j+i} - P_i| \right) \quad (1)$$

where $|P|$ and $|F|$ denote the length of the vector P and F respectively.

$\mathcal{D}(F,P)$ defines the smallest distance between the pattern P and one subset of the descriptor F. By using this distance it is possible to estimate the similarity between each document and the pattern and therefore select a subset of pages (with distance below a given threshold) from the larger dataset. In this way we obtained the $4,956$ pages used in the experiments in the *Brandenburg* collection.

As a side effect of this column alignment it is also possible to crop the left page in a two-page document and use only this portion for the subsequent steps. In Figure 6 we show the whole page and in Figure 6*b* the cropped area. As we can see from the image, the cropped part needs to be cleaned by removing the noisy parts on the top and bottom. These parts are removed by first binarizing the page and the identifying the largest blobs in the top and bottom part of the page obtaining the image shown in Figure 6*c*.

This last image is used as input to the neural network. Since the records span the whole two-pages image (Figure 6*a*) it is possible to count the number of records only considering the smaller image in Figure 6*c*. In this way, we can reduce the image size and we can deal with less noisy pages. It is therefore also easier to generate the synthetic pages.

5. Experiments

In this Section we describe the experiments performed to analyze the record counting. Two main approaches are considered for training the models (Figure 2).
In the first case we use only real data for the training considering the cross-validation data for stopping the training.
In the second case we first train the network using only synthetic images (real images are used as a validation set to stop the learning). This pre-trained network is subsequently fine-tuned using the real images in the training set that are different from those used as validation set.
In both cases the test is made with one disjoint set of real images and suitable measures are computed for performance evaluation.

5.1. Performance evaluation

The system performance are measured with two values.
The *Accuracy* is the percentage of pages where the number of records is correctly identified. The *Error* is the percentage of errors in the record count when making a decision on one page at a time. This value is defined according to Equation (2) where r_i is the actual number of records in page i, p_i is the predicted value ($\lfloor p_i + \frac{1}{2} \rfloor$ is the rounded predicted value), and N is the number of test pages.

$$Error = \frac{\sum_{i=1}^{N} \left| \lfloor p_i + \frac{1}{2} \rfloor - r_i \right|}{\sum_{i=1}^{N} r_i} \quad (2)$$

5.2. Esposalles

In the *Esposalles* collection we performed several experiments to identify the best training strategy and compare the results obtained by the proposed approach with those described in [2] considering the same split of the data in training, validation, and test datasets (we refer to this partitioning of data as Benchmark Split). Due to the limited size of the dataset in [2] we performed also a stratified cross validation to estimate the error rate on a larger dataset.

Table 1. *Esposalles* dataset. Comparing different backgrounds in the benchmark split.

Model	Initial weights	Training	Accuracy (%)	Error (%)
AlexNet	Random	Synth/W	85.0	2.6
AlexNet	Synth/W	Real+DA	95.0	0.8
AlexNet	Random	Synth/E	90.0	1.7
AlexNet	Synth/E	Real+DA	97.5	0.4

Table 2. *Esposalles* dataset. Random initialization vs pretrained Imagenet in the benchmark split.

Model	Initial weights	Training	Accuracy (%)	Error (%)
AlexNet	Imagenet	Synth	100.0	0.0
AlexNet	Random	Synth	90.0	1.7
AlexNet	Synth	Real+DA	97.5	0.4
NIN	Imagenet	Synth	100.0	0.0
NIN	Random	Synth	95.0	0.9
NIN	Synth	Real+DA	100.0	0.0

Table 3. *Esposalles* dataset. Results on benchmark split.

Model	Initial weights	Training	Accuracy (%)	Error (%)
AlexNet	Imagenet	Real	87.5	2.14
AlexNet	Imagenet	Synth	100.0	0.0
NIN	Imagenet	Real	100.0	0.0
NIN	Imagenet	Synth	100.0	0.0
VGG16	Imagenet	Real	92.5	1.29
VGG16	Imagenet	Synth	100.0	0.0
[Alvaro et al][2]			80.0	-

Table 4. *Esposalles* dataset. Average results with five fold cross-validation.

Model	Initial weights	Training	Accuracy (%)	Error (%)
NIN	Imagenet	Real	94.5	0.9
NIN	Imagenet	Synth	97.1	0.5
VGG16	Imagenet	Real	86.1	2.4
VGG16	Imagenet	Synth	93.5	1.1

5.2.1. Esposalles*: Benchmark Split*

In the Benchmark experiments we compare the results obtained by our system with those presented in [2] using the same split (150 pages for training, 10 for validation, and 40 for test).

In the first experiment (Table 1) we evaluated the impact of the use of a white background (Synth/W) instead of the background extracted with the procedure described in Section 2.1 (Synth/E) when building the synthetic dataset. In both cases the initial training is made with the synthetic images and the 160 training and validation images are used to stop the training. In the fine-tuning we first augmented the 160 training and validation images (see [4] for additional details) and therefore used 1350 pages for fine-tuninng and 90 as a validation set. We compared the two training data using the AlexNet architecture and concluded that in general it is better to use a real background than a white one. In the subsequent experiments we always used extracted backgrounds.

In the second experiment we verified that training a network pretrained on Imagenet might provide better results with respect to a random initialization of the weights (Table 2). In particular, for both the AlexNet and the NIN architectures one 100% accuracy is achieved by training the pretrained network with synthetic data only. On the other hand when using randomly initialized weights we need to perform a finetuning with real data to achieve the 100% accuracy with NIN (97.5% with AlexNet). In the subsequent experiments with the *Esposalles* dataset we trained networks pretrained on Imagenet.

In the third experiment we considered one recent deep model still pre-trained on Imagenet (the *VGG*16 architecture [23]). In Table 3 we report the results obtained on the benchmark split when considering three architectures: AlexNet, NIN, and VGG16. For each architecture we compare the performance when finetuning the Imagenet network with real data (without the data augmentation considered in the first two experiments) and when training it with synthetic data like in the second experiment, but without performing a finetuning. We can notice that the training with real data has in general worst performance with respect to the training with the synthetic data.

In Table 3 we also compare the results achieved by our system with those reported in [2] where the right number of records is predicted for 80% of the test documents. It is very important to observe that the system in [2] is designed to segment the records and therefore the record counting is only one additional information that is computed from the segmentation.

From this experiment we can notice that the networks trained with Synthetic data provide one 100% accuracy also without finetuning. With these data, the NIN slightly outperforms the other architectures.

5.2.2. Esposalles*: Five Fold Cross-validation*

Since the number of labeled pages in the *Esposalles* collection is limited, in this experiment we tested the system with a five fold stratified cross validation. The *Esposalles* dataset contains 44 pages with 5 records, 152 with 6 records and only 4 with 7 records. The five folders contain, on average, 40 pages each. In each fold we used 160 pages to perform the fine-tuning using 142 pages for training and 18 as validation set; we left the remaining 40 pages for evaluating the performance on real data.

In Table 4 we report the average values of the results obtained in the five folders. In this experiment we considered the two most promising architectures: *NIN* and *VGG*16. Like in Table 3 we compared the training using only real data of networks pretrained on Imagenet and the training the same networks us-

Table 5. ***Brandeburg*** **dataset. Comparison of different architectures (Random initialization of weights).**

Model	Initial weights	Training	Accuracy (%)	Error (%)
Alexnet	Random	Real	66.53	6.19
Alexnet	Random	Synth	46.63	12.40
Alexnet	Synth	Real	74.87	4.40
NIN	Random	Real	74.47	4.53
NIN	Random	Synth	51.26	9.84
NIN	Synth	Real	77.79	3.96
NIN + FC	Random	Real	76.38	4.18
NIN + FC	Random	Synth	48.34	10.72
NIN + FC	Synth	Real	76.98	4.06
NIN softmax	Random	Real	69.35	6.93
NIN softmax	Random	Synth	45.60	14.26
NIN softmax	Synth	Real	69.09	6.98

Table 6. ***Brandeburg*** **dataset. Comparison of different architectures (Imagenet initialization of weights).**

Model	Initial weights	Training	Accuracy (%)	Error (%)
Alexnet	Imagenet	Real	70.55	5.26
Alexnet	Imagenet	Synth	55.58	10.09
Alexnet	Synth	Real	79.90	3.66
NIN	Imagenet	Real	82.51	2.98
NIN	Imagenet	Synth	57.59	9.14
NIN	Synth	Real	83.22	3.08
VGG16	Imagenet	Real	86.23	2.35
VGG16	Imagenet	Synth	60.20	7.46
VGG16	Synth	Real	85.93	2.48

ing the synthetic data. In both cases one network trained using only synthetic data outperforms one network trained using a very limited number of real data. It is worth recalling here that for each folder we use for the Real training on average 142 real training data (the validation set contains 18 pages). For the Synth training we use $81,060$ synthetic pages while the validation and test sets are the same of the Real training.

5.3. Brandenburg

The second main group of experiments is made using the $4,956$ pages of the *Brandenburg* collection described in Section 4. Since this dataset is significantly larger than the *Esposalles*, we made more detailed experiments. In particular, we analyze the change of performance when varying the training set size. We split the pages in three sets: $3,165$ pages for training, 796 for validation and 995 for test.

Once again the first experiments have been performed to identify the most promising architectures (Table 5). We considered architectures based on the AlexNet and NIN models using a random initialization of weights. For NIN we considered one version with the global average pooling as last layer (NIN in Table 6) and one with a fully connected layer as last layer (NIN+FC in Table 6). In the last case the weights are initialized to compute the average over the input features map. The best results are obtained with the NIN with global average pooling. We also considered one architecture with the output based on one softmax layer with 9 outputs corresponding to the number of classes (NIN softmax in Table 6). Not surprisingly, in the latter case we obtained worst results.

Also in this first experiment with the *Brandeburg* dataset the NIN architecture outperforms the AlexNet one confirming the results for the Esposalles collection. In almost all the cases, the finetuning of a network initially trained on synthetic data improves the results on the test set with respect to the training only on real data (without data augmentation).

One exception is the NIN softmax experiment. However, it is worth noticing that the *Accuracy* and *Error* are worst than in the other tests. Apart from the softmax NIN in all the cases the finetuning of a network trained with synthetic data has higher *Accuracy* and lower *Error* with respect to a network trained on real data only.

Since this collection is larger than the *Esposalles* one we considered also one experiment where all the training and test are made with real data and the networks are pretrained on Imagenet (Table 6). For the Alexnet and NIN architectures the best results are obtained when finetuning with real data one network initially trained with synthetic data. On the other hand the training with only real data on VGG16 pretrained on Imagenet provides the best overall results.

One of the claims of this paper is that when the number of training data available is limited, then the performance can be improved by using synthetic data. In order to investigate the effect of the number of training pages used for finetuning, in Table 7 we show the results of two of the experiments described in Table 6 (training with real data one network pretrained on Imagenet and finetuning with real data one network trained on synthetic data) when using different sizes for the real training set. For example the smallest training set contains 396 images ($1/8$ of the total $3,165$ pages available for training) and the corresponding validation set contains 104 images. The test sets are left unchanged with respect to the experiments in Table 6.

With the exception of VGG16 trained with 100% of the data in all the cases the finetuning of a network trained with synthetic data outperforms the same model only trained on real data. We can also notice that in general the gap between the two approaches narrows when the size of the training set increases.

To provide additional information on the results we show in Figure 8 the confusion matrix for the results obtained with the VGG16 model. In is worth to notice that for nearly all the types of pages the results are in a range of ± 1 with respect to the correct record count.

To further analyze the results in Figure 9 we show some pages in the collection with five records. From these examples we can notice how variable are the pages in the collection. The pages are grouped in the Figure according to the results achieved when training one VGG116 model with the two experiments summarized in Table 7. The number of records for pages

Table 7. *Brandeburg* dataset. Varying the size of the data used for training or for finetuning.

Model	Measure(%)	Initial weights	Training	Trainset size 1/8	1/4	1/2	1/1
Alexnet	Accuracy	Imagenet	Real	44.22	51.15	57.39	70.55
		Synth	Real	65.93	71.96	77.89	79.90
	Error	Imagenet	Real	11.55	9.61	7.99	5.26
		Synth	Real	6.89	5.43	4.13	3.66
NIN	Accuracy	Imagenet	Real	56.08	61.81	73.57	82.51
		Synth	Real	67.44	71.76	76.18	83.22
	Error	Imagenet	Real	8.49	7.04	4.61	2.98
		Synth	Real	6.29	5.48	4.51	3.08
VGG16	Accuracy	Imagenet	Real	70.45	72.96	79.19	86.23
		Synth	Real	74.07	76.98	80.60	85.93
	Error	Imagenet	Real	5.56	4.89	3.60	2.35
		Synth	Real	4.83	4.26	3.46	2.48

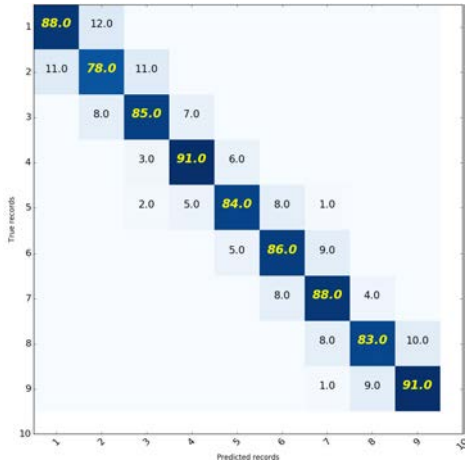


Fig. 8. Confusion matrix for the best VGG16 model.

in the first group is correctly identified with both approaches. On the other hand the number of records for pages in the second group is not identified by either methods. For instance, the first two pages in the group have a long comment at the bottom, while in the last one the first record has been canceled and therefore is not counted in the ground-truth.
For the groups in the bottom part of the Figure the two approaches provide different results. In the first two pages the first record is very difficult to read and faded and this type of degradation was probably not properly modeled by the synthetic document generator. The latter tool was however able to model the irregular distribution of records in the last three examples, that are probably rare in real training pages.

6. Conclusions

In this paper we addressed the identification of the number of records in handwritten historical documents by using Convolutional Neural Networks trained on synthetic data. We performed several experiments to study the capability of synthetic data to model real handwritten documents belonging to two different collections. In the experiments we tested three well known deep architectures that we modified to compute the number of records in a page. With both datasets the NIN-based architecture provided better results than the one based on AlexNet. On the other hand the VGG16 architecture shown better results in the larger *Brandenburg* collection.

We are currently studying the information that can be extracted from the feature maps in the convolutional neural networks. This type of information can be of interest for understanding which parts of the input image are more informative to compute the record counting task. The latter study may lead at the end to address the record segmentation with deep convolutional networks.

Acknowledgments

We would like to thank the authors of [2] for sharing the annotated dataset used to perform the experiments.
We thank also the anonymous reviewers for their insightful comments and suggestions that helped us to improve the manuscript. This work is partially supported by a research grant from Ancestry.com.

References

[1] Afzal, M.Z., Capobianco, S., Malik, M.I., Marinai, S., Breuel, T.M., Dengel, A., Liwicki, M., 2015. Deepdocclassifier: Document classification with deep convolutional neural network, in: 13th Int.l Conf.Document Analysis and Recognition,, pp. 1111–1115.

[2] Alvaro, F., Fernandez, F.C., Sánchez, J., Terrades, O.R., Benedí, J., 2015. Structure detection and segmentation of documents using 2D stochastic context-free grammars. Neurocomputing 150, 147–154.

[3] Bulacu, M., van Koert, R., Schomaker, L., van der Zant, T., 2007. Layout analysis of handwritten historical documents for searching the archive of the cabinet of the dutch queen, in: 9th Int.l Conf.Document Analysis and Recognition, pp. 357–361.

[4] Capobianco, S., Marinai, S., 2016. Record counting in historical handwritten documents with convolutional neural networks, in: Proc. 1st Int.l Workshop on Deep Learning for Pattern Recognition, DLPR. URL: `http://arxiv.org/abs/1610.07393`.

[5] Capobianco, S., Marinai, S., 2017. Docemul: a toolkit to generate structured historical documents. `https://github.com/scstech85/DocEmul`.

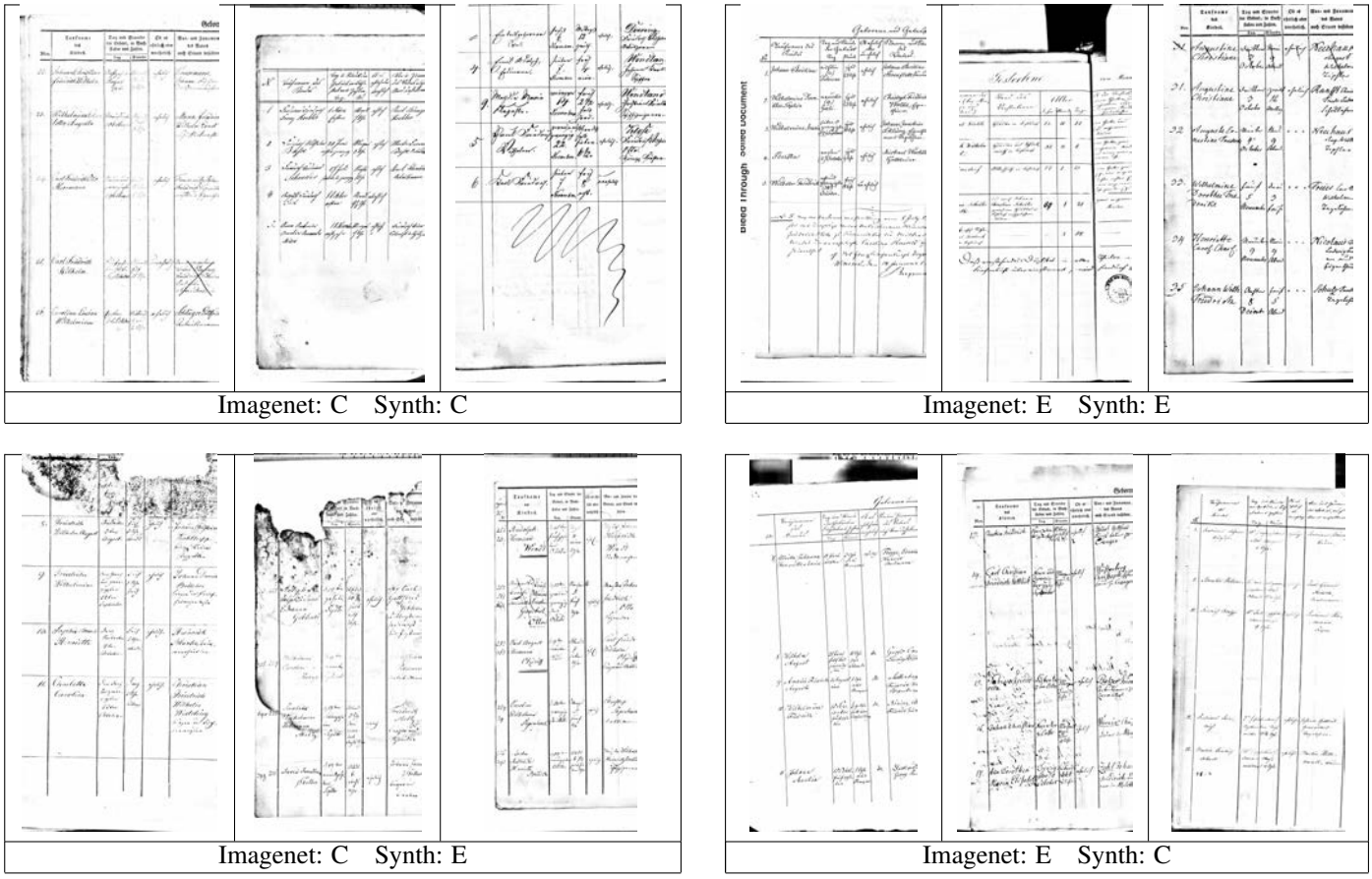


Fig. 9. ***Brandenburg*** **dataset. Examples of pages and record counting results for the VGG16 model and training starting from Imagenet or Synthetic initial weights (as in Table 7). C means a correct record count, while E means an error in the record count.**

[6] Capobianco, S., Marinai, S., (accepted) 2017. Docemul: a toolkit to generate structured historical documents, in: 14th Int.l Conf. Document Analysis and Recognition.

[7] Cruz, F., Terrades, O., 2014. EM-based layout analysis method for structured documents, in: 22nd Int.l Conf. on Pattern Recognition, pp. 315–320.

[8] Delalandre, M., Valveny, E., Pridmore, T., Karatzas, D., 2010. Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. International Journal on Document Analysis and Recognition (IJDAR) 13, 187–207.

[9] Fernández, D., Marinai, S., Lladós, J., Fornés, A., 2013. Contextual word spotting in historical manuscripts using Markov logic networks, in: Proc. 2nd Int.l Workshop on Historical Document Imaging and Processing, HIP@ICDAR13, pp. 36–43.

[10] Kim, I.J., Xie, X., 2015. Handwritten Hangul recognition using deep convolutional neural networks. International Journal on Document Analysis and Recognition (IJDAR) 18, 1–13.

[11] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105.

[12] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition, in: Proceedings of the IEEE, pp. 2278–2324.

[13] Lempitsky, V., Zisserman, A., 2010. Learning to count objects in images, in: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (Eds.), Advances in Neural Information Processing Systems 23. Curran Associates, Inc., pp. 1324–1332.

[14] Lin, M., Chen, Q., Yan, S., . Network in network, in: Proceedings of ICLR, 2014.

[15] Luyen, D.T., Carel, E., Ogier, J.M., Burie, J.C., 2015. A character degradation model for color document images, in: Proc. 13th Int.l Conf. Document Analysis and Recognition, pp. 806–810.

[16] Marinai, S., Gori, M., Soda, G., 2005. Artificial neural networks for document analysis and recognition. IEEE Trans. Pattern Anal. Mach. Intell. 27, 23–35.

[17] Nielson, H., Barrett, W., 2006. Consensus-based table form recognition of low-quality historical documents. International Journal of Document Analysis and Recognition (IJDAR) 8, 183–200.

[18] Otsu, N., 1979. A Threshold Selection Method from Gray-level Histograms. IEEE Transactions on Systems, Man and Cybernetics 9, 62–66.

[19] Rahnemoonfar, M., Sheppard, C., 2017. Deep count: Fruit counting based on deep simulated learning. Sensors 17, 905.

[20] Sauvola, J.J., Pietikäinen, M., 2000. Adaptive document image binarization. Pattern Recognition 33, 225–236.

[21] Segui, S., Pujol, O., Vitria, J., 2015. Learning to count with deep object features, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops.

[22] de França Pereira e Silva, G., Lins, R.D., Gomes, C., 2014. Automatic training set generation for better historic document transcription and compression, in: 11th Int.l Workshop on Document Analysis Systems, DAS, pp. 277–281.

[23] Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition, in: Proceedings of ICLR 2014. URL: `http://arxiv.org/abs/1409.1556`.

[24] Walach, E., Wolf, L., 2016. Learning to Count with CNN Boosting. Springer International Publishing, Cham. pp. 660–676. URL: `https://doi.org/10.1007/978-3-319-46475-6_41`.

[25] Wang, C., Zhang, H., Yang, L., Liu, S., Cao, X., 2015. Deep people counting in extremely dense crowds, in: Proc. 23rd ACM Int.l Conf. Multimedia, pp. 1299–1302.

[26] Zhang, C., Li, H., Wang, X., Yang, X., 2015. Cross-scene crowd counting via deep convolutional neural networks, in: IEEE Conf. Computer Vision and Pattern Recognition (CVPR), pp. 833–841.