Text Line Extraction in Handwritten Historical Documents

Samuele Capobianco(✉) and Simone Marinai

Dipartimento di Ingegneria dell'Informazione,
Università degli Studi di Firenze, Florence, Italy
{samuele.capobianco,simone.marinai}@unifi.it
http://ai.dsi.unifi.it/

Abstract. We present a novel approach for the extraction of text lines in handwritten documents using a Convolutional Neural Network to label document image patches as text lines or separators. We first process the document to identify the most suitable patch size on the basis of an overall text line distance estimation. Using this information, we then extract several patches to train the CNN model. Finally, we use the trained model to segment text lines. We have evaluated this technique on the public database Saint Gall and on a private collection provided by the Ancestry company.

1 Introduction

Handwritten historical documents are widely available in Digital Libraries and archives. In particular, handwritten forms have been used in the past to record important personal facts. Significant examples are documents containing census information, birth records, and other public or private collections. The analysis of these documents is essential to reconstruct genealogies and to perform demographic studies. One first step to address the automatic transcription and information extraction from these documents is to localize and extract the text lines. Several techniques have been presented to address this task [1] and recent work focused on the use of artificial neural networks to find text lines in handwritten documents [2]. In this work we propose one solution to recognize the text lines and the separation between text lines using a Convolutional Neural Network. In the last few years, Convolutional Neural Networks [3] have been widely used to solve several tasks. In particular, these techniques have been used to segment [4] and localize objects [5] in Computer Vision as well as for page classification [6].

In our research we deal with documents having different layouts. However, one common feature is that in each page the distance between text lines is quite regular. We can use this prior to initialize the document analysis; subsequently by using a suitably trained neural model we can localize the text lines and the line separators. As in most training-based approaches, the solution is based on two main steps.

We first train a suitable CNN model. To this purpose, for each training page we estimate the average distance between text lines and use this information to

C. Grana and L. Baraldi (Eds.): IRCDL 2017, CCIS 733, pp. 68–79, 2017.
DOI: 10.1007/978-3-319-68130-6_6

label random patches used to train the model. Subsequently, we design a text line separator. For each test page we estimate the average distance between text lines and use this information to extract dense patches that are classified using the trained model. The classified patches are then combined to obtain an overall page segmentation.

In the following sections we first describe in detail the initial step aimed at computing the adaptive patch dimension. We then analyze the proposed application scenario and discuss the experiments performed on two datasets.

2 Adaptive Estimation of Patch Dimension

The proposed approach to solve the line segmentation is based on a patch-wise representation of the document images. The patch is expected to cover an area of the image containing text and background. In the pre-processing phase we therefore need to estimate an average distance between text lines. As we can notice from the example in Fig. 1a in our data there is a certain regularity in the distance between lines in each page and this regularity can be used to adapt the patch dimension in the page. Subsequently, we can label the patch as text line separator (if it covers the space, or other separators, between text lines) or as text (if it contains text in the middle of the patch area).

2.1 Estimate Average Text Line Distance

To estimate the average text line distance we measure the vertical projection profile of the page and then compute the distribution of the distance between peaks in the profile. The distribution is fitted into two clusters and the smallest value is used as estimate for the distance between text lines.

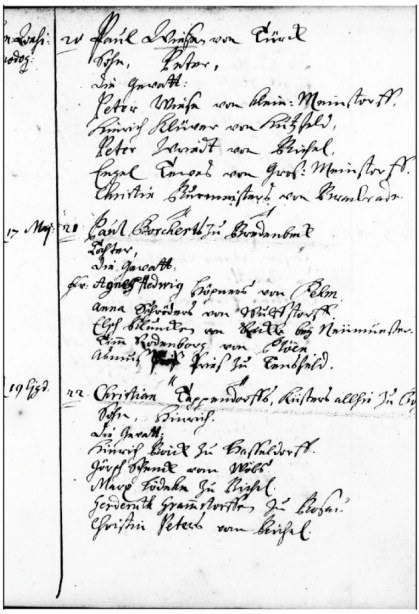
In details the computation is composed by several steps. In the first step we remove black bands at the top and bottom borders of the page. Then, we compute the projection profile along the vertical dimension and find its local maxima. We then retain only the peaks larger than 20% of the maximum.

Considering these peaks, we can compute the distribution of the distance between pairs of neighboring peaks. This set of values can be fitted by two centroids using the k-means algorithm. At the end we take the minimum centroid which defines the average distance between lines. The main steps are depicted in Fig. 1 where we show one input image, the computed projection profiles, and the peaks used to define the distance distribution.

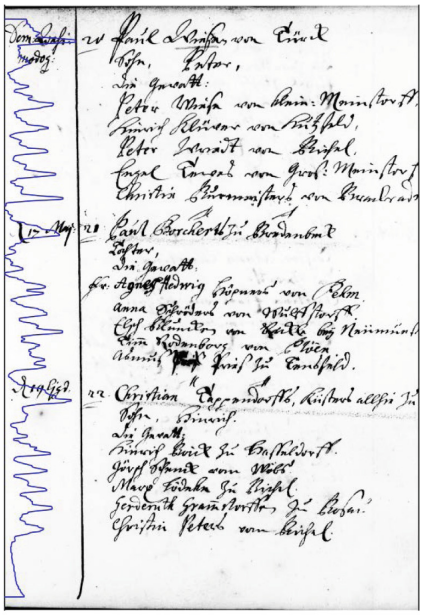
With this procedure it is possible to estimate the text line distance in each page. This distance is subsequently used to define the size of the patches extracted from the page.

2.2 Patch Definition

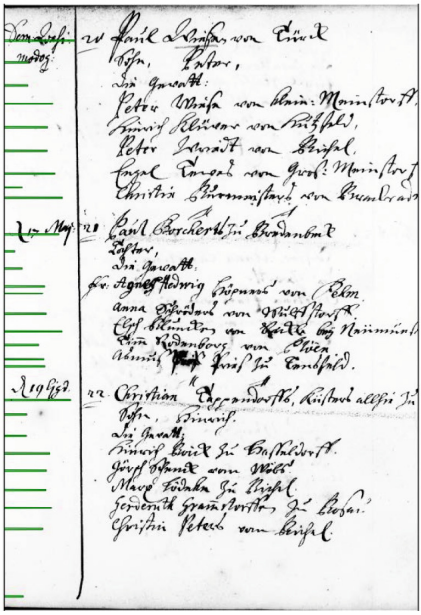
After the computation of the average text line distance, $\hat{h}$, we define the patch size (height, width) as $(\hat{h} \cdot \sqrt{2}, \frac{\hat{h}}{\sqrt{2}})$. The patch height is defined in order to



(a) Input image



(b) Projection lines



(c) Computed maximum peaks

Fig. 1. Steps to estimate the distance between text lines in one page.

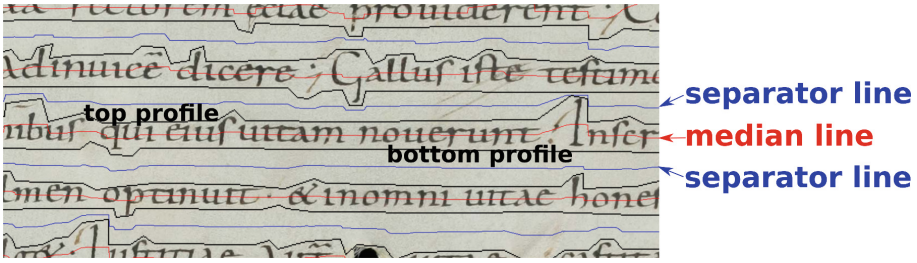


Fig. 2. Examples of separator and median lines and the corresponding top and bottom profiles.

capture almost two text lines with one separator (space or other) in between. The patch size has been chosen after some preliminary tests. In general we can observe that the classification of variable-size patches is a good option to be independent from the textline distance in the input image. A dense patch extraction with overlapping can cover all the lines providing a good hint on the presence of text line separators.

3 The Proposed Method

Taking into account the physical structure of the documents we can define two types of areas: the text line and the separator that is the area between two contiguous text lines. For the subsequent steps it is also useful to define the median line and the separator line. The median line is the middle line between the top profile of the text and the bottom profile. The separator line is the middle line between two consecutive median lines in the text area. We can see these lines in Fig. 2, in particular we show three text lines extracted from one page representing the top and bottom profiles with black lines, the separator and median lines in blue and red. These two types of lines can be used to label the extracted patches and subsequently train the CNN-based line segmentation. However, this information is not always available as groundtruth for the datasets. We therefore need to compute this representation from the available ground truth data.

For example, the ground truth of the *Saint Gall* dataset (more details in Sect. 4.1) is available as pixel-label images defining the text area. In this case we can compute the median line for each text line starting from the pixel level information. After that, we compute the separator line as the mid point between two consecutive median lines. In Fig. 6 we can see one example of the *Saint Gall* dataset and the corresponding ground truth information. Using this information we can compute the previously defined guidelines to label the extracted patches.

Unfortunately, sometimes the ground truth data does not contain this pixel-level labeling. In these cases we manually define only the separator lines and use this information to identify also text lines. More details on how to extract and label patches are described in Sect. 4. Using the document structure and

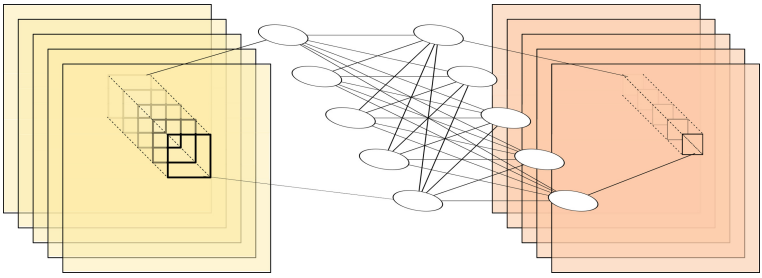


Fig. 3. The *mlpconv* layer is composed by three transformations, the first is a classic convolution followed by two inner products to map the results in the output feature maps.

the ground truth information, we can label each patch as separator, text, or background. The assigned class depends on the area covered by the patch on the document.

3.1 Network Architecture

Convolutional Neural Networks (CNN) are a particular type of Artificial Neural Networks that consist of alternated convolutional layers and spatial pooling layers [7]. The convolutional layers generate feature maps by linear convolutional filters followed by nonlinear activation functions (e.g. rectifier, sigmoid, tanh). In our work we extend the *Network in Network* (NIN) model proposed by Lin et al. [8]. This architecture (Fig. 4) consists of stacked blocks where each block is composed by a convolution operator followed by a Multi Layer Perceptron (MLP) (Fig. 3). This latter structure is called *mlpconv* layer.

The *mlpconv* maps the input area to the output feature vector with a multilayer perceptron (MLP) consisting of two fully connected layers with nonlinear activation functions. The MLP is shared among all local receptive fields. The feature maps are computed by sliding the MLP over the input similarly to CNNs and are then fed into the next layer. We show one example of *mlpconv* layer in Fig. 3. This layer computes a convolutional transformation with a defined kernel dimension followed by two inner product transformations. Each transformation layer is followed by a ReLU [9] layer. The results are computed considering the same number of feature maps for each transformation in the *mlpconv*.

In this work we use a modified version of the original architecture. In the first *mlpconv* layer we have a convolutional layer with a kernel dimension of $3 \times 7 \times 7$ computing 96 feature maps with stride 2 and padding 3. In the second *mlpconv* layer we have a convolutional layer with a kernel dimension $96 \times 5 \times 5$ computing 256 feature maps with stride 1 and padding 2. In the third *mlpconv* layer we have a convolutional layer with a kernel dimension $256 \times 3 \times 3$ computing 384 feature maps with stride 1 and padding 1. After that, we have a dropout layer [10] which randomly selects 50% of input neurons during the training phase. In the fourth *mlpconv* layer we have a convolutional layer with a kernel dimension $384 \times 3 \times 3$

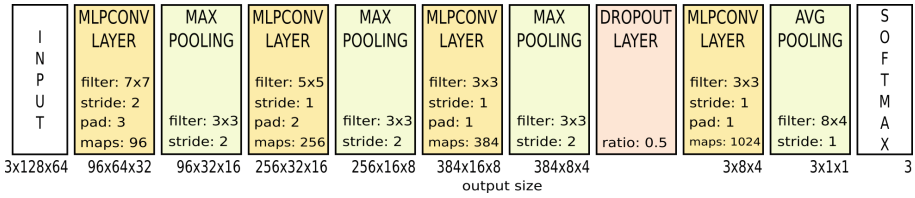


Fig. 4. The architecture of the model. Different transformations are depicted with different colors. The *mlpconv* transformation is depicted in orange, the pooling undersampling in green and the input and output layer in white. The output result is composed by three classes. (Color figure online)

computing 1024 feature maps with stride 1 and padding 1. For the last *mlpconv* layer, we define the number of feature maps according to the number of classes in our task. We compute the global average pooling over the previous feature maps providing them to the softmax layer which models the distribution over the class labels. We use the stochastic gradient descent to train a model initialized with random weight values. The training is stopped on the best classification accuracy on a validation set. We use the Adam optimization algorithm [11] with a fixed learning rate 0.0001 with momentum (0.9, 0.999) for the training phase.

The overall architecture can be seen in Fig. 4. The model is composed by four *mlpconv* and pooling layers. In orange, we denote the *mlpconv* transformation, in green the pooling transformation layer. For the last *mlpconv* we compute 1024 feature maps until the second inner product, instead, for the last inner product we have the number of neurons according to the output classes.

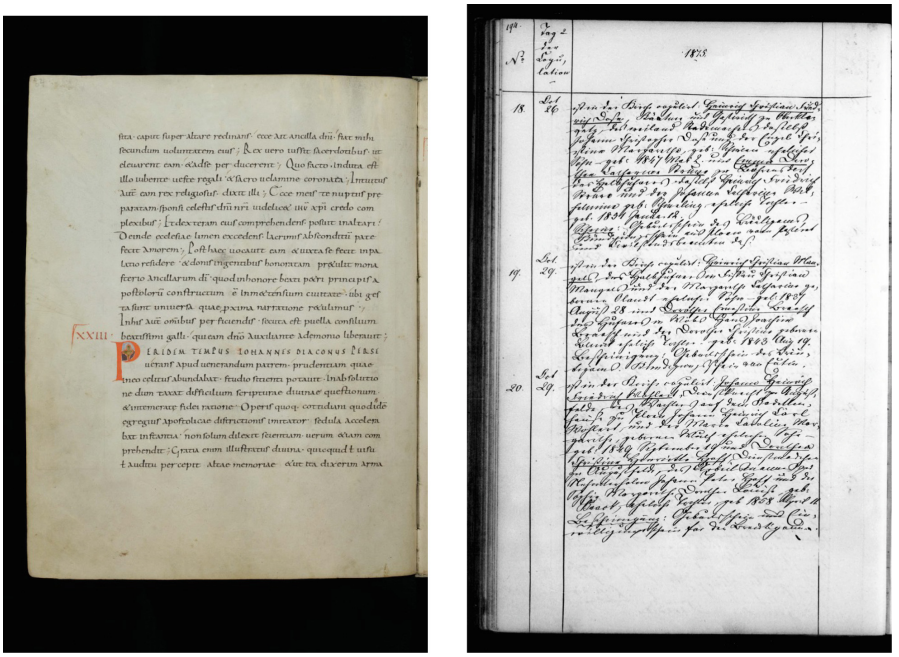
4 Experiments

In this section we describe the experiments for localizing text lines using two different dataset.

4.1 Data Corpus

It is no easy to find public databases for the tasks addressed in this work. The HisDoc [12] project presents some datasets for developing handwriting recognition systems. In particular, we use the *Saint Gall* dataset that consists of 60 pages scanned from one medieval manuscript written in Latin (Fig. 5a). The ground truth information is gathered from the Hisdoc Divadia site[1]. The proposed approach has been tested also on handwritten documents from a private collection provided by the Ancestry company (the global leader in family history and consumer genomics). This database consists of 54 structured documents written in English by several writers (Fig. 5b). The line ground truth has been produced using one tool specifically developed to manually segment the lines. The images

[1] http://diuf.unifr.ch/main/hisdoc/divadia.



(a) Saint Gall (b) Ancestry

Fig. 5. One example page for each dataset.

in this collection have been also used for performing experiments aiming at automatically counting the number of records in handwritten pages [13].

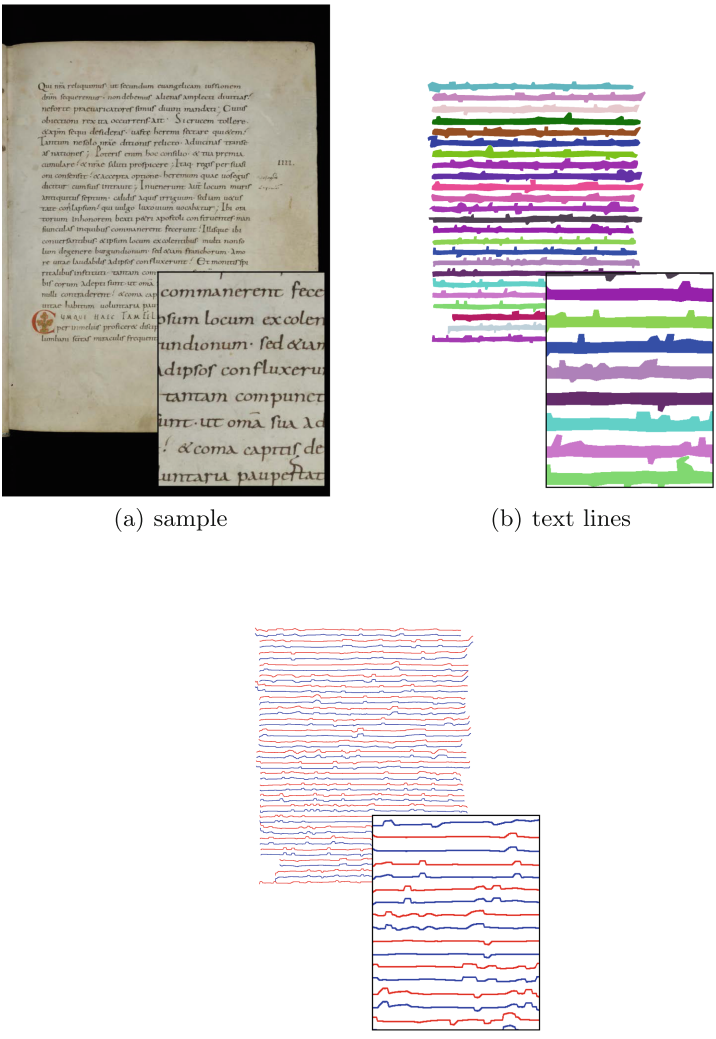
4.2 Extracted Patches

As previously mentioned, the first step is to extract the patches used to train the model. This extraction is based on the ground truth of the training set. As previously mentioned, for this research we consider two datasets with two different ground truth information.

In the *Saint Gall* dataset the ground truth is very detailed since it contains one XML file for each document. The XML file contains one pixel-wise representation for each text line. One example of this dataset is presented in Fig. 6 where we can see an input document, a pixel-wise representation of the ground truth for each text line, the computed median text line, and the text separator.

The ground truth produced for the Ancestry dataset defines only the separator line between two adjacent text lines. In Fig. 7 we can see one image example and its ground truth image.

Considering these labeled datasets we can build a large patch dataset to learn the CNN. We have seen how to compute the average distance between text lines; using this information we can define a patch with a suitable dimension

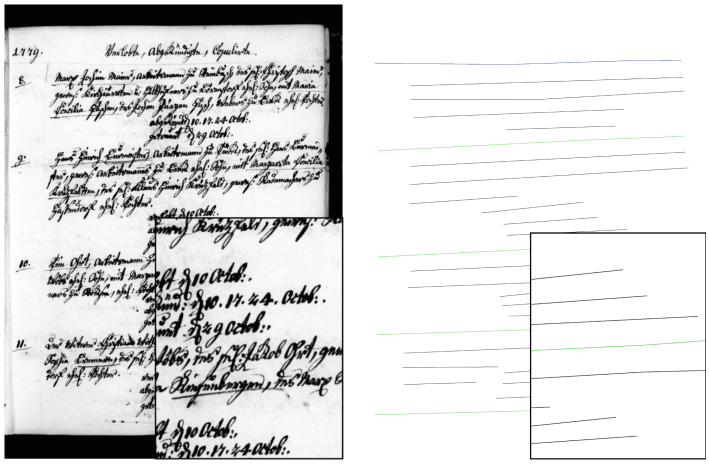


(a) sample (b) text lines

(c) median and separator lines

Fig. 6. In (a) one page from the *Saint Gall* dataset, in (b) the corresponding ground truth pixel-wise representation for each text line, in (c) the computed median text lines in red and the separator text lines in blue. (Color figure online)

to capture the document features. The creation of the image patches for the training set follows the ground truth guidelines, therefore in the first dataset we are able to define patches for three classes (text, separator and background) while only two classes can be considered for the second dataset. We can easily control the number of training patches for each class and we can therefore extract



(a) Input (b) Ground truth

Fig. 7. An input image from Ancestry collection with its ground truth image.

approximately the same number of patches for each class. We then scale each extracted patch to a fixed size of 128×64 pixels.

In the two experiments for each class we considered 100,000 training patches and 30,000 validation patches. The test patches are extracted from the test image as described in the following Section. For instance in the *Saint Gall* dataset the number of patches for each pages are around 600,000 (the number is not fixed, since the patches are extracted at random from background areas).

4.3 Textline Identification

The textline identification is made by extracting dense patches on the test images. Using a sliding window with the estimated dimensions, we classify the patches. The model used gives us an estimate of the probability for each class. This information can be used to infer the position of the lines in the image. We move the sliding window on the image with a stride size equal to 10% of the patch width and 1% of the patch height. In this way, we are able to compute several scores for the same position. We then compute the average probability along the horizontal direction with respect to the patch dimension. This approach is depicted in Fig. 8 where we describe the test procedure. In particular, we can see how we extract dense patches form the input image and also how we compute the probability score for the whole page.

4.4 Preliminary Results

We present in the following the preliminary results obtained for the two datasets.

In the Ancestry dataset, the ground truth defines only the text line separators, therefore we can consider only two classes: *text separator* and *no text*

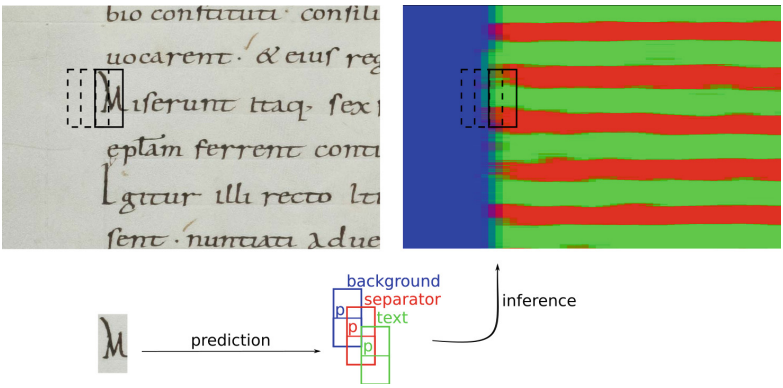
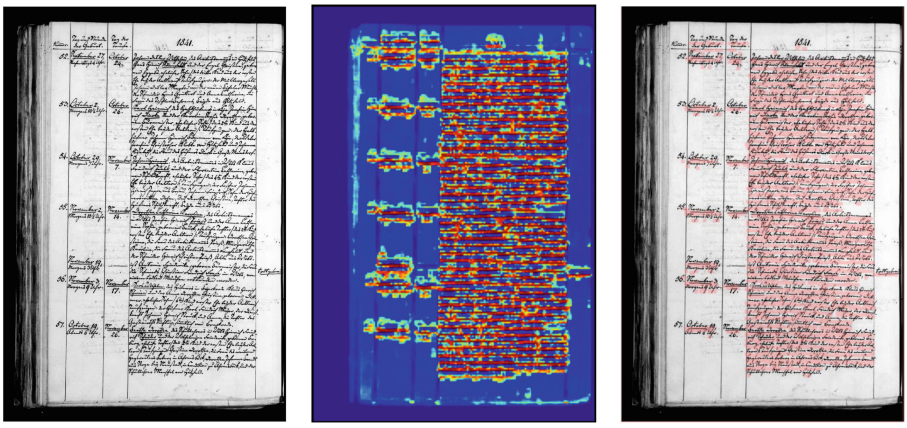


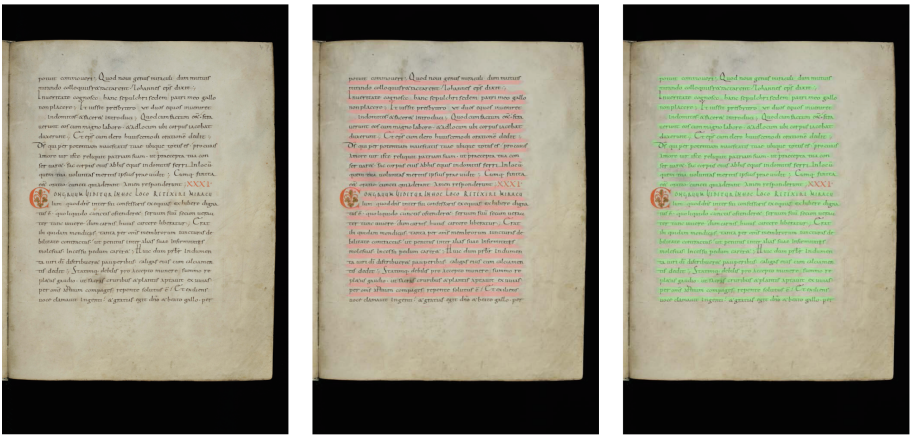
Fig. 8. Textline identification: for each patch one prediction is computed and results are combined to segment the page into three classes: background in blue, text line in green and separator in red. (Color figure online)

separator (background or text). The architecture used for this task is the same presented before. However, in the last *mlpconv* we have one output with only two feature maps. After the test step we can compute the results as we can see in Fig. 9. In the figure we present the results of the model considering only the probability value after the softmax function on the text line separator concept. The testing phase is made moving a window over the input image, therefore we have a probability score for each position. We are able to define the text areas computing the average probability score as we can see in Fig. 9b. This probability



(a) input (b) probability map (c) results over input image

Fig. 9. Probability Map to localize text line separators. In (a) the input image, in (b) the probability map computed by our framework and in (c) the results after a threshold over the input image.



(a) input image (b) results for separator lines (c) results for text lines

Fig. 10. Results obtained on the dataset with three classes. In (a) the input image, in (b) the results for separator lines and in (c) the results for text lines.

value represents the text line separator presence. The final result will be a probability map where we can discover the separator position. In this case, having only two classes, we want to discriminate better the result and using a threshold on this probability score, we can predict the text line separator as shown in Fig. 9c. In the figure, where we used a threshold of 0.7 to define the text line separators, we can see the probability map with the corresponding prediction results.

In the HisDoc project, we have more detailed ground truth where we can define median text lines and separator text lines. Using this information, we can use the proposed solution to predict three classes: text, separator and background. With this information, we can infer the text and separator line positions. An example of this approach is shown in Fig. 10 where we can see two different results, one for each class. In particular, in Fig. 10b we present the area for the separator class, while in Fig. 10c we present the area for the text line class.

5 Conclusions

In this paper we presented our preliminary work on the use of Convolutional Neural Networks to perform text line segmentation in historical documents. With these preliminary results we obtained a good starting point to extract the text lines in two different collections containing manuscript records. These results have been obtained considering a training made on small datasets and relatively homogeneous data. In future experiments, we will check the model with more data and we will investigate about possible changes in the model to improve the results.

Acknowledgments. This research is partially supported by a research grant from Ancestry.

References

1. Likforman-Sulem, L., Zahour, A., Taconet, B.: Text line segmentation of historical documents: a survey. Int. J. Doc. Anal. Recognit. **9**(2), 123–138 (2007)
2. Pastor-Pellicer, J., Afzal, M. Z., Liwicki, M., Castro-Bleda, M.J.: Complete system for text line extraction using convolutional neural networks and watershed transform. In: 12th IAPR Workshop on Document Analysis Systems, DAS 2016, Santorini, Greece, 11–14 April 2016, pp. 30–35 (2016)
3. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, pp. 2278–2324 (1998)
4. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR 2015, pp. 3431–3440 (2015)
5. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Integrated recognition, localization and detection using convolutional networks, Overfeat (2014)
6. Afzal, M.Z., Capobianco, S., Malik, M.I., Marinai, S., Breuel, T.M., Dengel, A., Liwicki, M.: DeepDocClassifier: document classification with deep convolutional neural network. In: 13th International Conference on Document Analysis and Recognition, ICDAR 2015, Nancy, France, 23–26 August 2015, pp. 1111–1115 (2015)
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). http://www.deeplearningbook.org
8. Lin, M., Chen, Q., Yan, S.: Network in network. CoRR, abs/1312.4400 (2013)
9. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: Gordon, G.J., Dunson, D.B. (eds.) Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2011), vol. 15, pp. 315–323 (2011). Journal of Machine Learning Research - Workshop and Conference Proceedings
10. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR, abs/1207.0580 (2012)
11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. CoRR, abs/1412.6980 (2014)
12. Fischer, A., Bunke, H., Naji, N., Savoy, J., Baechler, M., Ingold, R.: The HisDoc project. Automatic analysis, recognition, and retrieval of handwritten historical documents for digital libraries, vol. 38, pp. 91–106. De Gruyter (2014)
13. Capobianco, S., Marinai, S.: Record counting in historical handwritten documents with convolutional neural networks. CoRR, abs/1610.07393 (2016)