International Journal on Document Analysis and Recognition (IJDAR) (2018) 21:177–186
https://doi.org/10.1007/s10032-018-0304-3

SPECIAL ISSUE PAPER

Fully convolutional network with dilated convolutions for handwritten text line segmentation

Guillaume Renton[1] · Yann Soullard[1] · Clément Chatelain[1] · Sébastien Adam[1] · Christopher Kermorvant[1,2] · Thierry Paquet[1]

Received: 29 September 2017 / Revised: 16 May 2018 / Accepted: 17 May 2018 / Published online: 30 May 2018

Abstract
We present a learning-based method for handwritten text line segmentation in document images. Our approach relies on a variant of deep fully convolutional networks (FCNs) with dilated convolutions. Dilated convolutions allow to never reduce the input resolution and produce a pixel-level labeling. The FCN is trained to identify X-height labeling as text line representation, which has many advantages for text recognition. We show that our approach outperforms the most popular variants of FCN, based on deconvolution or unpooling layers, on a public dataset. We also provide results investigating various settings, and we conclude with a comparison of our model with recent approaches defined as part of the cBAD (https://scriptnet.iit.demokritos.gr/competitions/5/) international competition, leading us to a 91.3% F-measure.

1 Introduction

Text line detection is a central step of document layout analysis since it is commonly used in text recognition [22], as well as in higher-level processing such as document categorization [21]. It is well known that text line segmentation has a very strong impact on recognition performance. In the case of printed documents, this task is pretty trivial, even if some difficulties occur depending on the kind of documents (e.g., scan quality, background color, vertical lines). However, in the case of handwritten documents, overlapping between unstraight lines, irregularities of handwritten words and characters, and intrinsic high variabilities of handwriting make the text line detection much more difficult (see Fig. 1). Those difficulties may be increased when the document quality is low, which is often the case with historical documents, for example.

This work has been supported by the French National grant ANR 16-LCV2-0004-01 Labcom INKS. This work is founded by the French region Normandy and the European Union. Europe acts in Normandy with the European Regional Development Fund (ERDF).

✉ Guillaume Renton
guillaume.renton@gmail.com

1 Normandie Univ, UNIROUEN, UNIHAVRE, INSA Rouen, LITIS, 76000 Rouen, France

2 TEKLIA SAS, Paris, France

Another issue with text line segmentation comes with the definition of what is a text line. One can find various definitions in the literature, as shown in Fig. 2. A text line can be defined either as a baseline which corresponds to the basis of the text line [8], as a bounding box [17], or simply as a set of text pixels (i.e., the writing components) [32]. The last definition of a text line that can be found in the literature relies on X-height [32], which corresponds to the area between the baseline and the X-line. In other words, this is the area that covers the core of the text, without its ascenders and descenders (see Fig. 3).

Defining text line through their X-height brings many advantages over other representations. First, the X-height well depicts spaces between lines, even when lines overlap due to ascenders or descenders, in contrast to a bounding box representation which is unable to separate overlapping lines. Second, X-height representation seems suitable to easily get inputs for text recognizers. Indeed, it provides an image per line, in opposition to a text-level labeling that provides numerous connected components for a single line. Thus, dealing with a text-level labeling requires a post-processing before being fed to a text recognizer. Finally, X-height representation contains more information than the baseline since it is easy to get a baseline from a X-height labeling (as it is the lowest boundary of the core text), while the opposite is impossible. For all these reasons, we have retained the X-height representation.

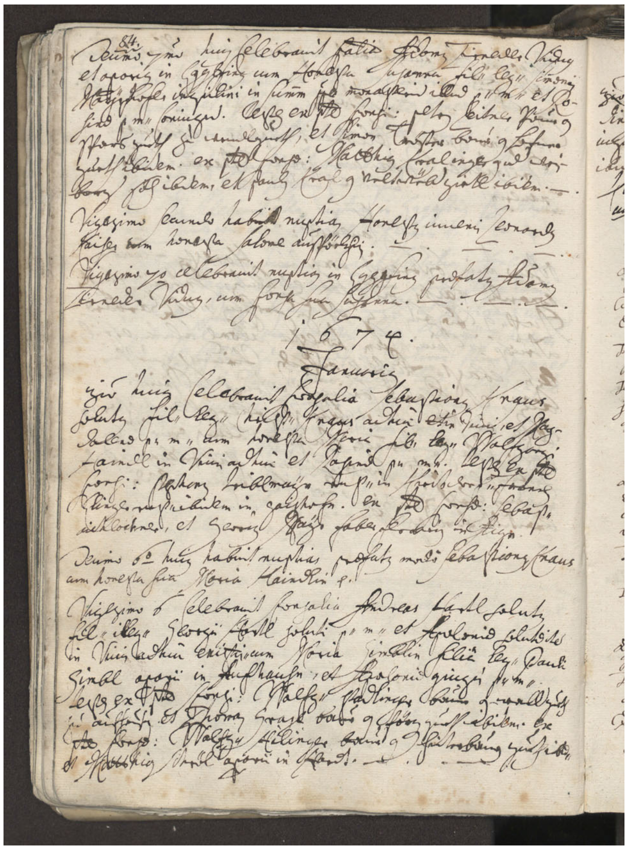
[illegible]

Fig. 1 Example of a historical document with unstraight lines that overlap, and irregularities of handwritten characters. Image extracted from the cBad competition [8]

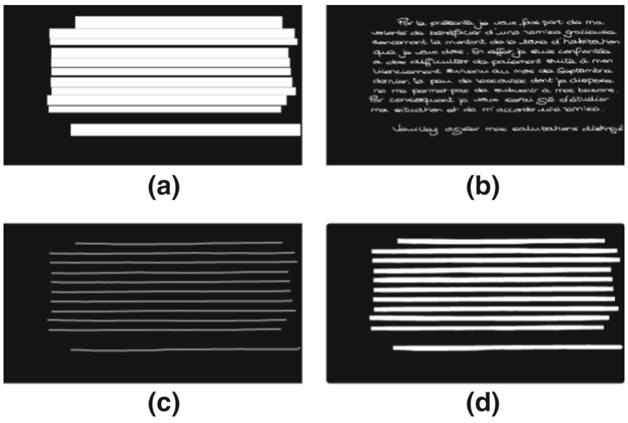

Fig. 2 **a** Bounding box labeling. **b** Text-level labeling. **c** Baseline labeling. **d** X-height labeling

Whatever the text line definition, there are two main types of methods to extract text lines. On the one hand, ad hoc methods rely on dedicated processing sequence such as filtering, projection profiles, mathematical morphology, and clustering. On the other hand, learning-based methods become more and more popular for text line segmentation, especially with the growth of deep learning methods.

In this paper, we present a new learning-based text line segmentation approach based on deep learning, applied on a X-height labeling. The proposed approach is an original variant of fully convolutional networks (FCNs) that have been recently investigated with success for semantic segmentation on natural scene images [4,14,23]. One of the main issues of FCN approaches is the way used to get an output with the same dimensions as the input. This is generally done using a deconvolution or unpooling process. We propose to circumvent such processes using dilated convolutions. In this work, we present first an in-depth study of our proposal which allows us to improve previous results for the cBAD competition and second a comparison of the main FCN architectures, including our proposal, which emphasizes the relevance of FCN based on dilated convolutions compared to traditional FCN based on deconvolution or unpooling layers.[1] We show that our method provides interesting text line segmentation results on real-world handwritten documents.

This paper is structured as follows: related works are presented in Sect. 2. Section 3 introduces the principles of fully convolutional networks. Our approach is described in Sect. 4, and Sect. 5 presents our experiments.

2 Related works

Text line segmentation methods can be divided in two groups: ad hoc methods and learning-based methods.

2.1 Ad hoc methods

Currently, ad hoc methods which are not based on training are the most used, as shown in the recent and very complete survey [5]. Among the large number of existing methods, we decided to present those who reported good results in competitions, especially in [19,30].

In [28], the authors use filters which can be rotated to detect text lines, and apply heuristic post-processes to separate connected lines. This top-down methods have shown good results on the International Conference on Document Analysis and Recognition (ICDAR) 2015 competition on text line detection [19].

Another method which achieved good results in text line detection is the bottom-up method described in [27]. The approach is based on superpixels to get connected components. The authors define a cost function to aggregate superpixels into a text line. This method won both the ICDAR 2013 Competition for handwriting segmentation [30] and the ICDAR 2015 competition on text line detection [19].

[1] Please note that a preliminary work has been presented at the ICDAR-WML workshop [25].

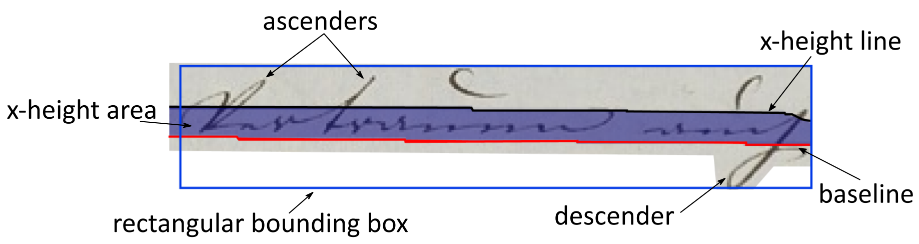

Fig. 3 A diagram showing terms used in text line definitions, from a text line segmentation using an FCN

Even though they achieved good results in international competitions, those methods have to be fine-tuned by hand, which is a tedious task and is generally dataset dependent.

2.2 Learning-based methods for text line segmentation

While deep learning approaches [12] have obtained great results in many application fields, very few works have investigated their use for text line detection. The main contributions have been presented by Moysset et al. [15–18]. The authors propose the use of a multi-dimensional long short-term memory (MDLSTM) neural network combined with convolutional layers to predict a bounding box around a line. Those methods obtained very good results, but they are limited to horizontal lines. Moreover, such types of networks are difficult to train and require large annotated datasets.

Although deep learning approaches are pretty uncommon in text line segmentation, they have been explored in related domains such as object detection and scene text detection. The next section reviews some works in those related domains.

2.3 Learning-based methods in related domains

In a scene text detection task, first works based on deep learning approaches use a sliding window method, by first extracting parts of the image using a sliding window process and then labeling them using a deep neural network as in [36]. However, using a sliding window process highly increases the processing time and it limits the context which can be used to take the decision. To limit the processing time, one solution is to use a preprocess to extract candidates and then take a decision for each of those candidates. This is the method used by [10], which extracts candidates using the Maximally Stable Extremal Regions (MSER) method and classifies them with a convolutional neural network.

The idea of extracting candidates before classifying them was also used in object detection, especially in different works of Girshick et al. [6,7]. In those works, the authors propose a Region-based Convolutional Network method based on a selective search method to extract candidates. In [6], they greatly increase the speed of such type of algorithms.

Still in object detection, recent algorithms such as those presented in [13,24] analyze the input images using a regular grid and take a decision for each tile of the grid. Those tiles are then gathered to take a global decision.

Finally, fully convolutional networks [14] (FCNs) have been recently defined and applied with success in semantic segmentation [14,23,35]. In [34], the authors apply FCNs for scene text detection. First, a Text-Block FCN is used to detect coarse localizations of text lines which are then extracted by taking into account local information using MSER components. Finally, another FCN is applied to reject false text line candidates. For text line segmentation [32], FCNs are used to detect text lines in which text components are then extracted.

In the next section, we present the fully convolutional networks and discuss their advantages compared to standard convolutional neural networks.

3 Fully convolutional networks

A fully convolutional network is a convolutional neural network (CNN) without dense layers. This characteristic brings multiple advantages. First, removing dense layers allows to work with variable input sizes, as convolutional layers do not require a fixed number of input. Second, in standard convolutional networks, dense layers contain a very large number of parameters. Thus, avoiding dense layers highly reduces the number of parameters. For example, in the well-known VGG16 [29] architecture, 120 million of the 138 million of parameters (87%) come from the dense layers, while there are only 3 dense layers against 13 convolutional layers. Third, FCNs are able to keep spatial information, in contrast to CNN where spatial information is aggregated into dense layers toward an output class (for classification) or an output value (for regression). Applied on images, FCN can therefore be used to produce a heatmap of the input image, containing a spatial description of the image. This advantage makes them really suitable for a semantic segmentation task.

A major issue when using an FCN relates to the way to rebuild an image from a lower resolution to the original one. Actually, using a convolutional neural network induces the use of pooling layers, which reduces the input resolution with the goal to increase the receptive fields without increasing the number of parameters. Thus, to have a pixel-level labeling of an input image (i.e., a heatmap of the same size as the input image), the network output resolution has to be increased. There are 3 methods that have been proposed for this task in

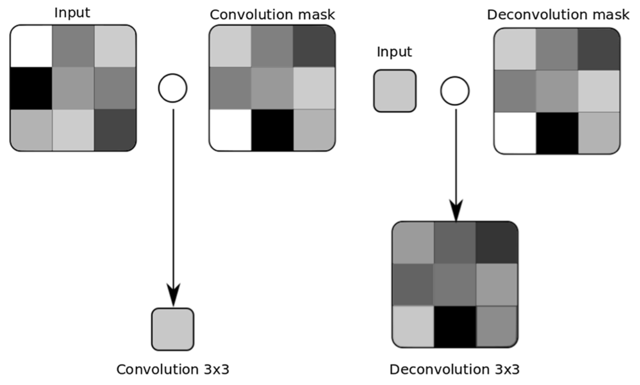

Fig. 4 Convolutional and deconvolutional layers. The deconvolution is performed using a convolutional layer applied with a stride equal to 1/3

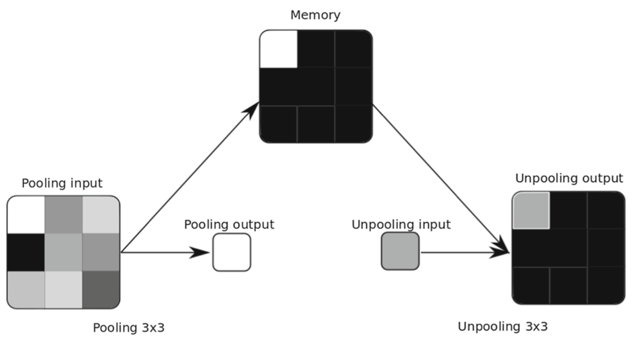

Fig. 5 Pooling and unpooling layers. For a pooling layer, winning positions are stored in memory and used for the related unpooling layer. Black cells relate to a zero value

the literature: deconvolution, unpooling and dilated convolutions.

3.1 Deconvolution

The deconvolution principle has been first used in convolutional networks by Long et al. [14] and then used in many works [23,32,35]. The idea is to create the inverse layer of a convolutional layer. For this, on the one way a deconvolution filter is applied with a stride equal to $\frac{1}{f}$, to up-sample the output by increasing the input f times with zeros and applying convolution on this sparse input or, on the other way, a filter is applied on a single pixel (Fig. 4).

These deconvolution filters have to be trained, making the network deeper. This particularity is both an advantage and a drawback since deepening the network makes it more expressive, at the expense of a heavier network that requests more data to be well trained.

3.2 Unpooling

While the deconvolution is the opposite of the convolution, the unpooling is the opposite of the pooling. The idea is to store the winning activation in the different pooling layers. Then, unpooling layers are applied in a symmetric way to pooling layers, and each unpooling layer is related to a pooling layer. Finally, to up-sample outputs, each pixel is set to the corresponding winning activation, while its neighborhood is set to 0 (Fig. 5).

Contrary to deconvolution, unpooling layers do not increase the number of parameters, but only the memory. However, in practice, as the losing activations are set to 0, the rebuilt image is sparse and lacks the information. Thus, unpooling layers are often combined with convolution layers, which increase the number of parameters.

Unpooling was used by Badrinarayanan et al. in [1] with convolution layers, while [20] used both unpooling and deconvolution.

3.3 Dilated convolutions

While deconvolution and unpooling allow to generate an image with a higher resolution than its input, a dilated convolution never reduces the original resolution, i.e., the one of the images given as input of the network.

In standard convolutional networks, there are two ways to reduce the resolution: (i) using a stride higher than 1 in a convolution layer and (ii) using pooling layers. But it is also possible to keep the same resolution after a convolutional layer by applying a stride equal to 1, with padding to solve the border effects. However, avoiding pooling layers is problematic, since they are used to increase the filter's receptive field and thus the context which is considered within the successive convolution layers.

To solve this problem, a solution consists in increasing the filter size, but it leads to strongly increase the number of parameters, as the number of parameters in the network is the square of the filter size. For example, a 9×9 receptive field requires 81 parameters, against only 9 parameters for a 3×3 receptive field coupled with a 3×3 pooling layer (which would result in an equivalent 9×9 receptive field). Finally, to get the same receptive field than VGG16, the number of parameters will explode from 9 to 4225 for each filter.

Using a dilated convolution is one way to solve this problem. It is based on the "A trous" algorithm proposed by Holschneider et al. [9]. This algorithm has been firstly used with wavelet transform to fill filters with zeros and thus increase the size of the receptive fields without increasing the number of parameters.

Let x be the input of the convolutional layer (i.e., the output of the previous layer or the input image), x is of dimension $H \times W \times D^I$ where H, W and D^I relate to the height, width and the number of channels, respectively. Let f be the weighted filter (convolutional kernel) of size $H^f \times W^f \times D^I$. To preserve the input size in output, one considers a stride $s = 1$ and the input is padded by adding rows and columns with zeros ($\lfloor \frac{H^f-1}{2} \rfloor$ rows on the top, $\lfloor \frac{H^f}{2} \rfloor$ rows on the bottom, $\lfloor\frac{W^f-1}{2}\rfloor$ columns on the left and $\lfloor\frac{W^f}{2}\rfloor$ columns on the right). From an input x that has been padded, the output of a standard convolution is obtained using the following equation:

$$y\left[i,j,d^o\right]=\sum_{k=0}^{H^f}\sum_{l=0}^{W^f}\sum_{d=0}^{D^I} f\left[k,l,d,d^o\right]\times x[i+k,j+l,d] \tag{1}$$

where d^o relates to the channel index in output and $\lfloor\frac{H^f-1}{2}\rfloor \leqslant i \leqslant H$ and $\lfloor\frac{W^f-1}{2}\rfloor \leqslant j \leqslant W$.

Regarding dilated convolutions, one defines an additional term r referring to the dilated rate, i.e., the scale factor of the filter. By considering a convolutional kernel of size $H^f \times W^f \times D^I$ as above, the convolution is applied on windows of height $\tilde{H}^f = H^f + (H^f-1)\times(r-1)$ and width $\tilde{W}^f = W^f + (W^f-1)\times(r-1)$ in the image. Thus, an input image is padded by adding $\lfloor\frac{\tilde{H}^f-1}{2}\rfloor$ rows on the top, $\lfloor\frac{\tilde{H}^f}{2}\rfloor$ rows on the bottom, $\lfloor\frac{\tilde{W}^f-1}{2}\rfloor$ columns on the left and $\lfloor\frac{\tilde{W}^f}{2}\rfloor$ columns on the right. Similarly to Eq. 1, the output of a dilated convolution is:

$$y[i,j,d^o]=\sum_{k=0}^{H^f}\sum_{l=0}^{W^f}\sum_{d=0}^{D^I} f[k,l,d,d^o]\times x[i+r\times k, j+r\times l, d] \tag{2}$$

where one recalls that d^o relates to the channel index in output and that $\lfloor\frac{\tilde{H}^f-1}{2}\rfloor \leqslant i \leqslant H$ and $\lfloor\frac{\tilde{W}^f-1}{2}\rfloor \leqslant j \leqslant W$.

Dilated convolution has some similarities with convolutions performed at multiple scales as the receptive field size is changed between the layers. One can also see that a dilated convolution is a generalization of the standard convolution. The standard convolution is obtained for a dilation rate equal to 1. Dilated convolutions have been used in many works for semantic segmentation [2–4,33], showing interesting results. This may be explained by the advantage that a dilated convolution brings: the receptive fields can be adjusted easily, without reducing the resolution nor increasing the number of parameters, despite a higher number of computations due to a constant high resolution (equal to the resolution in input of the network). Figure 6 illustrates dilated convolutions.

4 Proposed approach

In this section, we present our method based on a fully convolutional network with dilated convolutions for a text line segmentation task. Here, text lines are defined by the X-height (i.e., the core text). We start by motivating our approach in 4.1, and then, we present our network architecture in Sect. 4.2.

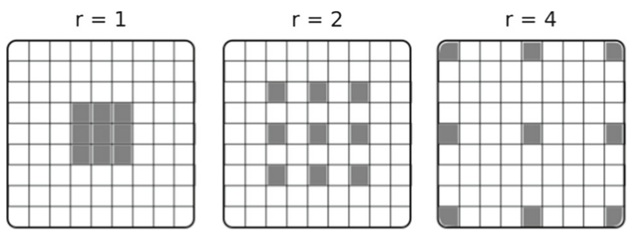

Fig. 6 Receptive field of dilated convolution for different dilation rate r

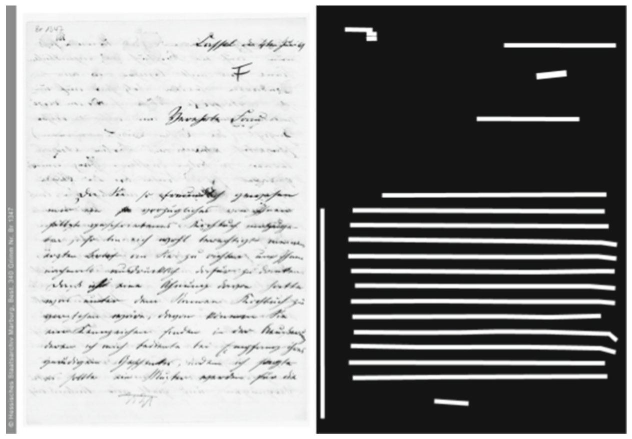

Fig. 7 Example of X-height labeling

4.1 Motivations

As shown in Sect. 1, the X-height labeling brings many advantages. First, it makes the separation between overlapping lines easier than a labeling using bounding box. Thus, a neural network is able to learn features representing these separations. A similar behavior happens with spaces between words, which have to be classified as a part of a text line and not as a blank. Another advantage of the X-height labeling comes from the class balancing. Indeed, if one considers the text line segmentation task as a semantic segmentation problem, each pixel has to be labeled as a text line or not. This produces a highly imbalanced problem, especially for text pixel and baselines labeling, and in such a case, a neural network tends to predict only the majority class. Thus, X-height and bounding boxes labeling seem more appropriate than the two others labeling, as the imbalance between the two classes is smaller. From those advantages, we focus on the X-height labeling. Figure 7 shows an example of original document and its X-height ground truth.

As text line segmentation can be seen as a semantic segmentation problem, we decided to use fully convolutional networks that provide good results for such a task. As discussed above, there are 3 types of FCN

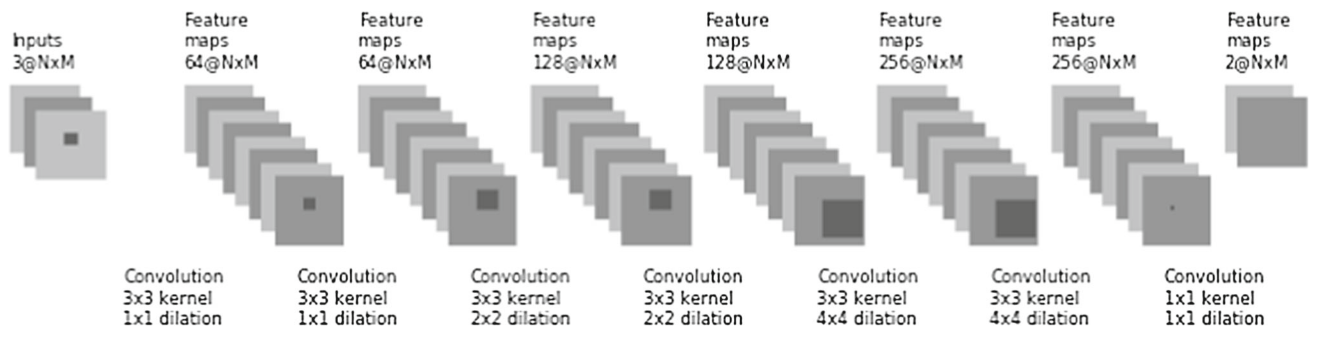
Inputs 3@NxM
Feature maps 64@NxM
Feature maps 64@NxM
Feature maps 128@NxM
Feature maps 128@NxM
Feature maps 256@NxM
Feature maps 256@NxM
Feature maps 2@NxM
Convolution 3x3 kernel 1x1 dilation
Convolution 3x3 kernel 1x1 dilation
Convolution 3x3 kernel 2x2 dilation
Convolution 3x3 kernel 2x2 dilation
Convolution 3x3 kernel 4x4 dilation
Convolution 3x3 kernel 4x4 dilation
Convolution 1x1 kernel 1x1 dilation

Fig. 8 Our network architecture: the input resolution is always the same and the receptive fields are increased due to the dilation

models: deconvolution-based, unpooling-based and dilated convolution-based models. In our opinion, the reconstruction part which is applied in deconvolution-based and unpooling-based FCN can be a problem. Indeed, for an application in text line segmentation, the reconstruction can sometimes be coarse. In semantic segmentation, coarse outputs can be adjusted by conditional random fields [11], which has been applied in many works [2,3,35]. However, CRFs can not be used here as they are based on pixel variations (so they can be applied only on a text-level labeling). In addition, coarse outputs lead to under-segmentations (i.e., merged lines), which is problematic for using them as input of a text recognition system. This is why we define an FCN based on dilated convolutions that are less subject to provide coarse outputs. Dilated FCNs have other several advantages as presented in Sect. 3.3 such as the fact that the number of parameters is not increased.

4.2 Network architecture

We investigate two network architectures as reference: one with 7 layers and one with 11 layers. The network architecture with 7 layers is presented in Fig. 8. The first two layers are standard convolutions with a dilation of 1, then two layers with a dilation of 2 and finally two layers with a dilation of 4. Dilation rates are used to replace pooling layers, in order to keep the same receptive fields than after a 2×2 pooling layer. The first 6 layers of VGG16 and the 6 first layers of our network use the same size and numbers of filters, while the only difference comes from the use of dilated convolutions instead of pooling. We made this choice since this architecture has proven to be an effective feature extractor. An output layer is added to get predictions, with a dilation of 1 and a filters size of 1. The idea behind these dilations is that text line detection does not require large context to be effective.

Regarding the 11 layers, the 6 first layers are the same as for the 7 layers architecture. Then, there is two convolutional layers with a dilation rate of 2 and two other ones with a dilation rate of 1. Finally, an output layer with a dilation of 1 and a filters size of 1 is added to get predictions. Such an architecture has some similarities with traditional FCN for which there is several deconvolution layers and unpooling layers to get a progressive reconstruction.

5 Experiments

In this section, we investigate the behavior of fully convolutional networks with dilated convolutions for a text line segmentation task. We begin by introducing our experimental setting, before evaluating the different types of FCN described in Sect. 3: FCN with deconvolution, unpooling or dilation. Then, we observe the influence of the number of layers and the variation of the acceptation threshold on the text line class. Finally, we evaluate our approach as participant of the international competition cBAD.[2]

5.1 Experimental setting

We experiment our approach on the dataset provided for the cBAD competition held in the International Conference on Document Analysis and Recognition (ICDAR 2017) and focused on baseline detection. This dataset is made of 216 archival documents images for training and 539 archival documents images for test. Those images are provided from 7 different archives. Since no validation set is provided, we separated the 216 first documents in 2 parts: 176 are used in training, while the 40 remaining are used for validation.

As we process images of variable size, working with high-resolution images may exceed the size of the GPU memory. Therefore, images that do not fit the GPU memory are reduced to a smaller resolution. In our experiments, the maximum size of the largest side has been set to 608 pixels, the ratio between height and width being kept.

The goal of this competition consists in detecting baselines, whereas our approach predicts X-height area. However, both baselines and X-height area are given in the ground truth. Thus, our approach is trained using the X-height labeling as

[2] https://scriptnet.iit.demokritos.gr/competitions/5/.

text line representation, and we extract the related baselines as the lower bound of X-height areas for evaluation.

5.2 Metrics and evaluation

To evaluate the different methods, we refer to the metrics of the cBAD competition [8]. Three metrics are defined to evaluate the detected text lines: the precision, the recall and the F-measure, computed from the predicted baselines.

To compute those metrics, the organizers first define a coverage rate between a hypothesis baseline and a ground truth baseline. It consists in discretizing both ground truth and hypothesis baselines and matching every point of each hypothesis baseline with a point of the ground truth baselines. Then, a distance cost is computed depending to the gap between the pairs of points.

The recall is then directly computed from the coverage function, by dividing the sum of the coverage rate for each baseline by the number of ground truth lines.

Regarding the precision, an alignment function is defined to match ground truth baselines with hypothesis baselines. This allows to extract a set of baseline pairs that matches. From that, the coverage rate of each couple is computed and then divided by the number of hypothesis lines to get the precision.

Finally, the F-measure is computed in a standard way as the harmonic average of precision and recall.

5.3 Comparison of FCN using different image rebuild strategies

In this work, we experiment the three different FCNs described in Sect. 3 on the cBAD competition data set. Thus, we trained an FCN based on deconvolution, an FCN based on unpooling and an FCN based on dilated convolutions. We also compare these approaches with a network combining deconvolution and unpooling layers, as the one presented in [20].

To keep a fair comparison, we managed to use similar size of receptive fields and filter numbers. Thus, we used 7 or 11 layers for each network as presented in Sect. 4.2. For the network architecture with 7 layers, we used the dilated-based network architecture of Fig. 8, inspired from the first convolutional layers of VGG-16. In the deconvolution and unpooling-based networks, pooling layers are added after the convolution layer 2 and 4. To perform the deconvolution, a deconvolution layer is used on the last layer with a stride of 4 to up-sample the output. For the unpooling network, an unpooling layer with a rate of 4 is used before a convolutional layer at the end of the network. The network combining deconvolution and unpooling network is composed of one deconvolutional layer and one unpooling layer.

Table 1 Results obtained by fully convolutional networks using four strategies: deconvolution, unpooling, deconvolution and unpooling, and dilated convolutions

Architecture		Evaluation metrics		
Method	Layers	Precision	Recall	F-measure
Deconvolution	7	62.2	76.5	68.6
	11	**72.7**	83.9	77.9
Unpooling	7	56.6	74.7	64.4
	11	72.5	83.4	77.6
Deconv + Unpool	7	71.3	47.2	56.8
	11	72.1	84.0	77.6
Dilated	7	70.8	82.3	76.1
	11	72.2	**85.5**	**78.3**

Best results are shown in bold

The four resulting networks are pretty similar, and only the last layers differ.

As deconvolution and unpooling-based networks generally have several deconvolutional or unpooling layers, we also evaluate such architectures. For that, we define networks composed of 11 layers with 2 deconvolutional layers or 2 unpooling layers. The network combining deconvolution and unpooling network contains 2 unpooling layers and 2 deconvolutional layers with a stride of 1. For the dilated convolutional network, we apply 4 additional convolutional layers (for layers 7–10) with a dilation rate of 2 for the two first layers and a dilation rate of 1 for the two next ones.

Each network is trained on the cBAD training set until the validation set converges. The best network on the validation set is then selected, and results on the test dataset are then submitted. Raw results (without post-processing) are presented in Table 1.

One can observe that, both for 7 and 11 layers, the dilated convolution networks generally outperform deconvolution and unpooling networks. Besides, dilated convolutions also produce a slightly lighter network than a deconvolution one, since the deconvolution layer requires more parameters. For instance, in the case of 7 layers, unpooling and dilated convolution networks use about 1,145,922 parameters, while the deconvolution one uses about 1,211,714 parameters. In the case of 11 layers, we have 1,698,882 for the dilated architecture, while deconvolution architecture uses 1,871,426 parameters. However, increasing the number of layers strongly increases the number of computations as the size of the network input is kept, leading to a slower computation time. Based on this comment and on the fact that we have good results with 7 layers, we keep the network architecture with 7 layers for the next experiments.

5.4 Tuning the network architecture

In this part, we discuss the influence of the network architecture. As the size of the network dynamically changes from

Table 2 Results of an FCN based on dilated convolution for 5, 7 and 9 layers

Architecture	Precision	Recall	F-measure
5 layers	26.81	16.99	20.8
7 layers	70.77	**82.33**	**76.11**
9 layers	**76.14**	74.82	75.47

Best results are shown in bold

an image to another, the parameters of an FCN with dilated convolutions are only the number of layers, the number of feature maps and the dilation rate of each layer. As shown in Sect. 4.2, our network architecture is based on the first convolutional layer of the famous VGG16 convolutional network [29]. Thus, we decided to explore at what point increasing or decreasing the number of layers (and the dilation rate) in our network could improve or deteriorate our results. For this, we decided to evaluate 3 network architectures: an architecture of 5 layers where only a dilation rate of 1 and 2 is applied, an architecture of 7 layers (Fig. 8), and an architecture of 9 layers where the two last dilated convolutions have a dilation rate of 8. Table 2 shows those results.

As one can see, reducing the number of layers is really troublesome, since the system provides very poor results. Moreover, the maximum size of the receptive fields for the 5 layers architecture is too low: this network is unable to take enough context to take a correct decision. On the other hand, the 9 layers architecture has receptive fields with a correct size. But this architecture requires more parameters. Thus, the 9 layers architecture has 2,326,082 parameters, while the 7 has 1,145,922 and the 5 has 260,418. Those numbers are pretty low compared to the millions VGG16 has, for example, but the gap between the 7 and the 9 layers is high. In addition, due to the few training samples that we use, the 9 layers architecture tends to overfit and provides a lower F-measure than the 7 layers architecture.

Finally, we decided to retain the 7 layers architecture, which is a good compromise between the 5 layers architecture which lacks of receptive fields, and the 9 layers architecture which overfits.

5.5 Effect of pre-training

It is known that pre-training a network both increases convergence speed and model ability to get a better generalization. To perform a pre-training, we have selected additional data coming from the READ competition.[3] which contains 10,000 document images with text paragraph regions. This dataset does not provide the X-height areas, but only the text regions that generally contain several text lines.

[3] https://scriptnet.iit.demokritos.gr/competitions/8/.

Table 3 Effect of pre-training on performances

Training	Precision	Recall	F-measure
Without Pre-Training	70.77	82.33	76.11
With Pre-Training	84.95	87.59	86.25

In order to produce the X-height labeling that one needs to train our network, one has to first segment text regions into text lines, then to match the extracted lines with the ground truth in order to remove undesired extracted lines and finally to get the X-height labeling on the lines that have been kept. First, lines are extracted from text regions using steerable filters, a handcrafted line segmentation method providing moderate results. Once extracted, a text recognition is performed using the method described in [31]. The predicted character sequences are then aligned with the text lines of the ground truth using a dynamic programming algorithm. It consists in computing edit distances between the predicted lines and the ground truth and then matching them using a dynamic time programing like algorithm (which does not enable that a text line matches with more than one another sequence). Each extracted line for which the prediction matches with a text line from the ground truth is added to the training dataset, while the X-height area related to the line comes from the mask built in the steerable filters method.

These additional documents (8000 for training, 2000 for validation) have been used to pre-train the FCN in a transfer learning framework. Table 3 shows the effect of the pre-training over the results of our network. We observe a significant improvement on the test dataset, confirming the effectiveness of transfer learning on computer vision tasks.

5.6 Effect of rejection threshold

The FCN has been trained for a binary classification task (text line or background). Therefore, the FCN produces in output a probability matrix that each pixel belong to a X-height area, also called heatmap. This heatmap has to be thresholded in order to provide the X-height areas.

By default, the network selects the highest probability between the text line output and the background output, equivalent to a threshold of 0.5. Here we investigate different values of the decision threshold and show their effect on the network performance. Figure 9 compares the output for different thresholds values applied on the original prediction.

Results are presented in Fig. 10. As expected, varying the threshold significantly modifies the proportion of pixels labeled as text lines, thus impacting the recall and the precision of the network. One can observe that the highest F-measure value is obtained for a threshold of 0.45.

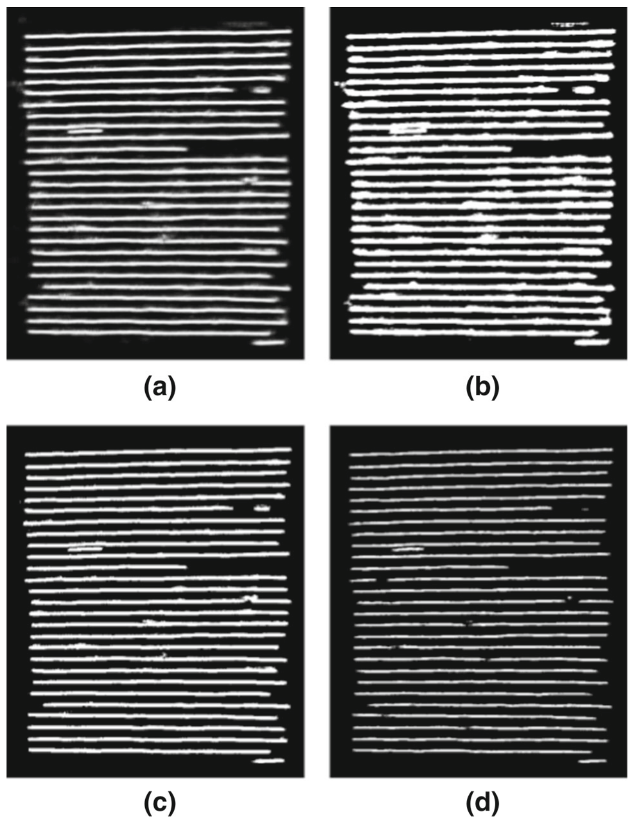

Fig. 9 **a** Network output. **b** 0.1 threshold. **c** 0.5 threshold. **d** 0.9 threshold

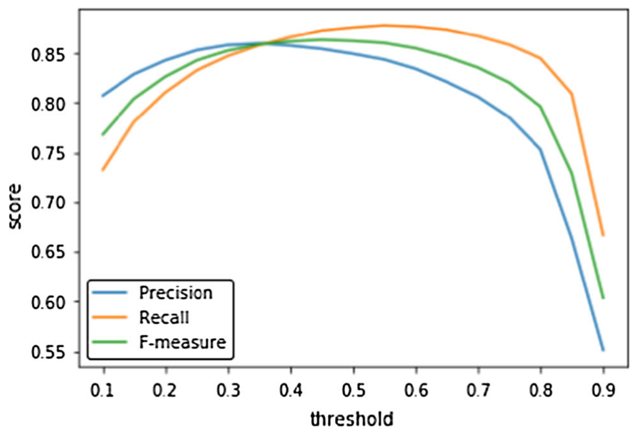

Fig. 10 Evolution of precision, recall and F-measure depending on the reject threshold (i.e., the minimum value for a pixel to be considered part of an x-height region)

5.7 cBad competition

For cBAD competition, we present results for 2 architectures: one with 7 layers and one with 11 layers, as presented in Sect. 4.

Our models have been pre-trained on the READ and cBAD datasets, with an optimized rejection threshold. We also added a simple post-processing to merge baselines that are potentially oversegmented. This post-processing consists in computing the average distance between the points representing a baseline and each regression line calculated on the points representing another baseline. Two lines are merged when the average gap is under a fixed threshold. This leads to our currently best model for both architectures.

Table 4 Comparison of our FCN methods compared to the main submitted systems

Method	Precision	Recall	F-measure
DMRZ	97.3	97.0	97.1
This work (11 layers)	94.9	88.1	91.3
This work (7 layers)	89.7	89.9	89.8
UPVLC	93.7	85.5	89.4
BYU	87.8	90.7	89.2
IRISA	88.3	87.7	88.0

We now discuss the results obtained during the cBAD competition and compare our approach with state-of-the-art methods (see Table 4). The proposed approaches based on FCN with dilated convolution provide the second best performances after the DMRZ system. Note that the DMRZ system adapted their post-treatment for each of the 7 archives, whereas post-treatment on our system is really light.

Up to date, deep learning approaches have been rarely used in text line segmentation, but there is currently an increased interest in these kinds of methods. Thus, both DMRZ and BYU use deep learning-based approaches. BYU even use a 10 layers fully convolutional network with deconvolution layers, while DMRZ uses a U-net [26]. Besides, IRISA uses an approach based on a blurred image combined with a description of text lines, while UPVLC approach is based on clustering over a set of interest points. Thus, our method follows the dynamic of deep learning-based approaches with a new method based on a convolutional network.

6 Conclusion

In this paper, a novel approach based on a variant of deep fully convolutional network (FCN) with dilated convolutions was presented for handwritten text line segmentation. Fully Convolutional Networks do not include dense layers, which brings numerous advantages as reducing the number of parameters, allowing to work with variable input sizes and keeping spatial information. The dilated convolutions keep the resolution of the input image, and there is no need to reconstruct the image as in an FCN with deconvolution or unpooling layers. In addition, our model is trained to identify X-height labeling which provides us a suitable text line representation, while limiting under- and oversegmentations.

We show that our model can outperform the most popular variants of FCN, based on deconvolution or unpooling layers. We also compare our system to recent approaches designed as part of the cBAD competition of the International Conference on Document Analysis and Recognition.

We believe that this approach can benefit from recent advances in deep learning to be improved such as a more intensive use of transfer learning, or other training tricks such as dropout or batch normalization. Another interesting perspective to this work is its extension to handle multi-resolution document images, which could be effectively achieved by exploiting dilated convolution with several ratio within the same training.

References

1. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder–decoder architecture for image segmentation (2015). arXiv:1511.00561
2. Chen, L., Papandreou, V., Kokkinos, I., Murphy, K., Yuille, A.: Semantic image segmentation with deep convolutional nets and fully connected crfs (2014). arXiv:1412.7062
3. Chen, LC., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs (2016). arXiv:1606.00915
4. Chen, LC., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation (2017). arXiv:1706.05587
5. Eskenazi, S., Gomez-Krämer, P., Ogier, J.M.: A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognit. **64**, 1–14 (2017)
6. Girshick, R.: Fast r-cnn. In: ICCV, pp. 1440–1448 (2015)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR, pp. 580–587 (2014)
8. Grüning, T., Labahn, R., Diem, M., Kleber, F., Fiel, S.: Read-bad: a new dataset and evaluation scheme for baseline detection in archival documents (2017). arXiv:1705.03311
9. Holschneider, M., Kronland-Martinet, R., Morlet, J., Tchamitchian, P.: A real-time algorithm for signal analysis with the help of the wavelet transform. In: Wavelets, pp. 286–297. Springer (1989)
10. Huang, W., Qiao, Y., Tang, X.: Robust scene text detection with convolution neural network induced mser trees. In: ECCV, pp. 497–511 (2014)
11. Krähenbühl, P.: Koltun, V.: Efficient inference in fully connected CRFs with gaussian edge potentials. In: NIPS, pp. 109–117 (2011)
12. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., Berg, A.: Ssd: Single shot multibox detector. In: ECCV, pp. 21–37. Springer (2016)
14. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR, pp. 3431–3440 (2015)
15. Moysset, B., Adam, P., Wolf, C., Louradour, J.: Space displacement localization neural networks to locate origin points of handwritten text lines in historical documents. In: Workshop on Historical Document Imaging and Processing, August (2015)
16. Moysset, B., Kermorvant, C., Wolf, C.: Full-page text recognition: learning where to start and when to stop. In: ICDAR (2017)
17. Moysset, B., Kermorvant, C., Wolf, C., Louradour, J.: Paragraph text segmentation into lines with recurrent neural networks. In: ICDAR, pp. 456–460 (2015)
18. Moysset, B., Louradour, J., Kermorvant, C., Wolf, C.: Learning text-line localization with shared and local regression neural networks. In: ICFHR (2016)
19. Murdock, M., Reid, S., Hamilton, B., Reese, J.: Icdar 2015 competition on text line detection in historical documents. In: ICDAR, pp, 1171–1175 (2015)
20. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV, pp. 1520–1528 (2015)
21. Paquet, T., Heutte, L., Koch, G., Chatelain, C.: A categorization system for handwritten documents. IJDAR **15**(4), 315–330 (2012)
22. Parvez, M.T., Mahmoud, S.A.: Offline arabic handwritten text recognition: a survey. ACM Comput. Surv. (CSUR) **45**(2), 23 (2013)
23. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters—improve semantic segmentation by global convolutional network (2017). arXiv:1703.02719
24. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. CoRR, abs/1612.08242 (2016)
25. Renton, G., Chatelain, C., Adam, S., Kermorvant, C., Paquet, T.: Handwritten text line segmentation using fully convolutional network. In 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, vol. 5, pp. 5–9. IEEE (2017)
26. Ronneberger, O., Fischer, P., Brox, T.: U-net: convolutional networks for biomedical image segmentation. CoRR, abs/1505.04597 (2015)
27. Ryu, J., Koo, H.I., Cho, N.I.: Language-independent text-line extraction algorithm for handwritten documents. Signal Process. Lett. **21**(9), 1115–1119 (2014)
28. Shi, Z., Setlur, S., Govindaraju, V.: A steerable directional local profile technique for extraction of handwritten arabic text lines. In: ICDAR, pp. 176–180 (2009)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556 (2014)
30. Stamatopoulos, N., Gatos, B., Louloudis, G., Pal, U., Alaei, A.: Icdar 2013 handwriting segmentation contest. In: ICDAR, pp. 1402–1406 (2013)
31. Stuner, B., Chatelain, C., Paquet, T.: LV-ROVER: lexicon verified recognizer output voting error reduction. CoRR, abs/1707.07432 (2017)
32. Vo, Q.N., Lee, G.: Dense prediction for text line segmentation in handwritten document images. In: ICIP, pp. 3264–3268 (2016)
33. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions (2015). arXiv:1511.07122
34. Zhang, Z., Zhang, C., Shen, W., Yao, C., Liu, W., Bai, X.: Multi-oriented text detection with fully convolutional networks (2016). arXiv:1604.04018
35. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.: Conditional random fields as recurrent neural networks. In: ICCV, pp. 1529–1537 (2015)
36. Zhu, S., Zanibbi, R.: A text detection system for natural scenes with convolutional feature learning and cascaded classification. In: CVPR, pp. 625–632 (2016)