Combination of Two Fully Convolutional Neural Networks for Robust Binarization

Romain Karpinski and Abdel Belaïd(✉)

Université de Lorraine-CNRS-LORIA, Campus scientifique,
54500 Vandoeuvre-Lès-Nancy, France
{romain.karpinski,abdel.belaid}@loria.fr
http://read.loria.fr/

Abstract. To be able to process historical documents, it is often required to first binarize the image (background and foreground separation) before applying the processing itself. Historical documents are challenging to binarize because of the numerous degradations they suffer such as bleed-through, illuminations, background degradations or ink drops. We present in this paper a new approach to tackle this task by a combination of two neural networks. Recently, the DIBCO binarization competition has seen a growing interest in the use of supervised methods to binarize challenging images. Inspired by the winner of the DIBCO 17 competition, which uses a fully convolutional neural network (FCN), we propose a combination of two FCNs to obtain better performance. While the two FCNs have the same architecture, they are trained on different representations of the input image. The first one uses downscaled image to capture the global context and the object locations. The second one works on patches of native resolution to help defining precisely the boundaries of the characters by capturing the local context. The final prediction is obtained by combining the results of the two FCNs. We show in the experiments that this strategy provides better results and outperforms the winner of the DIBCO17 competition.

Keywords: Historical documents · Binarization · Fully convolutional neural network

1 Introduction

Image binarization is the task of transforming an input image, whether in grayscale or color, into the binary space (0 or 1 values). The binary values represent whether a pixel belongs to the background or to the foreground (e.g. text pixels for textual documents). This type of image processing technique arises from the need to preprocess the information to simplify the next processing steps. These next processing steps can be for example text line detection [16], word spotting [9] or writer identification [11]. By first performing a binarization, one can expect an improvement of the pipeline because the image is easier to process. However, the binary image must be of highest quality so the next step is

C. V. Jawahar et al. (Eds.): ACCV 2018, LNCS 11363, pp. 509–524, 2019.
https://doi.org/10.1007/978-3-030-20893-6_32

not penalized. For example, if all pixels from a text character, are considered by the binarization algorithm as background, then the binary image will not contains this character. As a result, the required information can be missing from the input image thus resulting very likely in an error.

Obtaining good results in the context of historical documents can be tough because of the numerous degradations they suffer from. Historical documents are often very challenging since they can contain ink drops, bleed-through, stains, various illumination, etc. Nowadays, a huge number of techniques based on machine learning arises to handle this kind of documents. Machine learning approaches often equal or outperform classical methods thanks to their generalization capabilities. In this case, the binarization task is seen has a pixel labelling task called semantic segmentation. Semantic segmentation is the task of assigning a class probability for each pixel in an image (i.e. probability to be a text pixel for instance). The winner *Ilin et al.*, of the last DIBCO17 competition [22] on binarization, is a machine learning based method. It uses a special architecture of fully convolutional neural networks (FCN) [15] called U-net [23] which provide a high performance pixel labeling with only few labeled data. In this paper, we demonstrate on the DIBCO17 dataset that we can further improve the performance on the binarization task by combining the prediction of two FCNs.

The paper is organized as follow: in Sect. 2 we present the related work on binarization and fully convolutional neural networks. In Sect. 3, the proposed approach is described, then in Sect. 4, experiments are performed to demonstrate the effectiveness of our approach. Finally, conclusions and future work are drawn in Sect. 5.

2 Related Work

2.1 Binarization

The binary version of an image is obtain by thresholding the color or grayscale image. Generally, there are mainly two different kind of thresholding algorithms: local and global thresholding. Global thresholding approaches compute a single threshold for the whole image. In the context of historical documents global thresholding approaches such as Otsu [20] often has a suboptimal performance compared to local approaches. This is due to the fact that degradations can occur locally in an image such as illuminations, shadows or geometric deformations. On the other hand, local thresholding approaches are capable of adapting the binarization based on the local information of the image. However, they are often slower than global thresholding algorithm because of their need to compute statistics for each pixels to gather context from neighboring pixels. Local thresholding is usually done by using a small part of the image where statistics are computed to find the right threshold such as in Niblack et al. [18], Sauvola et al. [24] and Wolf et al. [27]. Other approaches such as Almeida et al. [4] employ bilateral filters on RGB images before applying the Otsu algorithm. Methods, such as Gatos et al. [8] and [28], use an estimation of the background and/or

foreground to perform the binarization. Recently, the trend in binarization algorithm is to use machine learning approaches to perform the binarization. All these approaches mainly use convolutional neural networks (CNN) [14] such as the work of Pastor-Pellicer et al. [21]. The work of Calvo et al. [6] uses convolutional auto encoders to filter out the background and only keep the text pixels. Recently, Afzal et al. [3] and Westphal et al. [26] showed that recurrent neural networks can also be used to binarize patches. Tensmeyer et al. [25] used a fully convolutional neural network to perform the binarization. The next subsection will provide more details on fully convolutional neural networks as it is used in our proposed method.

2.2 Fully Convolutional Neural Networks

Usually, convolutional neural networks are composed of a series of convolutional layers followed by fully connected layers at the end. The use of the fully connected layers forces the input to be of a fixed size. Fully convolutional neural network originates from Long et al. [15] where the fully connected layers are replaced by upsampling and convolutional layers. One big advantage of this architecture is that it can process images of arbitrary sizes by using only convolutional layers and has very less weights in comparison. Generally, FCNs can be divided in two parts: the encoder part and the decoder part. The encoder part takes the input image and compresses it into a dense representation where the number of feature maps is high and the size is reduced. The decoder part has to decode the dense representation and upsample it to retrieve the original size. Therefore, a fully convolutional neural network outputs an image of the same size as the input. Also, it allows to perform semantic segmentation on full images which was not possible due to the complexity of the fully connected layers. Semantic segmentation is used in various areas such as baseline detection [7] or road line detection for autonomous driving [5].

Our approach combines the state of the art U-net with the module Squeeze and Excite (module SE) from Hu et al. [12] as it has shown to improve the performance of residual networks. Inspired from Grüning et al. [10] we use their architecture, called ARU-net, which has an attention mechanism to capture the information at different scales before performing the final classification. The proposed method does not only perform the binarization at patch level but also at the full image level to take advantage of the differences between these two scales. Results from the DIBCO17 competition indicate that combining local and global thresholding has better performances than using only one of those. Therefore, we decided to combine the outputs of two FCNs: one for a labeling with local context by using patches and one for a labeling with global context by using full resized images.

Our contribution to this paper is as follows:

1. We propose a new binarization method that combines the strength to suppress the weaknesses of two networks that work on different scales to produce state of the art results.

2. We show that these two networks alone have lower performances compared to their combination.
3. The effect of horizontal flipping as a data augmentation technique is studied in the context of binarization with FCNs and its interest shown.

3 Proposed Approach

An overview of the proposed approach can be found in Fig. 1. The method has two branches where each contains a neural network, more precisely two fully convolutional neural networks. The upper branch works on image patches without any rescaling. Then, the image is reconstructed from the patches by using a sliding window where only the center of each patch is kept for the final prediction. The centers of the patches are extracted to avoid border effects since the pixels that are close to the patch borders do not have a complete context to perform the prediction. The lower branch makes predictions on the whole rescaled image and labels all its pixels in one step. Then, the prediction is upsampled to retrieve the original image size. Finally, predictions from both branches are thresholded using a global threshold and combined using the pixel wise logical AND operator. Identical neural networks are employed for the two branches. The architecture is described in the next subsection.

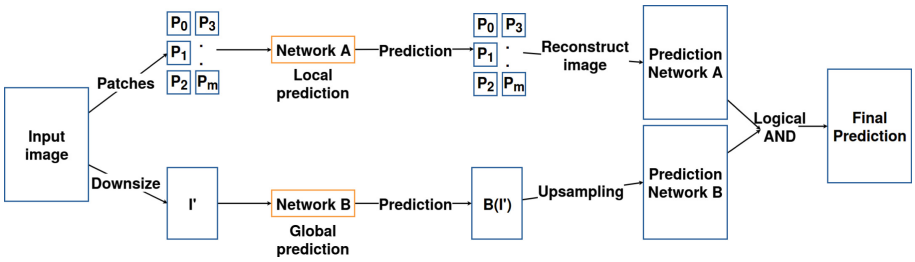


Fig. 1. Overview of the proposed approach.

3.1 The ARU-net

For the pixel labeling, we use a modified version of the ARU-net architecture of Grüning et al. [10] for both the local (patches) and global (full image) labeling. This architecture has been proven to provide better results on text baselines detection than a classical U-net with only a few addition of weights in the network. An overview of the ARU-net architecture can be seen in Fig. 2. It is composed of two neural networks: an Attention Network (AN) and a RU-net which is a U-net which has residual connections within its blocks. The U-net architecture will be described in Sect. 3.3 and the residual connections in 3.4. The ARU-net works by using the input image I plus n different scales S_n of

the input image. For simplicity in the notations, we state that S_0 is I. The S_n scales are the successive downsampling of I by a factor of f, using average pooling. Therefore, the image size $S_n^{size} = S_n^{width} \times S_n^{height}$ relatively to I^{width} and I^{height} of I is computed by Eq. 1. Similarly, S_n^{height} is computed using the same equation but with I^{height} instead of I^{width}.

$$S_n^{width} = I^{width} / f^n \tag{1}$$

where f^n denotes the scaling factor.

All input images S_n are passed to the RU-net and the AN. The RU-net computes the features maps F_n which will be weighted and used to perform the pixel classification. The AN outputs the attention feature maps A_n that will be used to weight the F_n feature maps. As we have downsampled the images for $n > 0$, we need to perform the inverse operation of upsampling. Therefore, all F_n and A_n feature maps are upsampled using the nearest neighbour method to match the size of S_0. The number of different input image scales is called $Scale_{depth}$. Once all feature maps and attention maps have been computed for all the scales S_n, the attention maps A_n are concatenated and softmaxed pixel-wise to distribute the weights along each scale giving AW_n feature maps. Then, all the feature maps F_n are weighted by their corresponding attention maps AW_n giving WF_n. Finally, all weighted feature maps WF_n are concatenated and a final convolutional layer performs the classification.

As said before, all S_n images are going through the same RU-net and AN. This makes the two networks more resilient to the scale variations that can occur between the different images. This is especially powerful when processing text images where the text size and annotations can have different shapes and scales.

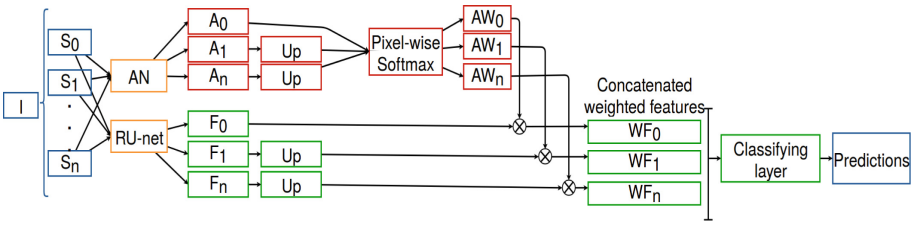


Fig. 2. ARU-net architecture overview. I is the input image. S_n is the scale of the image. AN is the attention network. RU-net is the network that extracts the features from the input image. A_n is the attention map. Boxes containing Up mean that the upsampling is done to match the size of S_0. AW_n is the softmaxed attention map. WF_n is the weighted features maps. Blue boxes are inputs, yellow are the neural networks, red are the attention maps and green, the feature maps. (Color figure online)

The architecture used for the attention network will be now defined.

3.2 Attention Network

The attention network designed in the original paper is a simple fully convolutional neural network. Its architecture can be best viewed in Table 1. The attention is computed with only 4 layers and a low number of filters with the final layer outputing one feature map. This single feature map will represent the attention of a given scale S_n. While this network is not suitable to perform the labelling, it has a sufficient number of filters to spot where the attention should be focused on. This attention mechanism will specialize all attention scales to spot one type of characteristics such as background, text or noise.

Table 1. Architecture of the Attention Network as described by Grüning et al. [10]

Input	#Filters	Kernel size	Activation
S_n			
Conv2D	12	4	Relu
MaxPooling 2×2			
Conv2D	16	4	Relu
MaxPooling 2×2			
Conv2D	32	4	Relu
MaxPooling 2×2			
Conv2D	1	4	Relu

The next subsection will detail the feature extractor RU-net by first describing the general principal of the U-net architecture, then the RU-net will be explained with the addition of the SE module.

3.3 U-Net

The U-net architecture was originaly proposed by Ronneberger et al. [23] to perform semantic segmentation of biomedical images. An example of U-net architecture is shown in Fig. 3. It consists of a contracting path which captures the context and an expanding path which retrieves the original image size. The contracting and expanding paths are both symetrical thus the architecture can be seen as close to the U shape (hence the name). A path consists of blocks of convolutions followed by max pooling for the contracting path and upsampling for the expanding path. To increase the resolution of the expanding path, skip connections are used between symetrical blocks of both paths. Skip connections consists in concatenating the upsampled block with the corresponding contracting block. Then, a convolution layer can be applied to this concatenation to learn to combine both contexts. Therefore, when the next upsampling is performed it has the context of the contracting block and the context from the previous expanding block. Moreover, the expanding path starts with a high number of

filters which allow to take into consideration a lot of high level features. Also, it does not use fully connected layer and contains only pixel information so the spatial structure is kept. The advantage of the U-net is that it requires few images to have a good generalization [23].

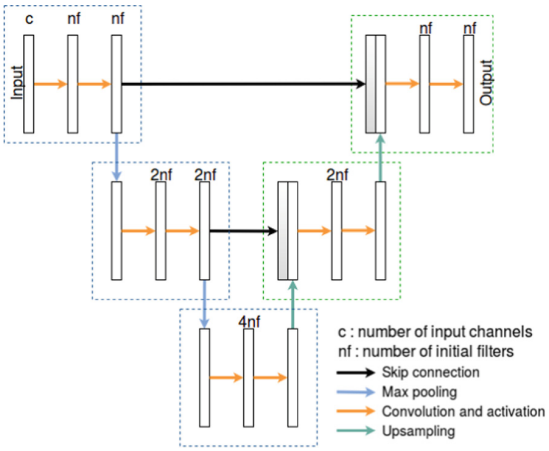
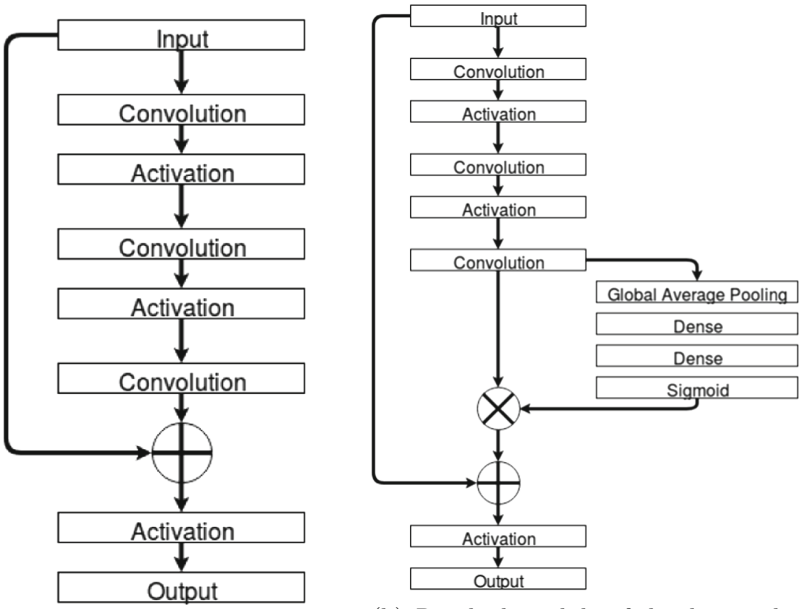


Fig. 3. Example of a U-net architecture with a depth of 3.

Now that the U-net architecture has been described, we need to define the blocks used to transform the U-net into a RU-net. Therefore, the next section describes the RU-net blocks.

3.4 RU-net

A RU-net is a U-net with residual connections within each blocks. It consists of two branches: the first one is the unactivated input, the second one is a classical chain of convolution layers followed by an activation function. The number of convolutions with activation block defines the depth of the residual connection. The last convolution layer is activated only after the unactivated input is summed to its unactivated output. The depth of the residual connection Res_{depth} corresponds to the number of convolution layers used. Figure 4(a) depicts the general principle of a residual connection of depth $Res_{depth} = 3$. To the residual modules we add the Squeeze and Excitation module which help to model an explicit relation between the feature maps (see Fig. 4(b)). First, the input feature maps are reduced to one feature by performing global average pooling. Then, two dense layers are applied to the feature maps to model the relation between them. Finally, the sigmoid activation is employed and its output is used to weight the original input feature maps. Compared to the original ARU-net architecture, we added the SE module to all residual blocks in the RU-net.



(a) Residual module of depth 3.

(b) Residual module of depth 3 with Squeeze and Excitation module.

Fig. 4. Residual module of depth 3 without and with SE module.

3.5 Full Image Binarizer

The goal of the full image binarizer is to detect the objects to binarize in the image. It means that it must find the approximate location of each object without detecting noises. In our case, the objects are the characters to binarize. Therefore, this network has the responsability to detect isolated noises such as bleed through, background degradations or ink drops. It aims to do so by capturing a large context of the image using the full image. Indeed, it is a lot easier to detect all these degradations when we can have the context of the full image. In fact, at a given pixel position, one may found bleed through very far from the closest main text making it hard to detect if we can only see bleed through. In the worst scenario, there is only bleed through in the image and the only thing that prevents the binarizer to falsely detect it is to have a global context.

One could employ this network to perform the binarization directly, but as it will be shown in the experiment section, this would greatly fail because of the downscaling which requires an upscaling step to match the original image size. In fact, this network will poorly detect the boundaries because of both the downsampling and the upsampling operations but will be stronger than patches to approximating the locations of characters.

3.6 Patch Image Binarizer

The goal of the patch image binarizer is to detect precisely the characters and especially their boundaries by using a local context on native resolution patches. We avoid, by using patches of the unresized images, the loss of precision due to the resizing made by the full image binarizer. However, this network can have trouble to identify noises that require a global context such as bleed through because of the constrained field of view on the local patch. Fortunately, these two networks have different complementary drawbacks and advantages that we can use by combining them to obtain a better prediction.

3.7 Prediction Combination

Before applying the predictions combination, predicted images are thresholded with the help of an analysis of the histogram. Therefore, a global threshold value is used to obtain for each binarizer a binary image. Then, the two predictions are simply passed through a bit-wise AND operator to get the final prediction. One could employ a more sofisticated mechanism to combine the predictions, however we choose in this paper to show that even a simple operation such as the bitwise AND operator improves the results.

4 Experiments

To verify the performance of the proposed approach, experiments are performed on the DIBCO competition on binarization. The next section describes the DIBCO competition and their metrics.

4.1 The DIBCO Competition

The DIBCO competition aims to evaluate the state of the art on the task of binarization. Documents are collected from two collections: IMPACT [1] for the printed and READ [2] for the handwritten images. We gathered training data from all the previous DIBCO competitions which gives us 87 training images. From those training images, 10 of them were randomly chosen for validation. The test consists in 10 printed and 10 handwritten images with several degradations such as bleed through, noises or holes in characters. The metrics used by the competitions are the FMeasure, the pseudo FMeasure, the PSNR and the DRD. The FMeasure defined by Eq. 2 is the harmonic mean between the precision and recall.

$$FM = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2}$$

The pseudo FMeasure [19] is the weighted FMeasure by considering some pixels more important than others. It uses, similarly to the FMeasure, the Pseudo Recall and Pseudo Precision and take into consideration the distance to the

contour of the ground truth. The PSNR, defined in Eq. 3, measures the pixel similarity between two images. The higher the value, the closer two images are from each others. Here C is the difference between foreground and background pixels, n the number of pixels in the image, y_i is the i-th pixel of the ground truth image and $\hat{y}_i$ is the i-th pixel of the predicted binary image.

$$PSNR = 10 \times \log(\frac{C^2}{MSE}) \tag{3}$$

where MSE is defined by Eq. 4

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

The Distance Reciprocal Distortion Metric or DRD [17], defined by Eq. 5, conveys the visual distortion as an human would see these distortions for binary images. It is defined in Eq. 5 where S is the number of wrongly classified pixels, $NUBN$ is the number of ground truth blocks of 8×8 which have at least one white and one black pixel. It uses, for each element in S, the DRD_k Eq. 6 which computes the distortion of a given pixel using a normalized weight matrix W_{Nm} of size 5×5. The ground truth block used to compute the distortion is centered around the k-th pixel at position (x, y) of the prediction y. The distortion is then the weighted sum of all pixels from the ground truth block that differs from the center pixels from y.

$$DRD = \frac{\sum_{k=1}^{S} DRD_k}{NUBN} \tag{5}$$

$$DRD_k = \sum_{i=-2}^{2}\sum_{j=-2}^{2} |y(i,j) - \hat{y}(x,y)| \times W_{Nm}(i,j) \tag{6}$$

4.2 Training Configuration

For the patch generation, random crops of fixed size are extracted from original images. Two patch sizes have been used for the experimentations: 512×512 and 1024×1024. Regarding full images, they are resized to have their maximum side to a maximum side MS. If the image maximum side $I_{side}^{max} = max(I^{width}, I^{height})$ is inferior to MS then the image is not resized. In other words, only images that are large, are resized while the others are left intact. Since the ARU-net requires the image to be divisible by a power of d and $d = 2$ in our case, we fixed $MS = 1024 = 2^{10}$. Remarkably, border padding is applied to images to rescaled images to make their size equal to $MS \times MS$ by adding zeros. This technique simplifies the U-net architecture as images will have the same size and therefore avoid the cropping of skip connections to fit the decoder size. This operation is summarized by Eq. 7. The RMSprop optimizer is used with an initial learning

rate of 0.001 which is exponentially decayed over time by a factor k = 0.03. The loss function used to optimize the networks is the cross-entropy. Filters of size 3 are used for the U-net and filters of size 4 for the AN. A $Res_{depth} = 3$ is used in combination with $Scale_{depth} = 5$ for the ARU-net and the RU-net has a depth of 5 with an initial number of filters equal to 8. We train each network until convergence is reached.

$$I = \begin{cases} I & \text{if } I_{side}^{max} < MS \\ I^{size} * MS / I_{side}^{max} & \text{else} \end{cases} \tag{7}$$

4.3 Data Augmentation

To improve the performances of both our networks, we make use of data augmentation. Online data augmentation is performed to provide a large amount of different images. Data augmentation aims to add variability in the data by generally introducing noises or linear transforms. Online data augmentation produces augmented images during the training. The interest of doing it in an online way is that it is unlikely that we will submit twice the exact same image to the network. Also, to make our networks more resilient to bleed through, we decided to use the original images with synthetic bleed through using DocCreator [13]. This software allows to apply several different degradations such as holes, 3D deformations, bleed through or phantom characters. In our experiments, we only used the bleed through augmentation with a varying intensity scale $\gamma \in [0.2; 0.5]$. An example of synthetic bleed through image is provided in Fig. 5. Patches and image are then augmented using random rotations with an angle $\theta \in [-180; 180]^\circ$ and they can be horizontally flipped with Gaussian or salt and pepper noises.

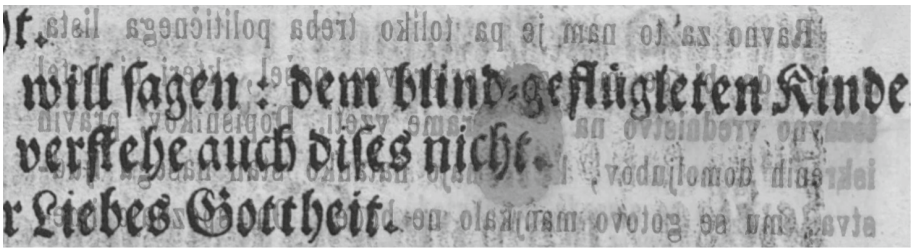


Fig. 5. Example of synthetic bleed through.

4.4 The Effect of Horizontal Flipping

We investigate the effect of horizontal flipping as one could think that it could confuse the network by having all the characters backwards compared to the bleed through. When there is no bleed through and the text does not have a lot more higher intensity than the background, if the image is horizontally flip

it can be considered as a realistic bleed through. A small experiment has been conducted to verify whether the use of the transform confuses or not a neural network. To do so, patch binarizer has been trained for 50 epochs with and without horizontal flipping. Figure 6 shows the loss value after each epoch for the two strategies.

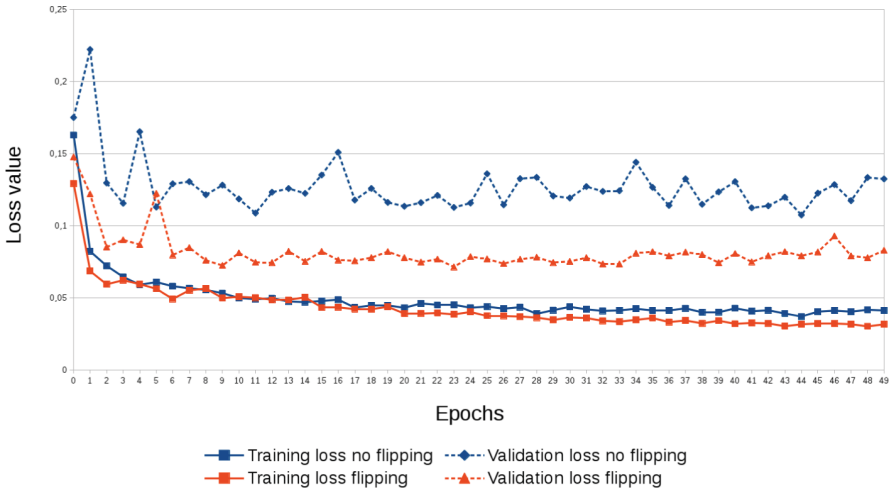


Fig. 6. The importance of horizontal flipping. Continuous lines represent training loss values and dashed lines, validation loss values.

Surprisingly, horizontal flipping is helping the network figuring out what the bleed through is. This can be explained by the fact that the text orientation does not matter so much but rather the relative pixel intensities between the main text and the bleed through. Usually, we can detect the bleed through by using only the text orientation (backwards characters) when there is only one kind of text. However, when we use random rotations, the text can be downward which makes it difficult for the network to distinguish downward from backward. Since all augmentation methods are independently performed on the image, we can also have both rotations and horizontal flips. Figures 7 and 8 show the differences between a downward text and backward text. For instance, Fig. 7 shows that is it not the text orientation that allows to distinguish the main text but only the pixels intensities. This is respected since we usually found ink of the same intensity in both recto and verso images. When the bleed through is present, the ink is overshadowed by the paper, thus making the bleed through lighter than the main text pixels.

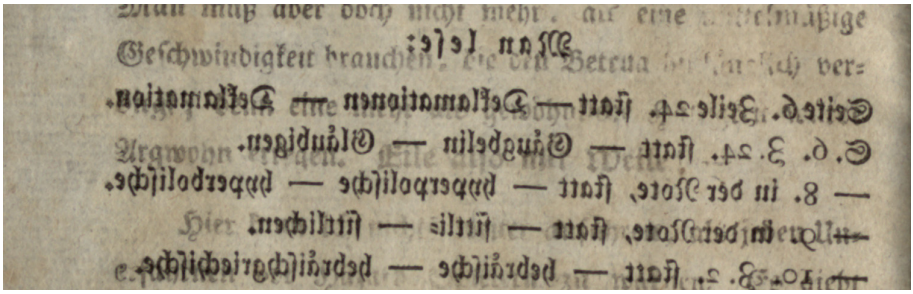


Fig. 7. Example of image flipped horizontally.

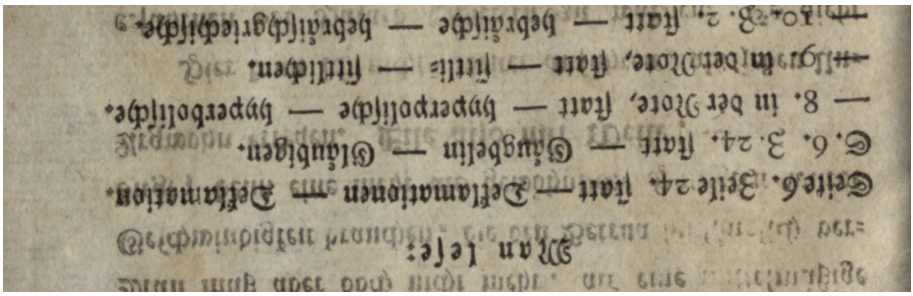


Fig. 8. Example of image rotated at 180°.

The only thing that remains to detect bleed through are the relative pixel intensities between the bleed through and the text pixels. The use of horizontal flipping improves the training because it helps the network learn this concept.

4.5 Results

Results of the experiments can be found in Table 2. One can see that not all methods improve the results compared to the state of the art. Those methods are the patch only with a patch size of 512 and the full image binarizer. While we know that the winner of DIBCO17 used patches, we do not know what size they used. However, when combining these two methods, we could obtain better results since they are complementary. To verify whether the patch context size influences the performance or not, we used patches of size 1024. The results shows that we obtained slightly better results but not for all metrics. Only the pseudo FMeasure did not improved, meaning that the errors made were more far from the ground truth (noises for example). When we combined the patches of size 1024 with the full image binarizer, we obtained again an improvement for all the metrics except for the pseudo FMeasure. Even if bigger patches improve the performances, using the full image binarizer, still helps to correct some mistakes even when using a simple combination of the prediction such as the logical AND operator.

Table 2. Results of our approach compared to the winner of DIBCO17.

DIBCO17 Test Set	FMeasure	Pseudo FMeasure	PSNR	DRD
Winner DIBCO17	91.04	92.86	18.28	3.4
Patch only 512	88.36	89.21	16.97	5.07
Patch only 1024	91.87	91.68	18.24	2.89
Image only	86.29	87.16	17.48	4.22
Combined patches 512	91.44	**93.05**	**18.519**	2.94
Combined patches 1024	**92.12**	92.49	**18.516**	**2.64**

4.6 Discussions

While the proposed approach outperforms the winning method of DIBCO17, it has the drawback of using two networks instead of one. Experiments showed that using big patches improves the results of the neural network. It suggests that if we could process images in their natural resolution it could further improve the state of the art. However, the results of the single neural network are still behind the combination of the two. Moreover, using the full rescaled image adds only little time, compared to the patch based network. Generally, most of the time used to compute the image prediction comes from the patch binarizer and not from the full image binarizer. The reason is simple: we need to label every pixel of the image at its original resolution with the patch binarizer in order to retrieve the prediction of the full image. For the full image binarizer, we have a lot less pixels to label because we predict on the downscaled images and upscale them after to obtain a prediction of the original size image. When using the big patches, the number of images to label is equal to $m + 1$ where m is the number of patches to label the whole image. Therefore, we can affirm that our approach adds only little execution time compared to other patch based methods when m is large (e.g. when processing large images).

5 Conclusion

We proposed in this paper a new robust method to binarize handwritten and printed document images. It consists in two fully convolutional neural networks with distincts objectives. The first one works on a reduced scale of the image and is precise to detect while the second woks globally. We show that these two networks alone are not capable of achieving better results. However, with a simple combination of the strength and weaknesses of the two FCNs, we were able to produce binarizations of very high quality. We showed that the proposed approach produces state of the art results on the DIBCO17 competition. While this technique works well, we still train the two networks independently. In future work, we plan on finding ways to add global image context to the patch binarizer so it can be trained end-to-end with a single neural network.

References

1. Impact project. http://www.impact-project.eu
2. Read project. http://read.transkribus.eu/
3. Afzal, M.Z., Pastor-Pellicer, J., Shafait, F., Breuel, T.M., Dengel, A., Liwicki, M.: Document image binarization using LSTM: a sequence learning approach. In: Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing, pp. 79–84. ACM (2015)
4. Almeida, M., Lins, R.D., Bernardino, R., Jesus, D., Lima, B.: A new binarization algorithm for historical documents. J. Imaging **4**(2), 27 (2018)
5. Alvarez, J.M., Gevers, T., LeCun, Y., Lopez, A.M.: Road scene segmentation from a single image. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7578, pp. 376–389. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33786-4_28
6. Calvo-Zaragoza, J., Gallego, A.J.: A selectional auto-encoder approach for document image binarization. arXiv preprint arXiv:1706.10241 (2017)
7. Fink, M., Layer, T., Mackenbrock, G., Sprinzl, M.: Baseline detection in historical documents using convolutional u-nets. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 37–42. IEEE (2018)
8. Gatos, B., Pratikakis, I., Perantonis, S.J.: Adaptive degraded document image binarization. Pattern Recogn. **39**(3), 317–327 (2006)
9. Giotis, A.P., Sfikas, G., Gatos, B., Nikou, C.: A survey of document image word spotting techniques. Pattern Recogn. **68**, 310–332 (2017)
10. Grüning, T., Leifert, G., Strauß, T., Labahn, R.: A Two-Stage Method for Text Line Detection in Historical Documents (2018). http://arxiv.org/abs/1802.03345
11. He, S., Wiering, M., Schomaker, L.: Junction detection in handwritten documents and its application to writer identification. Pattern Recogn. **48**(12), 4036–4048 (2015)
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507 (2017)
13. Journet, N., Visani, M., Mansencal, B., Van-Cuong, K., Billy, A.: DocCreator: a new software for creating synthetic ground-truthed document images. J. Imaging **3**(4), 62 (2017)
14. LeCun, Y., et al.: Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989)
15. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
16. Louloudis, G., Gatos, B., Pratikakis, I., Halatsis, C.: Text line detection in handwritten documents. Pattern Recogn. **41**(12), 3758–3772 (2008)
17. Lu, H., Kot, A.C., Shi, Y.Q.: Distance-reciprocal distortion measure for binary document images. IEEE Sig. Process. Lett. **11**(2), 228–231 (2004)
18. Niblack, W.: An Introduction to Digital Image Processing. Prentice-Hall, Englewood Cliffs (1986)
19. Ntirogiannis, K., Gatos, B., Pratikakis, I.: Performance evaluation methodology for historical document image binarization. IEEE Trans. Image Process. **22**(2), 595–609 (2013)
20. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)

21. Pastor-Pellicer, J., España-Boquera, S., Zamora-Martínez, F., Afzal, M.Z., Castro-Bleda, M.J.: Insights on the use of convolutional neural networks for document image binarization. In: Rojas, I., Joya, G., Catala, A. (eds.) IWANN 2015. LNCS, vol. 9095, pp. 115–126. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19222-2_10
22. Pratikakis, I., Zagoris, K., Barlas, G., Gatos, B.: ICDAR 2017 competition on document image binarization (DIBCO 2017). In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp. 1395–1403. IEEE (2017)
23. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
24. Sauvola, J., Pietikäinen, M.: Adaptive document image binarization. Pattern Recogn. **33**(2), 225–236 (2000)
25. Tensmeyer, C., Martinez, T.: Document image binarization with fully convolutional neural networks. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 99–104. IEEE (2017)
26. Westphal, F., Lavesson, N., Grahn, H.: Document image binarization using recurrent neural networks. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), pp. 263–268. IEEE (2018)
27. Wolf, C., Jolion, J.M., Chassaing, F.: Text localization, enhancement and binarization in multimedia documents. In: 2002 Proceedings of 16th International Conference on Pattern Recognition, vol. 2, pp. 1037–1040. IEEE (2002)
28. Afzal, M.Z., Krämer, M., Bukhari, S.S., Yousefi, M.R., Shafait, F., Breuel, T.M.: Robust binarization of stereo and monocular document images using percentile filter. In: Iwamura, M., Shafait, F. (eds.) CBDAR 2013. LNCS, vol. 8357, pp. 139–149. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05167-3_11