Document Image Binarization Using Recurrent Neural Networks

Florian Westphal, Niklas Lavesson, Håkan Grahn
Department of Computer Science and Engineering
Blekinge Institute of Technology
Karlskrona, Sweden
{*florian.westphal, niklas.lavesson, hakan.grahn*}*@bth.se*

***Abstract*—In the context of document image analysis, image binarization is an important preprocessing step for other document analysis algorithms, but also relevant on its own by improving the readability of images of historical documents. While historical document image binarization is challenging due to common image degradations, such as bleedthrough, faded ink or stains, achieving good binarization performance in a timely manner is a worthwhile goal to facilitate efficient information extraction from historical documents.**

In this paper, we propose a recurrent neural network based algorithm using Grid Long Short-Term Memory cells for image binarization, as well as a pseudo F-Measure based weighted loss function. We evaluate the binarization and execution performance of our algorithm for different choices of footprint size, scale factor and loss function. Our experiments show a significant trade-off between binarization time and quality for different footprint sizes. However, we see no statistically significant difference when using different scale factors and only limited differences for different loss functions. Lastly, we compare the binarization performance of our approach with the best performing algorithm in the 2016 handwritten document image binarization contest and show that both algorithms perform equally well.

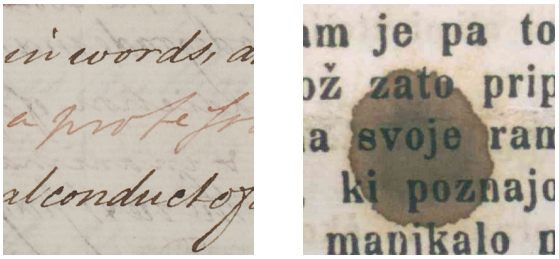
***Keywords*-image binarization; recurrent neural networks; Grid LSTM; historical documents**

I. Introduction

Image binarization, the separation of text foreground from page background, is an important processing step when analyzing historical document images. This separation enables algorithms for layout analysis or document transcription to focus only on the written text without the disturbing background noise. Similarly, image binarization improves document readability for human audiences. It is challenging, however, in historical documents, due to common degradations found in those documents, such as bleedthrough, faded ink or stains, as illustrated in Figure 1.

While many previous binarization algorithms were based on hand-crafted image processing steps, the recent rise of deep learning methods in document analysis has led to more and more deep learning based image binarization approaches. However, mostly the use of convolutional neural networks (CNNs) for binarization has been explored in the literature. To examine a different, less explored direction, we follow the idea of Afzal et al. [1] to use recurrent neural networks (RNNs) for image binarization.

The main difference between CNNs and RNNs are the RNN's recurrent connections, which allow the results of previous processing steps, called time steps, to affect the result of the current time step. This behavior of the recurrent connections allows the RNN to 'remember' previously seen inputs, which should enable it to make better binarization decisions by recognizing the relationship between pixels further apart from each other, such as pixels forming a character and pixels forming the character's diacritical mark. To avoid the vanishing gradient problem, a common issue in RNNs due to their large depth produced by the recurrent connections, Long Short-Term Memory (LSTM) cells are commonly used [3].



(a) Faded Ink and Bleedthrough. (b) Stains Covering Parts of the Text.

Figure 1. Common Image Degradations in Images of Historical Documents. Images from the DIBCO 2013 dataset [2].

In this paper, we present our work with the following main contributions:

- proposal of an RNN based binarization algorithm
- proposal of a dynamically weighted loss function
- evaluation of the impact of the algorithm's hyperparameters, footprint size, scale factor and loss function, on the binarization performance
- identification of a significant trade-off between binarization quality and time for different footprint sizes

Lastly, we compare our proposed approach with other state-of-the-art approaches. While our approach with a footprint size of 4, a scale factor of 2 and trained with our dynamic loss function performs equally well as the best performing algorithm from the 2016 binarization contest [4], it shows a lower performance than the currently best performing approach on the H-DIBCO 2016 dataset by Vo et al. [5].

II. Background

This section introduces the two main components this paper is based on. Those components are Grid Long Short-Term Memory (Section II-A), which is used to construct the proposed RNN based binarizer, and Pseudo F-Measure (Section II-B), the base for the proposed loss function.

A. Grid Long Short-Term Memory

Grid Long Short-Term Memory (Grid LSTM) networks, proposed by Kalchbrenner et al. [6], are a variant of LSTM networks [3]. The main difference to standard LSTM networks is that Grid LSTMs provide support for multidimensional processing. This support is achieved by arranging LSTM cells along each of the Grid LSTM's N dimensions, where the LSTM cells along a dimension have their own weight matrices $\mathbf{W_d}$, as well as their own memory vectors $\mathbf{m_d}$. However, the input to a Grid LSTM layer, consisting of the inputs from each input dimension $\mathbf{x_d}$ and the hidden vectors from all dimensions from the previous time step $\mathbf{h_d}$, is shared between all dimensions as one concatenated vector $\mathbf{h}$:

$$\mathbf{h} = \begin{bmatrix}\mathbf{x_1}^\intercal & \mathbf{h_1}^\intercal & \dots & \mathbf{x_N}^\intercal & \mathbf{h_N}^\intercal\end{bmatrix}^\intercal \tag{1}$$

Thus, the N outputs of an N-dimensional Grid LSTM layer are computed as follows:

$$\begin{aligned}(\mathbf{h'_1}, \mathbf{m'_1}) &= \mathrm{LSTM}(\mathbf{h}, \mathbf{m_1}, \mathbf{W_1}) \\ &\vdots \\ (\mathbf{h'_N}, \mathbf{m'_N}) &= \mathrm{LSTM}(\mathbf{h}, \mathbf{m_N}, \mathbf{W_N})\end{aligned} \tag{2}$$

Another feature of Grid LSTMs is the possibility to define priority dimensions. The idea here is that when computing the LSTM output of a priority dimension, the vector $\mathbf{h}$ contains the current time step hidden vectors $\mathbf{h'_d}$ of the non priority dimensions instead of the hidden vectors of the previous time step.

B. Pseudo F-Measure

Pseudo F-Measure (F_{ps}) is an evaluation measure for the binarization quality of a binarization algorithm for document images, which was proposed by Ntirogiannis et al. [7]. Unlike other commonly used evaluation measures, such as F-Measure or the peak signal-to-noise ratio (PSNR), F_{ps} takes into account the fact that labelling errors, which impact the readability of the actual text, are worse than those that do not affect the readability.

This focus on readability is achieved by weighting the labelling errors when computing pseudo recall (R_{ps}) and pseudo precision (P_{ps}), based on the location of the mislabelled pixels relative to the ground truth foreground. When computing R_{ps}, missed foreground pixels are penalized more and thus affect the final result more, the closer they are to the middle of the ground truth character line. Similarly, when computing P_{ps}, misclassified background pixels are penalized more if they are located in a line width wide area around a ground truth line. Within this area, pixels closer to the ground truth are penalized less than pixels that are further away.

III. Recurrent Binarization

The main contributions of this paper are the proposed Grid LSTM based network architecture for binarization, as described in Section III-A, and the customized loss function, described in Section III-B.

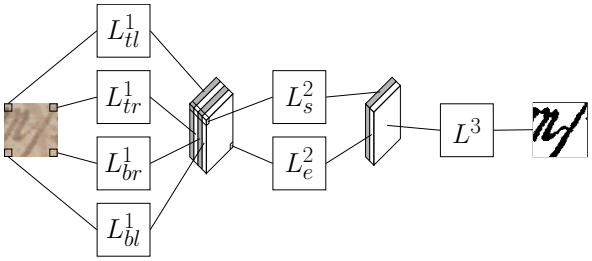


Figure 2. Overall Network Architecture

A. Architecture

In our approach, we divide the image to be binarized into non-overlapping blocks of 64×64 pixels, padding the image with zeros if necessary. These blocks are then converted into input sequences to the RNN and binarized separately from each other. In general, each image block is read by four separate Grid LSTM layers, each beginning to read the image from a different corner. This is illustrated in Figure 2 through the four different Grid LSTM blocks reading the provided image block from the top-left corner, L^1_{tl}, from the top-right corner, L^1_{tr}, from the bottom-left corner, L^1_{bl}, and from the bottom-right corner, L^1_{br}.

After processing the whole block, the output sequences from all four Grid LSTM layers are aligned and combined into one feature sequence. The purpose of the alignment is to ensure that all outputs correspond to the same respective input pixels. The produced feature sequence is then read again by two separate Grid LSTM layers in a bi-directional fashion with one layer beginning at the start of the sequence, L^2_s, and one beginning at the end, L^2_e. The final binarization result is then produced by a fully connected layer, L^3, which makes the final decision on the pixel labels based on the aligned and combined output sequences of the previous bi-directional Grid LSTM layer.

With this understanding of the general data flow through the network, we can now have a more detailed look on how the input Grid LSTM layer reads a given image block. The image block to process is first divided into smaller non-overlapping blocks of $f \times f$ pixels, with f being the configured footprint size. At each time step, one of these blocks is then converted into an input vector for the first input dimension $\mathbf{x_1}$ and processed by the respective Grid LSTM layer. The order in which these input blocks are processed depends on the respective input layer. For example, L^1_{tl} reads the blocks from left to right and from top to bottom, while L^1_{br} reads the blocks from right to left and from bottom to top.

Additionally to the input block in the first input dimension, we make use of the ability of Grid LSTMs to process multidimensional data and provide the input block above the current one as input vector $\mathbf{x_2}$ to the second input dimension. As shown in Figure 3, this second input block has the same size as the first input block, i.e., $f \times f$. In order to provide the network with even more context information, we provide a scaled down version of the surroundings of the first input block as input vector $\mathbf{x_3}$

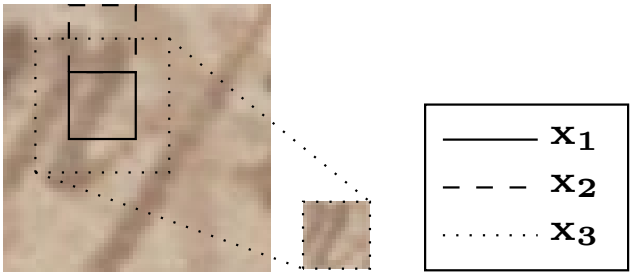


Figure 3. Network inputs for the 3 input dimensions, $\mathbf{x_0}$, $\mathbf{x_1}$, $\mathbf{x_2}$, for a footprint size of 16×16 and a scale factor of 2.

to the third input dimension. The area taken into account is determined by the configured scale factor sf, since the scaled down input block is required to have the same size as the other two blocks, i.e., $f \times f$. Thus, the size of the surrounding area considered is $(sf \cdot f) \times (sf \cdot f)$.

Provided with the three described input vectors $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$, each of the four Grid LSTM input layers produces its output in the first dimension, which is configured as priority dimension and thus is affected directly by the other two input dimensions and the one non-input dimension. As described before, this output is further processed by a bi-directional Grid LSTM layer, which has only one input and one non-input dimension.

Two interesting parameters of the described architecture are the footprint size f and the scale factor sf. For example, we would expect f to have an impact on training and binarization time, since larger footprints reduce the number of time steps per image block. Conversely, a large f may negatively impact the binarization performance, due to a loss of focus. Similarly, sf is interesting, since a larger value for sf provides more context information, but comes at the cost of a loss in resolution.

B. Dynamic Loss

The neural network architecture described in the previous section is trained using supervised learning. For this, it is important to quantify the difference between the current binarization result and the available ground truth as loss. This computed loss can then be used to appropriately adjust the tuneable weights in the neural network using back propagation through time. Since image binarization only has two labels, i.e., 0 for foreground and 1 for background, a suitable loss function would be the binary cross-entropy loss $\mathcal{L}$, as defined in Equation 3.

$$\mathcal{L} = y_i \cdot -log(\hat{y}_i) + (1 - y_i) \cdot -log(1 - \hat{y}_i) \quad (3)$$

This loss function produces large values if the label $\hat{y}_i$ assigned to pixel i is different from the target label y_i and is minimized during training. While this loss function provides us with a suitable measure of the current distance to the ground truth, it does not take into account the large class imbalance in document images, which is due to the fact that a page generally consists of more background than foreground. In order to account for this class imbalance, a static weight w can be added to $\mathcal{L}$ to emphasize certain labelling errors more than others. As shown in the resulting loss function $\mathcal{L}_{stat}$ in Equation 4, the added weight w allows to configure the relative importance of background labelling errors over foreground errors.

$$\mathcal{L}_{stat} = w \cdot y_i \cdot -log(\hat{y}_i) + (1 - y_i) \cdot -log(1 - \hat{y}_i) \quad (4)$$

While applying a static weight is a reasonable heuristic to address class imbalance, it does not encourage actual readability of the binarized document. Therefore, we propose to use the idea of pseudo F-Measure (F_{ps}) [7] in the loss function and to weight labelling errors dynamically depending on their location to promote readability. The loss function for this dynamic loss $\mathcal{L}_{dyn}$ can be formulated as follows:

$$\begin{aligned}\mathcal{L}_{dyn} = {} & w_i \cdot y_i \cdot -log(\hat{y}_i) \\ & + w_i \cdot (1 - y_i) \cdot -log(1 - \hat{y}_i)\end{aligned} \quad (5)$$

The dynamic label weights w_i for background pixels are computed in the same way as the weights for pseudo precision (P_{ps}) and penalize especially background labelling errors, which occur in a certain area around foreground elements. The weights for foreground pixels on the other hand are computed as the weights for pseudo recall (R_{ps}) and penalize foreground label errors more, which affect the center of a line.

IV. Experiment Design

In this study, we assess the overall binarization performance of our proposed approach, as well as the impact variations of the footprint size f, the scale factor sf and the chosen loss function have on this performance. For this evaluation, we have implemented[1] our proposed approach using TensorFlow 1.0.1[2] and performed all experiments on a computer with an Intel i7-6700K quad-core CPU @ 4.00 GHz, 32 GB DDR4 RAM and an Nvidia GeForce GTX 980. In all experiments, we used the competition datasets from DIBCO 2009 [8], H-DIBCO 2010 [9], DIBCO 2011 [10], H-DIBCO 2012 [11], DIBCO 2013 [2], H-DIBCO 2014 [12] and H-DIBCO 2016 [4]. The achieved binarization performance was evaluated using the measures commonly used in these competitions, i.e., F-Measure, F_{ps}, the peak-signal to noise ratio (PSNR) and the distance reciprocal distortion metric (DRD).

In general, we train our binarization algorithm on 5 out of the 7 datasets, while one dataset is used for validation and one is used for testing. For each evaluated configuration, we train the network 7 times, so that each dataset is used once for validation and once for testing. In each training run, we train the network from scratch for 201 epochs, which has been found sufficient for all network configurations to converge. Here, one epoch consists of 2048 image blocks, which is processed in batches containing 32 image blocks. For training, we use the Adam

[1]The implementation and additional experiment raw data can be found at: https://github.com/FlorianWestphal/DAS2018

[2]https://www.tensorflow.org

optimization algorithm [13] with TensorFlows' default parameters. After completing the training, we choose the model which produces the lowest labelling error on the validation set for evaluation.

For our evaluation of the impact of f on the binarization performance, we evaluate 3 different footprint sizes, f_2, f_4 and f_8 with a value of 2, 4 and 8 for f respectively. Since we use 5 LSTM cells per input pixel and input dimension, the number of LSTM cells in all Grid LSTM layers varies depending on f. In order to avoid an unfair comparison between f_2 and f_4, we increase the number of LSTM cells per layer for f_2 to 20 cells per input pixel. This ensures similar model complexities for these two configurations. However, based on initial measurements, we do not see the need to adjust the model complexity of f_2 or f_4 when comparing these configurations to f_8. Additionally to the binarization performance, we evaluate the impact of f on the training and binarization time.

The impact of sf on the binarization performance is evaluated for 3 different scale factors, sf_1, sf_2 and sf_4 with a value of 1, 2 and 4 for sf respectively. Here, sf_1 represents the case in which no scale factor is used, since it results in $\mathbf{x_1}$ and $\mathbf{x_3}$ being identical at each time step.

Lastly, we evaluate the impact of the used loss function on the binarization performance. As loss functions, we use the binary cross-entropy loss $\mathcal{L}$, statically weighted loss functions with a weight of 2, $\mathcal{L}^2_{stat}$, and 0.5, $\mathcal{L}^{0.5}_{stat}$, and the proposed dynamically weighted loss function $\mathcal{L}_{dyn}$.

V. Results and Analysis

In the following, we present the results of our experiments to evaluate the impact of footprint size (Section V-A), scale factor (Section V-B), and loss function (Section V-C) on the binarization performance of the proposed algorithm. Additionally, we compare the binarization performance of the best configuration of our algorithm with other state-of-the-art methods (Section V-D).

A. Footprint Impact

As described in the previous section, we measure the impact of different footprint sizes on the training and binarization time. Table I shows the average training times over the 7 training runs, as well as the average binarization time for all test set images of the 7 runs. These measurements show that increasing the footprint size and thus decreasing the number of time steps per image block indeed decreases the training and binarization time. While going from f_2 to f_4 yields the expected speedup of 4, a speedup of only 2 is achieved going from f_4 to f_8. This reduced speedup is probably caused by the lower number of LSTM cells in f_4 compared to f_8.

While reducing training and binarization time is a worthwhile goal, this must not come at the expense of binarization quality. Therefore, we also analyze the footprint size's impact on the binarization result. As shown in Figure 4, the binarization quality as measured in F_{ps} is reduced by choosing a footprint size of 8. In order to quantify this observation, we perform the Friedman test

Table I
Average training and binarization times for the evaluated footprint values.

	f_2	f_4	f_8
Training (in hours)	29.6248	7.1041	3.6841
Binarization (in seconds)	8.4149	2.2259	1.4121

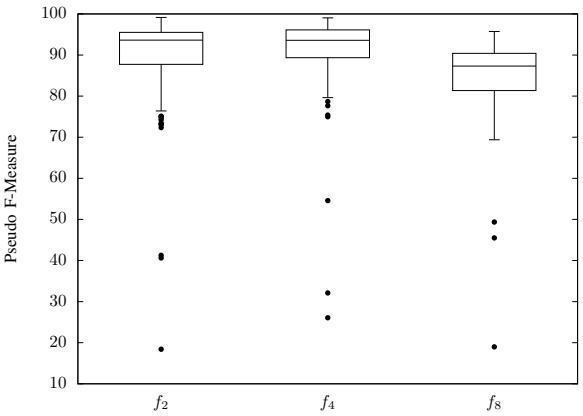


Figure 4. F_{ps} results for the evaluated variations in footprint size for all test set images. Boxes illustrate the first, second and third quartile; Whiskers indicate the lowest and the highest value within the 1.5 interquartile range; Dots mark outliers outside this range

to assess if there is a statistically significant difference in the distributions of the measured binarization quality for all images from the test sets for any of the quality measures. Based on the results shown in Table II, we can see that there is a statistically significant difference between the footprint sizes at the $p < 0.05$ level. An afterwards performed pairwise Wilcoxon rank-sum test with Holm correction shows that there is no statistically significant difference in binarization quality between f_2 and f_4, but between f_2 and f_8, and f_4 and f_8 for all 4 quality measures. This shows that f_4 yields the best trade-off between execution and binarization performance.

B. Scale Factor Impact

Similar to the previous analysis, we perform the Friedman test to identify differences in the measured binarization performance for the 3 evaluated scale factors. Based on the p-values reported in Table II, we can see that there is a statistically significant difference between those scale factors at the $p < 0.05$ level only for F-Measure.

Table II
Results of the Friedman test for the 4 used binarization quality measures for the evaluated variations in footprint, scale factor and loss function.

Measure	Footprint		Scale Factor		Loss Function	
	χ^2	p	χ^2	p	χ^2	p
F-Measure	86.116	< 0.0001	6.0233	0.049	25.409	< 0.0001
F_{ps}	80.023	< 0.0001	4.4651	0.107	81.516	< 0.0001
PSNR	82.977	< 0.0001	5.5116	0.064	27.028	< 0.0001
DRD	76.256	< 0.0001	4.2558	0.119	16.995	0.0007

Table III
RANKING OF ALL EVALUATED CONFIGURATIONS BASED ON ALL TEST SET IMAGES AND ALL 4 BINARIZATION QUALITY MEASURES.

Configuration	Score	Configuration	Score
f_4 - sf_2 - $\mathcal{L}_{dyn}$	1158.0	f_4 - sf_2 - $\mathcal{L}^2_{stat}$	1454.0
f_4 - sf_1 - $\mathcal{L}_{dyn}$	1269.0	f_2 - sf_2 - $\mathcal{L}_{dyn}$	1474.0
f_4 - sf_2 - $\mathcal{L}$	1368.0	f_4 - sf_2 - $\mathcal{L}^{0.5}_{stat}$	1780.0
f_4 - sf_4 - $\mathcal{L}_{dyn}$	1381.0	f_8 - sf_2 - $\mathcal{L}_{dyn}$	2500.0

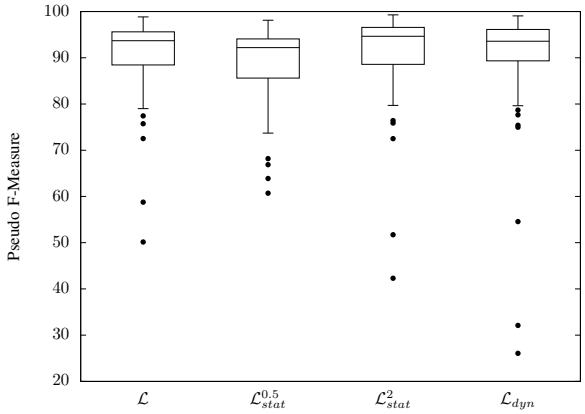


Figure 5. F_{ps} results for the evaluated loss functions for all test set images.

However, the post-hoc test performed for F-Measure does not indicate any statistically significant differences for any pair of scale factors. While there is no significant difference between the scale factors, we can observe a certain trend by ranking all evaluated configurations based on their performance on each test set image for each of the 4 binarization quality measures. The results of this ranking in Table III show that for the used datasets a scale factor of 2 tends to produce better binarization results than the other 2 evaluated scale factors.

C. Loss Function Impact

Figure 5 shows the binarization performance achieved by models built using one of the 4 analyzed loss functions on their respective test dataset. This box plot shows that all used loss functions produce a similar binarization performance with $\mathcal{L}^{0.5}_{stat}$ yielding slightly worse binarization results. The results of the performed Friedman test, as shown in Table II, indicate a statistically significant difference between the analyzed loss functions at the $p < 0.05$ level. When analyzing this difference using a pairwise Wilcoxon rank-sum test with Holm correction, we see that $\mathcal{L}^{0.5}_{stat}$ is significantly worse than the other 3 loss functions only for F_{ps}. For all other quality measures, there is no statistically significant difference between any of the loss functions. However, based on the ranking in Table III, we can observe the general trend for the analyzed datasets that $\mathcal{L}^{2}_{stat}$ and $\mathcal{L}_{dyn}$ perform similarly well, while $\mathcal{L}$ and $\mathcal{L}^{0.5}_{stat}$ perform worse.

Table IV
COMPARISON OF DIFFERENT IMAGE BINARIZATION APPROACHES ON THE H-DIBCO 2016 DATASET [4].

Measure	Hassaine et al.	Vo et al.	f_4-sf_2-$\mathcal{L}_{dyn}$
FM	88.72 ± 4.68	$\mathbf{90.10}$	88.79 ± 4.80
F_{ps}	91.84 ± 4.24	$\mathbf{93.57}$	92.53 ± 4.2
PSNR	18.45 ± 3.41	$\mathbf{19.01}$	18.43 ± 3.11
DRD	3.86 ± 1.57	$\mathbf{3.58}$	3.98 ± 1.85

D. Binarization Performance

Based on the performed statistical analyses and the ranking in Table III, we can see that the best performing configuration of our binarization algorithm uses a footprint size of 4 (f_4), a scale factor of 2 (sf_2) and was trained using our proposed dynamically weighted loss function ($\mathcal{L}_{dyn}$). In Table IV, we compare this configuration with the mean-wise best performing method in the H-DIBCO 2016 contest [4], i.e., the method by Hassaine et al., and with the current state-of-the-art on the H-DIBCO 2016 dataset, i.e., the method by Vo et al. [5]. For this comparison, we use the models, which were trained using the H-DIBCO 2016 dataset as test set. From Table IV, one can see that our approach has a higher performance than the method by Hassaine et al. on F-Measure (FM) and pseudo F-Measure (F_{ps}), while it has a lower performance than this method on PSNR and DRD. However, the method by Vo et al. [5] shows a higher performance than all other approaches.

VI. RELATED WORK

Over the past decades, many algorithms for image binarization have been proposed. Those algorithms include global threshold based methods, such as Otsu's method [14], local thresholding methods, such as the method by Sauvola and Pietikäinen [15], as well as graph cut based methods, such as Howe's algorithm [16]. In more recent years, the trend in document image binarization has shifted from those previous algorithms using hand-crafted features towards machine learning based methods mostly relying on deep learning techniques. Most deep learning based image binarization algorithms use convolutional neural networks (CNNs) [17], such as the approach proposed by Pastor-Pellicer et al. [18], or variations thereof, such as the deep supervised network (DSN) approach by Vo et al. [5] or the fully convolutional neural network (FCN) approach by Tensmeyer and Martinez [19].

However, the work most closely related to our own approach is the approach by Afzal et al. [1], which uses recurrent neural networks (RNNs) with long short-term memory (LSTM) cells for image binarization. The main difference to this approach is our evaluation of different footprint sizes, scale factors and loss functions, as well as an overall improved network architecture.

While not related to our approach in terms of basic network structure, the work by Tensmeyer and Martinez [19] is related to our work in their use of pseudo F-Measure (F_{ps}) for training. The difference between

between their and our approach is that they use F_{ps} directly as loss function for training, while we use only the weights of F_{ps} in a weighted cross-entropy loss function.

VII. Conclusion

In this paper, we have presented an RNN based binarization algorithm, which used the ability of the employed Grid LSTM cells to handle multidimensional input to incorporate context information in each binarization step. We have evaluated the execution and binarization performance of our approach when using different footprint sizes and have identified a significant trade-off between execution time and binarization quality with respect to footprint size. We showed that increasing the footprint size significantly reduces training and binarization time, at the cost of significantly reducing binarization performance when increasing the footprint size from 4 to 8. Thus, we conclude that a footprint size of 4 yields the best trade-off between execution and binarization performance.

Furthermore, we have analyzed the impact of different scale factors for additional context information on the binarization performance. However, we have not found any statistically significant differences in the binarization performance of the evaluated scale factors. Therefore, we can only make the observation, based on a ranking of all evaluated configurations, that a scale factor of 2 tends to perform better on the used DIBCO and H-DIBCO datasets.

Lastly, we have evaluated the impact of different loss functions on the binarization performance. Apart from the statistically significant drop in binarization performance as measured by F_{ps} when using a static weight, which penalizes foreground errors stronger than background errors, we could not identify any other statistically significant differences between the analyzed loss functions. However, the observed trend based on a ranking of all configurations is that the proposed dynamically weighted cross-entropy loss tends to perform well.

In a comparison based on binarization performance on the H-DIBCO 2016 dataset between two state-of-the-art solutions and our approach, our approach ties with the best performing algorithm in the H-DIBCO 2016 competition [4], but shows a lower performance than the currently best performing approach on this dataset.

Acknowledgment

This work is part of the research project "Scalable resource-efficient systems for big data analytics" funded by the Knowledge Foundation (grant: 20140032) in Sweden.

References

[1] M. Z. Afzal, J. Pastor-Pellicer, F. Shafait, T. M. Breuel, A. Dengel, and M. Liwicki, "Document Image Binarization using LSTM: A Sequence Learning Approach," in *Proc. 3rd Int. Workshop on Historical Document Imaging and Processing*, 2015, pp. 79–84.

[2] I. Pratikakis, B. Gatos, and K. Ntirogiannis, "ICDAR 2013 Document Image Binarization Contest (DIBCO 2013)," in *Int. Conf. Document Analysis and Recognition*, 2013, pp. 1471–1476.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] I. Pratikakis, K. Zagoris, G. Barlas, and B. Gatos, "ICFHR2016 Handwritten Document Image Binarization Contest (H-DIBCO 2016)," in *Int. Conf. Frontiers in Handwriting Recognition*, 2016, pp. 619–623.

[5] Q. N. Vo, S. H. Kim, H. J. Yang, and G. Lee, "Binarization of degraded document images based on hierarchical deep supervised network," *Pattern Recognition*, vol. 74, pp. 568–586, 2018.

[6] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid long short-term memory," *arXiv preprint arXiv:1507.01526*, 2015.

[7] K. Ntirogiannis, B. Gatos, and I. Pratikakis, "Performance Evaluation Methodology for Historical Document Image Binarization," *IEEE Trans. Image Process.*, vol. 22, no. 2, pp. 595–609, Feb 2013.

[8] B. Gatos, K. Ntirogiannis, and I. Pratikakis, "DIBCO 2009: document image binarization contest," *Int. J. Document Analysis and Recognition*, vol. 14, no. 1, pp. 35–44, 2011.

[9] I. Pratikakis, B. Gatos, and K. Ntirogiannis, "H-DIBCO 2010-Handwritten Document Image Binarization Competition," in *Int. Conf. Frontiers in Handwriting Recognition*, 2010, pp. 727–732.

[10] ——, "ICDAR 2011 Document Image Binarization Contest (DIBCO 2011)," in *Int. Conf. Document Analysis and Recognition*, 2011, pp. 1506–1510.

[11] ——, "ICFHR 2012 competition on handwritten document image binarization (H-DIBCO 2012)," in *Int. Conf. Frontiers in Handwriting Recognition*, 2012, pp. 817–822.

[12] K. Ntirogiannis, B. Gatos, and I. Pratikakis, "ICFHR2014 Competition on Handwritten Document Image Binarization (H-DIBCO 2014)," in *Int. Conf. Frontiers in Handwriting Recognition*, 2014, pp. 809–813.

[13] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285, pp. 23–27, 1979.

[15] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern recognition*, vol. 33, no. 2, pp. 225–236, 2000.

[16] N. R. Howe, "A Laplacian Energy for Document Binarization," in *Int. Conf. Document Analysis and Recognition*, 2011, pp. 6–10.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[18] J. Pastor-Pellicer, S. España-Boquera, F. Zamora-Martínez, M. Z. Afzal, and M. J. Castro-Bleda, "Insights on the use of convolutional neural networks for document image binarization," in *Advances in Computational Intelligence*, 2015, pp. 115–126.

[19] C. Tensmeyer and T. Martinez, "Document image binarization with fully convolutional neural networks," in *Int. Conf. Document Analysis and Recognition*, 2017, pp. 99 – 104.