Training Strategies for OCR Systems for Historical Documents

Jiří Martínek, Ladislav Lenc, Pavel Král

Training Strategies for OCR Systems for Historical Documents

Jiří Martínek[1,2], Ladislav Lenc[1,2], and Pavel Král[1,2]

[1] Dept. of Computer Science & Engineering
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic
[2] NTIS - New Technologies for the Information Society
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic
{jimar,llenc,pkral}@kiv.zcu.cz

Abstract. This paper presents an overview of training strategies for optical character recognition of historical documents. The main issue is the lack of the annotated data and its quality. We summarize several ways of synthetic data preparation. The main goal of this paper is to show and compare possibilities how to train a convolutional recurrent neural network classifier using the synthetic data and its combination with a real annotated dataset.

Keywords: CNN · Historical Documents · LSTM · Neural Network · OCR · Synthetic Data

1 Introduction

The efforts to preserve the cultural heritage of historical documents has become a significant task in the document processing field. A reasonable information retrieval on scanned documents is possible only after the text recognition process though. Optical character recognition (OCR) and handwritten text recognition (HTR) need to be implemented with a respect to the historical domain. In such a domain we often struggle with the quality of historical document and with a lack of annotated data.

Line–based OCR approaches are currently very widespread and utilized by the state–of–the–art systems. They use a whole text line as an input instead of individual characters. Therefore, it is not necessary to perform character segmentation, which is error–prone. Within this paper, we focus on a historical printed German Fraktur dataset.

A combination of deep convolutional and recurrent neural networks (CRNN) is nowadays used for line–based OCR [2]. Long short–term memory (LSTM) [8] is often used within the CRNN model. For this task, we used the a CRNN [6], [15] model inspired by the authors of [14].

To be able to train such a line–based OCR system, we need to provide a big annotated corpus. Manual annotation is a very time–consuming way of dataset creation and hence our annotated corpus consists of only ten manually annotated pages (1368 images of text lines). Since we split the above–mentioned dataset into training, validation and test sets, we consider the training set to be too small and inappropriate for training a neural network classifier. The lack of training data is a significant issue with regard to the ability to learn a language model. It was shown that an implicit language model is an important part of the OCR system [13].

Using synthetic data for OCR is a considerable step in historical documents processing. Jaderberg et al. present a framework for recognition and synthetic data generation [9]. Margner et al. [10] present synthetic data for Arabic OCR and another synthetic data generation was proposed by Gaur et al. [4].

There are a number of tools (i.e. tool for Arabic OCR in [10] or Aletheia proposed in [3]) that deal with the synthetic data generation and annotation. One of our synthetic datasets utilizes the Ocropus OCR System (more precisely OCRopy–linegen) [1], which is another example of a tool for text recognition and synthetic data generation.

The main contribution of this paper is to give an overview of alternative training strategies with a small amount of annotated data available. Three learning strategies will be proposed for this purpose: the first one uses only a few annotated real pages, the second one considers only synthesized text lines and the third method combines generated data with the real ones. We will also propose two generation methods which compose generated text-lines from real characters. We will compare the performance of these methods with OCRopy–linegen tool. We will further experiment with different white–space padding at the beginning of a line image.

The paper is organized as follows. The following section describes our CRNN classifier. The Section 3 describes the dataset. In Section 4 we show the synthetic data creation process. The Section 5 analyzes the training strategies. Experiments and results are shown in Section 6. The final section concludes the paper.

2 CRNN Classifier

The CRNN classifier use connectionist temporal classification (CTC) output layer. CTC is an output layer designed for sequence labeling with RNNs [5]. We utilize a CNN–LSTM line image classifier. It is capable of classifying unsegmented line images and retrieving the character sequences.

The input of our network is binarized line images with a height of 40 pixels. CNN and max–pooling layers reduce the dimension and extracts features, which are fed to the RNN, more precisely two bidirectional LSTM with 256 units. The output of the bidirectional LSTM layer is a set of dense layers with the softmax activation function. It represents a probability distribution of characters per each time frame. Let $\mathcal{A}$ be a set of symbols. The most probable symbol $\hat{a}_t$ of each

time frame t is determined as:

$$\hat{a}_t = \underset{a_i \in \mathcal{A}}{\operatorname{argmax}}\, p_t^{a_i} \tag{1}$$

where $p_t^{a_i}$ is a probability of observing character a_i at a given time t.

The final part of the classifier is a transcription layer, which decodes the predictions for each frame into an output sequence. The architecture of the classifier is depicted in Figure 1.

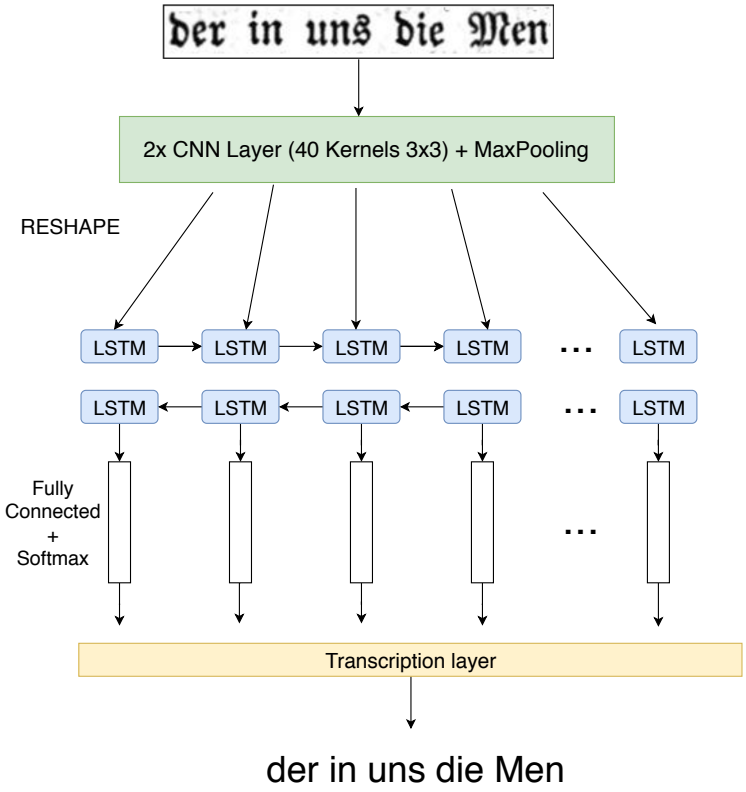

Fig. 1. CRNN Architecture

3 Dataset

Our manually annotated dataset consists of ten two–column pages of German newspaper from the second half of the nineteenth century. We chose two pages for testing, one for validation and the remaining seven for training. Table 1 shows detailed statistics of the dataset. A fragment of one page is shown in Figure 2.

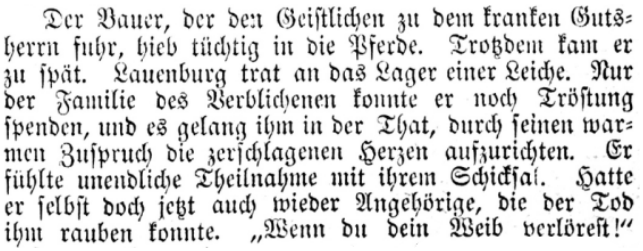
Der Bauer, der den Geiſtlichen zu dem kranken Gutsherrn fuhr, hieb tüchtig in die Pferde. Trotzdem kam er zu ſpät. Lauenburg trat an das Lager einer Leiche. Nur der Familie des Verblichenen konnte er noch Tröſtung ſpenden, und es gelang ihm in der That, durch ſeinen warmen Zuſpruch die zerſchlagenen Herzen aufzurichten. Er fühlte unendliche Theilnahme mit ihrem Schickſal. Hatte er ſelbſt doch jetzt auch wieder Angehörige, die der Tod ihm rauben konnte. „Wenn du dein Weib verlöreſt!“

Fig. 2. Example of the historical text

	Page #	**Line #**	**Word #**	**Character #**
Train	7	955	7653	50 426
Val	1	138	1084	6 669
Test	2	275	2163	13 828

Table 1. Statistical dataset info

4 Synthetic Data Creation

This section analyzes different types of synthetic data creation. Before getting to that, though, since the CRNN uses LSTM layers, the implicit language models issue is a well–posed question.

In [13] authors have shown that the implicitly learned language model can improve the recognition accuracy. Therefore, the texts used for synthetic data generation should be aimed at the domain, we work in.[3].

There are many ways to create synthetic text lines. First of all, we must provide a text source in the target domain. Then the simplest way to create an image of the text line is to take images of individual characters and put them together. It must be done with a respect to the target font though.

It is very convenient to have several different examples of each symbol. Then we ought to use a randomly picked image for a given symbol, due to the greater diversity of the synthetic dataset. This way can be considered as a simple form of data augmentation technique (e.g. Perez et al. [12]) because from one source text line it is possible to create several different corresponding images.

Another thing to consider is gaps between characters. The gap should be chosen with regard to a font style. The simplest solution is to estimate a constant pixel value and use it as a gap for each pair of letters. Another way is to specify a range (e.g. 1 px – 5 px) and use a random value within this range. Finally, it is possible to use real examples to provide precise values of gaps between each pair of characters. However, because of the quality of scans and noise, the extracted precise values could be distorted. Therefore we chose to use the random gaps approach with a range 1 px – 5 px. In Figure 3 there is an example of generated random space data.

[3] In our case historical German texts from the second half of the nineteenth century.

Fig. 3. Examples of generated data with random space.

4.1 Synthetic Data Generator

An alternative way of creating such a synthetic dataset is to use a text generating tool (e.g. OCRopy–linegen [1]). The tool usually has the possibility to choose a font, source texts and several types of image transformations (skewing, warping etc.) to make the desired image of the text line. Every rendered character is always the same and then it could be very useful in the case of printed text (see Figure 4). The images of annotated text lines are depicted in Figure 5.

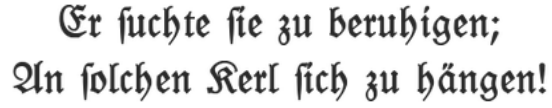

Fig. 4. Examples of generated data by OCRopy–linegen.

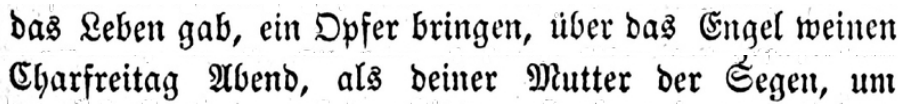

Fig. 5. Examples of real dataset lines.

5 Training Strategies

Within this section, we present three strategies for CRNN training.

Strategy 1: use only training part of the annotated real corpus for training

Strategy 2: use only synthetic data for training

Strategy 3: use synthetic data first and then gradually add pages of the training part of the annotated real corpus for training.

The first strategy is straightforward and does not use synthetic data. The second one is basically a comparison of the qualities of the synthetic data without the influence of the real dataset. Finally, the last strategy combines the synthetic and the annotated data and tries to find optimal settings (volume of pages and number of epochs).

For each strategy, there are more possibilities to enrich a training process, such as data augmentation, binarization or different white–space padding at a beginning of an image. The following experimental section covers the binarization and white–space padding issues.

6 Experiments

Each one of the previously introduced strategies is evaluated on the validation dataset. We developed two different types of synthetic data. The first one contains character images with random gaps and the second one uses the OCRopus (OCRopy-linegen).

Since there is only one text source, all these synthetic datasets should have the same setup from the point of view of the implicit language model.

6.1 Binarization and White–space–padding experiment

The original real images have an 8–bit grayscale png format. Using OpenCV[4] we thresholded the images to make them binarized by the Otsu's method [11]. After the first set of experiments, we discovered that binarized images obtained better results than grayscale ones, which is in compliance with Gupta et al. [7]. Hence we used the binarized images for all experiments.

The white–space–padding experiment verifies the influence of a white–space–padding at the beginning of the image (see Figure 6). The standard white–space padding is about 10 pixels (i.e. the white space between the leftmost edge of the image and the position of the first black pixel which is part of the text). We extended the padding by 50 pixels and we used the approach of Strategy 1 for the comparison (see Table 2).

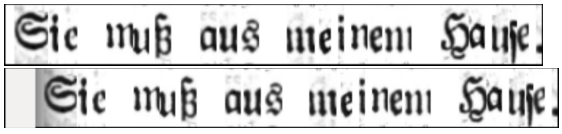

Fig. 6. Illustration of two different white–space–padding examples

The results show that it is better to use the larger padding. For further experimental setups, all images are padded by 50 px white space to reach about 60 px padding in total, which has been determined as the appropriate value.

[4] https://opencv.org/

	Train loss	Val loss	WER	CER	Edit distance
60 px padding	0.056	3.030	0.068	0.014	0.464
10 px padding	0.022	3.341	0.150	0.030	1.072

Table 2. Comparison of the influence of white–space – 100 epochs with real data

6.2 Annotated Dataset Experiment – Strategy 1

In this experiment, we tried to use seven pages of the annotated dataset to train a CRNN. The results of this baseline approach, as well as the course of training, are depicted in Figure 7.

The left image shows train and validation CTC loss, while the right one shows the character error rate (CER) and the word error rate (WER). Either of these images shows that after approximately 80 epochs of training the model reaches the best results and then starts to stagnate. The model is trained using the stochastic gradient descent (SGD) algorithm and the learning rate is set to 0.005. We tried to change the value of the learning rate and roughly speaking the results after 100 epochs were very similar and the only minor change was a slightly different course of training. Table 3 shows the results after 100 epochs.

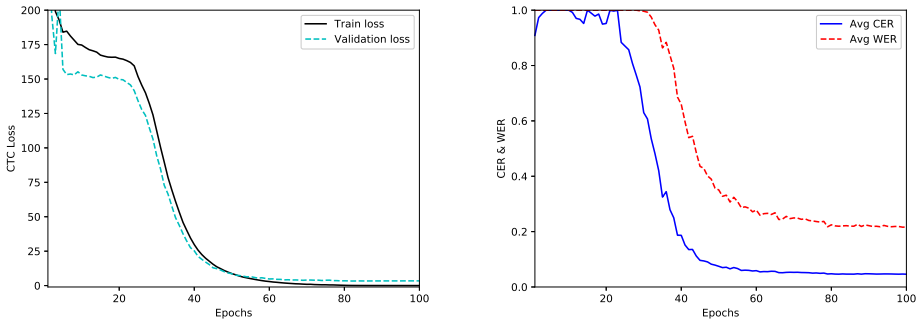

Fig. 7. WER, CER, Train & Val loss after 100 epochs with the learning rate 0.005 from the training annotated real dataset

Train loss	Val loss	WER	CER	Edit distance
0.056	3.030	0.068	0.014	0.464

Table 3. 100 epochs experiments results

The experiment showed that the model reached excellent results on the validation data and no significant overfitting occurred. On the other hand, the main

drawback is that limited implicit language model has been learned (only 955 unique text lines). Another important issue is a very long learning time (at least 80 epochs).

6.3 Synthetic Dataset Experiment – Strategy 2

This experiment compares the quality of our synthetic datasets. In the following Figures, we present the results of 25 epochs of training. The train loss of OCRopy–linegen dataset is after 25 epochs much lower and learning occurs faster, but on the other hand, the validation results (WER, CER and Edit distance) are better in the case of the Random space dataset. We can conclude that the Random space dataset is more similar to the real data.

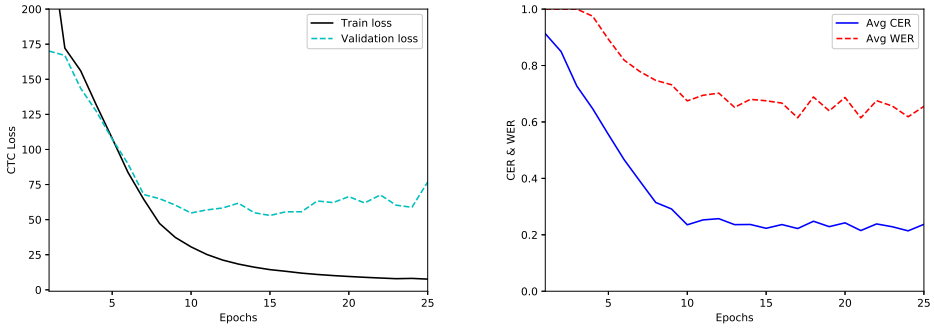

Fig. 8. WER, CER, Train & Val loss after 25 epochs from the random space dataset

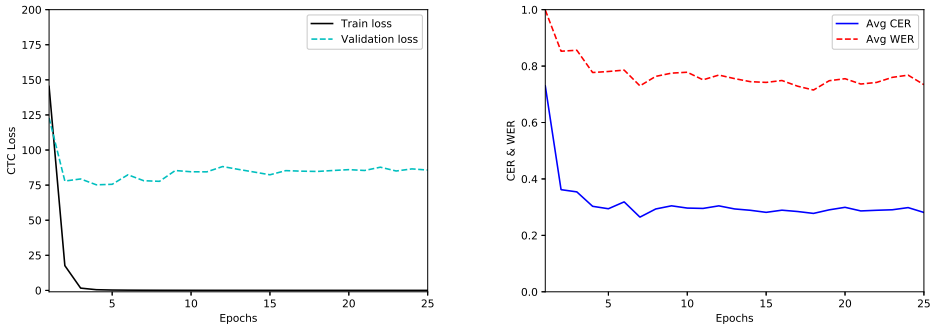

Fig. 9. WER, CER, Train & Val loss after 25 epochs from the OCRopy–linegen dataset

Another observation of these curves is that the overfitting occurs very quickly in both cases. The curves also indicate that training the model longer than for 5 or 10 epochs is not beneficial. Table 4 shows the results.

	Train loss	WER	CER	Edit distance
Random space	7.630	0.655	0.237	10.732
OCRopy–linegen	0.014	0.734	0.281	12.174

Table 4. Results of Strategy 2

6.4 Synthetic and Annotated Dataset Experiment – Strategy 3

This section describes the third strategy to train the model. First of all, we tried to verify our presumption from the previous section. The training course has shown that the training longer than 10 epochs is not profitable (see Figures 8 and 9). Therefore, we took the random space models based on 10, 25 and 50 epochs of synthetic training and then we added one page of real data for the additional 25 epochs of training (Figure 10). Our assumption was the more epochs of synthetic data training, the slower the learning occurs. Unfortunately, our assumption was not confirmed. But still, the model trained on 10 epochs of synthetic training is more suitable for additional real data training than the other ones and moreover, it has the lowest training demands.

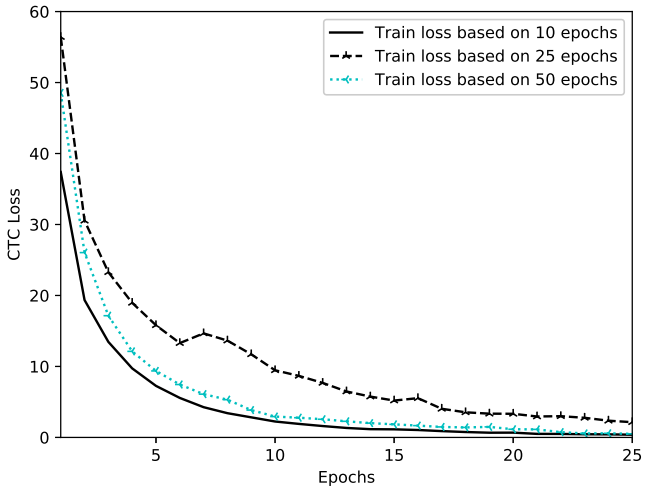

Fig. 10. Random space additional training based on 10, 25 and 50 epochs of synthetic training

In the light of a discussion above, we chose 10 epochs as a base for the additional training, because it reached the best results with a respect to the previous experiment.

Based on this 10 epochs, we took gradually 1–7 pages of the annotated train dataset and then we extended the training process of 25 epochs of additional training. The summary of results is in Figure 11. This experiment confirms an assumption that the more pages and epochs, the better the results.

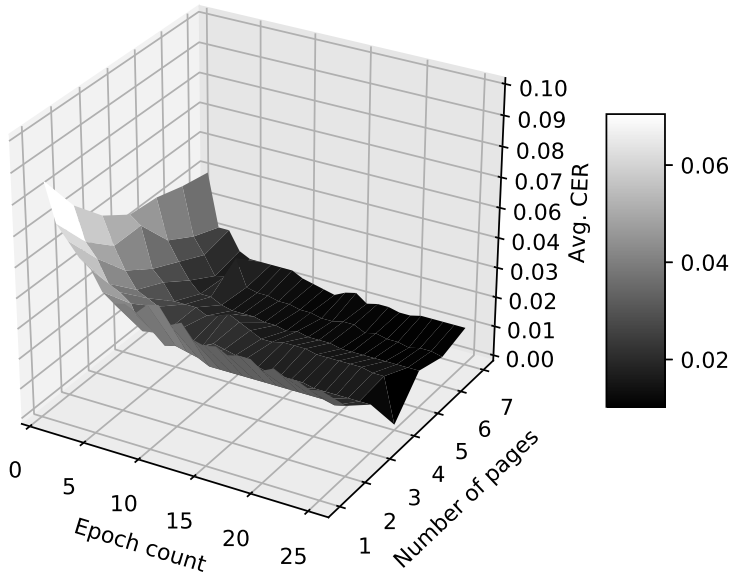

Fig. 11. Random space additional training

All other synthetic datasets show similar dependencies as Figure 11, so we present just a summary of the results in Table 5.

	Train loss	**WER**	**CER**	**Edit distance**
Strategy 3				
Random space	0.069	**0.065**	**0.012**	**0.428**
OCRopy–linegen	4.838	0.307	0.069	2.725
Strategy 2 – Random space	7.630	0.655	0.237	10.732
Strategy 1 – 100 epoch	**0.056**	0.068	0.014	0.464

Table 5. Summary of the results

7 Conclusions

In this work, we have studied and compared different methods of OCR training. Three learning strategies have been proposed for this purpose: the first one uses only a few annotated real pages, the second one considers only synthesized text lines and the third method combines generated data with the real ones. We have also proposed two generation methods which compose generated text-lines from real characters. We have compared the performance of these methods with OCRopy–linegen tool. We have further experimented with different white–space padding at the beginning of a line image.

We have shown that it is not possible to use only synthetic data to obtain sufficient OCR results. We have further reported that it can be sufficient to train the OCR models using only a small amount (about 7 pages) of real data to obtain a good recognition score. However, this process needs many training iterations and like we have indicated the implicit language model is limited. We have concluded that the best way is to combine both large synthetic and small real data to obtain the best recognition score at the low computation costs. We have also proven that our generation method obtains better results than the OCRopy–linegen tool. The last observation is that it is beneficial to use larger padding (about 60 pixels) at the beginning of the line image.

Acknowledgement

This work has been partly supported by Cross-border Cooperation Program Czech Republic - Free State of Bavaria ETS Objective 2014-2020 (project no. 211) and by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

References

1. Breuel, T.M.: The ocropus open source ocr system. In: Document Recognition and Retrieval XV, vol. 6815, p. 68150F. International Society for Optics and Photonics (2008)
2. Breuel, T.M.: High performance text recognition using a hybrid convolutional-lstm implementation. In: Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on, vol. 1, pp. 11–16. IEEE (2017)
3. Clausner, C., Pletschacher, S., Antonacopoulos, A.: Efficient ocr training data generation with aletheia. Proceedings of the International Association for Pattern Recognition (IAPR), Tours, France pp. 7–10 (2014)
4. Gaur, S., Sonkar, S., Roy, P.P.: Generation of synthetic training data for handwritten indic script recognition. In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, pp. 491–495. IEEE (2015)

5. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning, pp. 369–376. ACM (2006)
6. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: Advances in neural information processing systems, pp. 545–552 (2009)
7. Gupta, M.R., Jacobson, N.P., Garcia, E.K.: Ocr binarization and image preprocessing for searching historical documents. Pattern Recognition **40**(2), 389–397 (2007)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
9. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227 (2014)
10. Margner, V., Pechwitz, M.: Synthetic data for arabic ocr system development. In: Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on, pp. 1159–1163. IEEE (2001)
11. Otsu, N.: A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics **9**(1), 62–66 (1979)
12. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)
13. Sabir, E., Rawls, S., Natarajan, P.: Implicit language model in lstm for ocr. In: Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on, vol. 7, pp. 27–31. IEEE (2017)
14. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE transactions on pattern analysis and machine intelligence **39**(11), 2298–2304 (2017)
15. Simistira, F., Ul-Hassan, A., Papavassiliou, V., Gatos, B., Katsouros, V., Liwicki, M.: Recognition of historical greek polytonic scripts using lstm networks. In: Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, pp. 766–770. IEEE (2015)