

High-Performance OCR for Printed English and Fraktur using LSTM Networks

Thomas M. Breuel*, Adnan Ul-Hasan*, Mayce Al Azawi* and Faisal Shafait†

* Technical University of Kaiserslautern, 67663 Kaiserslautern, Germany; Email: {tmb, adnan, ali}@cs.uni-kl.de

† The University of Western Australia, Perth, Australia; Email: faisal.shafait@uwa.edu.au

Abstract—Long Short-Term Memory (LSTM) networks have yielded excellent results on handwriting recognition. This paper describes an application of bidirectional LSTM networks to the problem of machine-printed Latin and Fraktur recognition. Latin and Fraktur recognition differs significantly from handwriting recognition in both the statistical properties of the data, as well as in the required, much higher levels of accuracy. Applications of LSTM networks to handwriting recognition use two-dimensional recurrent networks, since the exact position and baseline of handwritten characters is variable. In contrast, for printed OCR, we used a one-dimensional recurrent network combined with a novel algorithm for baseline and x-height normalization. A number of databases were used for training and testing, including the UW3 database, artificially generated and degraded Fraktur text and scanned pages from a book digitization project. The LSTM architecture achieved 0.6% character-level test-set error on English text. When the artificially degraded Fraktur data set is divided into training and test sets, the system achieves an error rate of 1.64%. On specific books printed in Fraktur (not part of the training set), the system achieves error rates of 0.15% (Fontane) and 1.47% (Ersch-Gruber). These recognition accuracies were found without using any language modelling or any other post-processing techniques.

I. INTRODUCTION

Approaches to printed OCR can be broadly divided into segmented and unsegmented approaches. Many approaches work by segmenting input lines into characters or character candidates and then applying a classifier to the individual character candidates; the output of this is often a *recognition lattice*, which represents segmentation and recognition alternatives [1]. Another example of a segmentation-based OCR system is the open source Tesseract system [2]. Hidden Markov Models [3] are a common and successful example of unsegmented recognition. They have been applied to OCR as well [4].

Both kinds of systems are complex to develop and optimize. Segmentation-based OCR systems require carefully designed character segmentation methods, since segmentation errors generally lead to errors in the output. In our experience, segmentation errors are the limiting factor for the performance of segmentation-based OCR systems. Segmentation-based OCR systems also require careful estimation of segmentation and classification costs; in particular, the fact that segments of different length compete for being present on the best path through the recognition lattice makes estimating costs difficult and may require heuristic tuning of the cost functions. Hidden

Markov Models (HMMs) applied to OCR [5], [6] avoid many of the difficulties of segmentation-based OCR systems, but still require careful choices of model structures, and face similar issues in heuristic modifications of their cost functions to achieve overall good performance.

Convolutional neural networks have been applied in the past to handwriting recognition problems with some success [7]. However, in our experience and the experience of other groups, they have not yielded performance competitive with segmentation-based approaches for printed OCR. Likewise, there has been no report of competitive performance of Recurrent Neural Networks (RNN) [8] on OCR tasks.

Recurrent Neural Networks have had somewhat of a renaissance due to the Long Short Term Memory (LSTM) architecture [9], [10]. The LSTM architecture differs significantly from the architecture of previous recurrent neural networks [11], [12], [13] and appears to overcome many of the limitations and problems of those earlier architectures.

Recurrent Neural networks are considered good at context-aware processing and to recognize patterns occurring in time-series [8]. However, traditional recurrent neural networks have not shown competitive performance in large scale tasks like OCR and speech recognition, perhaps due to the *vanishing gradient problem* [14], [15]. The Long Short Term Memory [9] architecture was designed to overcome this problem. It is a highly non-linear recurrent network with multiplicative “gates” and additive feedback. Graves et al. [10] introduced bidirectional LSTM architectures for accessing context in both forward and backward directions. Both layers are then connected to a single output layer. To avoid the requirement of segmented training data, Graves et al. [16] used a forward backward algorithm to align transcripts with the output of the neural network.

For printed OCR, text lines come in many different sizes, yet the relative position and scale of characters is an important feature for distinguishing characters in Latin script and a variety of other scripts. Graves et al. [10] introduced a normalization process to scale the handwritten text-lines with varying writer speed and non-uniform skew. For printed OCR, different kinds of normalization appear to be required for good performance. Rashid et al. [17] divided the text-lines into ascender, descender and middle regions. In this paper, we follow a similar approach, but use shape models to make the



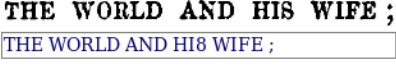
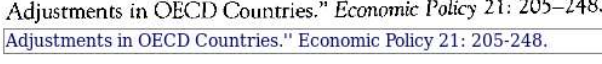


- (a)  stand in the window of his chamber in the morning,
- (b)  Travers on Constitutional Irritation.
- (c)  THE WORLD AND HIS WIFE ;
- (d)  Adjustments in OECD Countries." *Economic Policy* 21: 205–248.
- (e)  worte des Textes unter eine Composition und überließ es
- (f)  und Unanständigkeit die damalige fromme Musik gelit-

Fig. 1: Input/output from the LSTM-based OCR illustrating capabilities and errors. Generally, LSTM-OCR copes well with touching characters (a,b) and ligatures (e,f). Characters or styles that are rare in the training set are more prone to misclassification (c,e); this can be addressed with additional training data. The insertion in (f) is an artefact of the simple decoding process that turns the output of the neural network into a string; it is rare and probably can be eliminated with better decoding. Examples are taken from historical and modern Google book scans and Ersch-Gruber, not included in the training data.

detection of these regions more accurate and robust.

The goal of this paper is to demonstrate and evaluate the application of LSTM networks to OCR problems. This comprises a text line normalization step, followed by a direct application of LSTM networks to the normalized input. The paper presents evaluations on both standard Latin and Fraktur data sets.

II. TEXT-LINE NORMALIZATION

Text line normalization is an essential step in applying 1D LSTM networks to OCR, since 1D LSTM is not translationally invariant along the vertical axis. For Latin scripts, absolute position and scale along the vertical axis carries a significant amount of information and is essential for distinguishing a number of common characters. Taken together, these observations suggest that text line normalization combined with a 1D LSTM network could be a good choice for Latin script recognition.

In our system, text line size normalization is based on a trainable, shape based model. First a token dictionary is created from a collection of text files. This token dictionary further consists of two sub-dictionaries: 1) the shape dictionary, and 2) the geometrical models of text-lines (x-height and baseline information). The shape dictionary is created by clustering similar shapes from the given text-lines. k-means clustering was employed for this purpose. The geometrical models are created by updating the existing estimates of x-heights and baselines of individual character with the inputs from the new text-lines.

The normalization procedure for text line images is based on a dictionary composed of connected component shapes and associated baseline and x-height information. This dictionary is pre-computed based on a large sample of text lines with baselines and x-heights derived from alignments of the text line images with textual ground-truth, together with information about the relative position of Latin characters to the baseline and x-height. Note that for some shapes (e.g., p/P, o/O), the baseline and x-height information may be ambiguous; the information is therefore stored in the form of probability densities given a connected component shape. The connected components do not need to correspond to characters; they might be ligatures or frequently touching character pairs like “oo” and “as”.

When the baseline and x-height of a new text line need to be determined, the connected components are extracted from that text line and the associated probability densities for the baseline and x-height locations are retrieved. These densities are then mapped and locally averaged across the entire line, resulting in a probability map for the baseline and x-height lines across the entire text line. Maps of x-height and baselines of an example text line (Figure 2-(a)) are shown in Figure 2-(b) and (c) respectively. The resulting densities are then fitted with curves and used as the baseline and x-height for line size normalization. In line size normalization, the (possibly curved) baseline and x-height lines are mapped to two straight lines in a fixed size output text line image, with the pixels in between them rescaled using spline interpolation. The code for the normalization procedure has been released in open source form as part of the OCRopus system [1].

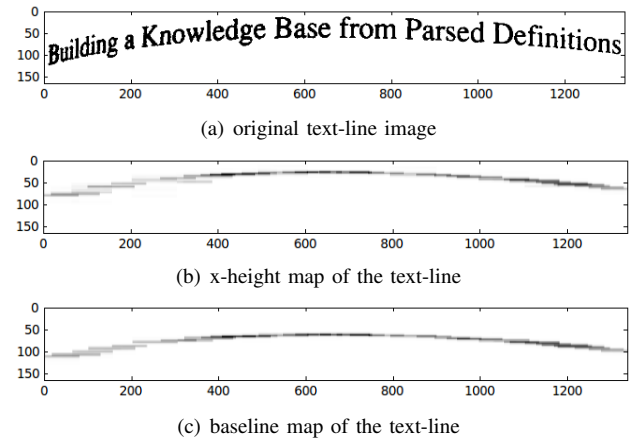


Fig. 2: Extraction of x-height and baseline of a text-line. (a) shows the original text-line image. (b) shows a map of predicted locations of a line at x-height, and (c) shows a map of predicted locations of the baseline. Note that the x-height is determined correctly for capital letters (although later processing is robust to occasional errors as well).

III. LSTM NETWORKS

For recognition, we use a 1D bidirectional LSTM architecture, as described in [10]. We did preliminary experiments on 2D LSTM architectures, as used for handwriting recognition [18], but found them not to be superior for printed OCR in those preliminary experiments. For all the experiments reported in this paper, we used a modified version of the LSTM library described in [19]. That library provides 1D and multidimensional LSTM networks, together with ground-truth alignment using a forward-backward algorithm (“CTC”, connectionist temporal classification; [16]). The library also provides a heuristic decoding mechanism to map the frame-wise network output onto a sequence of symbols. We have since reimplemented LSTM networks and forward-backward alignment from scratch and reproduced these results (our implementation uses a slightly different decoding mechanism). The configuration of the network and the number of weights mapping between and within layers is shown in Figure 3.

Training of the network proceeds by choosing text input lines randomly from the training set, performing a forward propagation step through the LSTM and output networks, then performing forward-backward alignment of the output with the ground-truth, and finally performing backward propagation. The intermediate results of this computation are illustrated in Figure 4.

IV. PERFORMANCE EVALUATION

To demonstrate the performance of the LSTM architecture on OCR tasks, we performed a number of evaluations. As error rate, we use the ratio of insertions, deletions, and substitutions relative to the length of the ground-truth, where the accuracy was measured at a character level. We have applied the LSTM-based system to printed English and Fraktur texts.

A. Printed English Recognition

For English input, we used the University of Washington (UW3) dataset [20], representing 1,600 pages of document

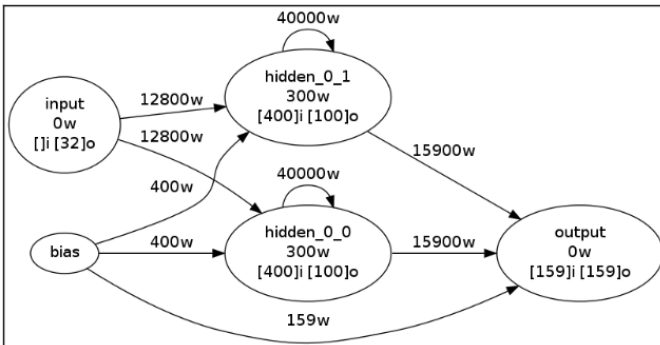


Fig. 3: Configuration of trained BLSTM network. There are 12,800 weights between input and hidden layers, while there are 15,900 weights between hidden layers and output layer. There are 40,000 weights for self-looping states at hidden layers. Input is a one dimensional with depth of 32.

images from scientific journals and other common sources. Text line images and corresponding ground-truth text were extracted from the data set using the layout ground-truth and transcriptions provided with UW3. Text lines containing mathematical equations were not used during either training or testing. Overall, we used a random subset of 95,338 text-lines in the training set and 1,020 text lines in the test set.

The text lines were normalized to a height of 32 in preprocessing step. Both left-to-right and right-to-left LSTM layers contain 100 LSTM memory blocks. The learning rate was set to $1e-4$, and the momentum was set to be 0.9. The training was carried out for one million steps (roughly corresponding to 100 epochs, given the size of the training set). Test set errors were reported every 10,000 training steps and plotted. The configuration of trained LSTM network is shown in Figure 3.

Our LSTM network was able to achieve 0.6% ($N = 50,632$) error on the test-set. There were a total of 313 errors and top confusions are *space* deletions (34 times), *period* confused with *comma* (25), *underscore* deletions (16), *period* confused with *underscore* (10), *comma* confused with *period* (6), *y* confused with *v* (5), *I* deletions (5) and *i* deletions (4).

To compare the results with other contemporary OCR systems, we used the same test-set. We used OCRopus [1], Tesseract [21] for comparison. The Tesseract system achieved a recognition error of 1.299% when run in line-wise mode with an English language model, and the previous segmentation-based OCRopus recognizer achieved 2.14%.

It should be noted that all these systems employ language modelling techniques to post-process the raw output, and in some cases other sophisticated techniques like font recognition and adaptivity. The LSTM network, on the other hand, achieved the results without any language modelling, adaptation, or post-processing. Running time is under a second for a regular text line on a modern desktop PC; our software parallelizes recognition over all available cores. Representative inputs and outputs are shown in Figure 1.

Table I presents the comparison of three OCR systems on the UW3 modified dataset.

B. Fraktur Recognition

We also applied the bidirectional LSTM architecture on Fraktur, a common historical German script similar to English Blackletter. Fraktur documents contain many touching characters, as well as ligatures. The training set was a fairly

TABLE I: Character error rates of the LSTM recognizer (without a language model or adaptation), the Tesseract [2] and the OCRopus segmenting recognizer [1]. Error rates are on the UW3 data set (English, modern print, $N=50,632$), the Fontane dataset (German, Fraktur, $N=8,988$) and the Ersch-Gruber dataset (German, Fraktur, $N=10,881$).

OCR System	English	Fontane	Ersch-Gruber
OCRopus-LSTM	0.6	0.15	1.37
Tesseract	1.3	0.9	1.47
OCRopus-lattice	2.14	-	-

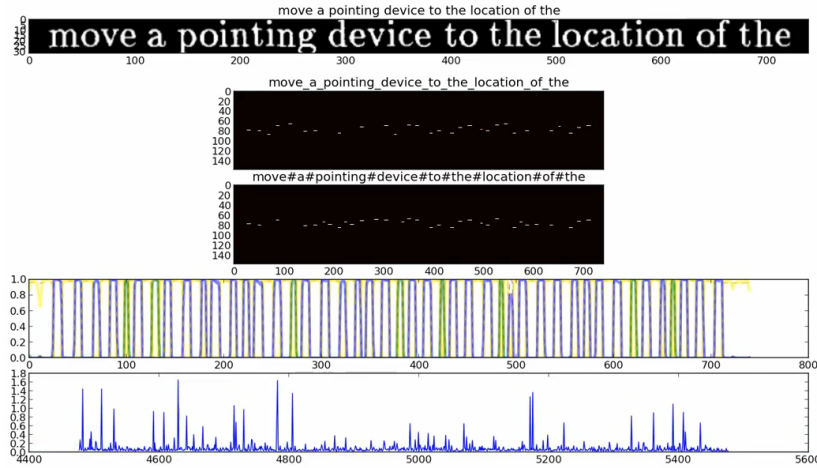


Fig. 4: An illustration of the training steps of the LSTM recognizer: (a) input text line together with ground-truth; we refer to each vertical slice of the input image as a *frame*, (b) frame-wise output from the LSTM network in image form, together with a decoded transcription, (c) frame-wise output from LSTM network after alignment with ground-truth using a forward-backward algorithm, (d) a graph of posterior probabilities for spaces (green), the top class (blue), and the reject class (yellow), (e) a graph of the total error signal used in the back-propagation step. Note that after only 5400 training lines, the network already recognizes this input perfectly. Interestingly, the error signal (e) has a very non-Gaussian distribution, with occasional large spikes. These spikes seem to correspond to internal reorganizations of the state space used by the LSTM model.

small set of about 20,000 text lines of mostly artificially generated characters. We did not evaluate test set error on the artificial training data, since that would not have been very informative. Instead, we tested performance on two books that we use as test cases: Theodor Fontane *Wanderungen durch die Mark Brandenburg* (a clean, high resolution scan) and the *Ersch-Gruber* encyclopedia (a noisy, lower resolution scan). Text in Antiqua (Latin) fonts was excluded from the evaluation. Since error rates were so low, we could quickly determine error rates and ground-truth using a spell checker and verifying any flagged words against the source image; since the texts contains few digits and little punctuation, this yields good error estimates. On randomly selected pages from Fontane representing 8,988 Fraktur characters, the error rate was 0.16%. On Ersch-Gruber, the error rate was 0.82% on randomly selected page images representing 10,881 Fraktur characters. These results are without a language model and without adaptation to the fonts found in these documents. Some examples of Fraktur errors are shown in Figure 1.

Like printed English, recognition results for Fraktur (for both Fontane and Ersch-Gruber) were compared with other OCR systems. The Tesseract system applied to these inputs yielded error rates of 0.898% (Fontane) and 1.47% (Ersch-Gruber), using a German dictionary and font adaptations.

V. CONCLUSIONS

The results presented in this paper show that the combination of line normalization and 1D-LSTM yields excellent OCR results for both Latin/Antiqua OCR and Fraktur OCR. Our benchmarks suggest that error rates for LSTM-based OCR without a language model are considerably lower than

those achieved by segmentation-based approaches (OCRopus, Tesseract) and HMM-based approaches, even with language models. We noted that the approach, treating the input text line like a sequence of “frames” over time is related to HMM-based approaches [4] and Document Image Decoding [23], but the LSTM approach has only three parameters, the input size, the number of memory units, and the learning rate.

A common and valid concern with OCR systems based on machine learning or neural network techniques is whether they will generalize successfully to new data [2]. That is, in machine learning, we would ordinarily determine the error rate of a system by taking a data set, dividing it into training and test sets, train the system on the training set and evaluate on the test set. However, our own experience, as well as that of other practitioners has been that such estimates do not reflect well the real-world performance of OCR systems, since OCR data is, in a sense, data whose distribution has “long tails”; that is, new text that an OCR system will be applied to often differs significantly from all training data. There is currently no standard testing procedure or widely used data set in the document analysis community to address this problem. However, there are several indications that LSTM-based approaches generalize much better to unseen samples than previous machine learning methods applied to OCR. One such indication is the excellent performance of the LSTM-based Fraktur recognizer when trained on artificial training data and tested on scanned Fraktur books. In addition, during LSTM training, we often observe very low error rates long before even one epoch of training has been completed, meaning that there has likely not been an opportunity to “overtrain”. In addition, LSTM-based systems have been found to generalize

well to novel data by other practitioners [9]. We will be trying to address this with further benchmarks in the future.

An unsatisfactory aspect of LSTM-based models is that they are “black box” models; that is, little is known why or how they achieve their excellent performance. Other users of LSTM models have shown that such models are capable of learning a variety of formal languages, and learn them better than previous approaches, but how that is achieved still has not been explained in detail. We are currently investigating and analyzing how these models work internally and how their operation relates to HMM and segmentation-based approaches.

The normalization procedure used for recognition is complex and requires a separate training step. This both complicates the training and limits the method to scripts in which connected components can give us useful information about the geometry of the printed text. We have therefore started exploring simpler, script and connected-component independent methods for line normalization.

Overall, although the method itself can certainly be refined and the benchmarking ought to be strengthened in future work, we believe that these results show that LSTM has a strong potential of outperforming existing approaches to OCR. In addition, the absence of a need for a separate language modelling step, of the need to heuristically choose “HMM structures” or “input features” or “character segmentation algorithms” is a big advantage in being able to cover new languages and scripts.

Training of 1D LSTM models for OCR is considerably simpler than previous training methods, since all training is carried out in terms of text line images and transcriptions. In addition, training from artificially generated data is highly successful, as the results from Fraktur recognition show. This has enabled the creation of a new labeling tools (*ocropus-gedit*) that allows new ground-truth to be generated and transcribed using a simple, single-page web based interface, and a new tool (*ocropus-linegen*) for generating large amounts of artificial training data from TrueType fonts and representative text; these will be described elsewhere. In terms of parameters, beyond text line normalization, all that is required is choosing the input size (currently 32) and the number of internal state units (currently 100). In these experiments, we did not optimize them. Preliminary results suggest that using larger input sizes and more internal state units results in faster learning and possibly additional reductions in error rates on UW3; we will be exploring the parameter space more thoroughly in future work. We have focused on 1D LSTM networks in this work because preliminary experiments suggested that they were faster and yielded better results; it is possible that some 2D LSTM variants might achieve better performance, and we will investigate this. Techniques like combining multiple models, word recognition, adaptivity, and language modeling can be incorporated into LSTM-based OCR, and we are actively exploring these.

An implementation of LSTM and line normalization methods is being released in open source form [24], allowing other practitioners to test, evaluate, and reuse the methods described

in this paper, as well as to easily train text recognizers for novel scripts.

REFERENCES

- [1] T. M. Breuel, “The OCRopus open source OCR system,” in *DRR XV*, vol. 6815, Jan. 2008, p. 68150F.
- [2] R. Smith, “History of the Tesseract OCR engine: what worked and what didn’t,” in *DRR XX*, San Francisco, USA, Feb. 2013.
- [3] L. R. Rabiner, “A tutorial on Hidden Markov Models and selected applications in speech recognition,” *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.
- [4] Z. Lu, R. M. Schwartz, P. Natarajan, I. Bazzi, and J. Makhoul, “Advances in the BBN BYBLOS OCR System,” in *ICDAR*. IEEE Computer Society, 1999, pp. 337–340.
- [5] S. España-Boquera, M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez, “Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 767–779, Apr. 2011.
- [6] Y. Bengio, Y. Lecun, C. Nohl, and C. Burges, “LeRec: A NN/HMM hybrid for on-line handwriting recognition,” *Neural Computation*, vol. 7, no. 6, pp. 1289–1303, Nov. 1995.
- [7] Y. Bengio, Y. LeCun, and D. Henderson, “Globally Trained Handwritten Word Recognizer Using Spatial Representation, Convolutional Neural Networks, and Hidden Markov Models,” in *NIPS*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds. Morgan Kaufmann, 1993, pp. 937–944.
- [8] A. W. Senior, “Off-line Cursive Handwriting Recognition using Recurrent Neural Networks,” Ph.D. dissertation, England, 1994.
- [9] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [10] A. Graves, M. Liwicki, S. Fernandez, Bertolami, H. Bunke, and J. Schmidhuber, “A Novel Connectionist System for Unconstrained Handwriting Recognition,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, May 2008.
- [11] J. L. Elman, “Finding Structure in Time,” *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.
- [12] W. Maass, T. Natschlager, and H. Markram, “Real-time computing without stable states: a new framework for neural computation based on perturbations,” *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, Nov. 2002.
- [13] H. Jaeger, “Tutorial on Training Recurrent Neural Networks, Covering BPTT, RTTL, EKF and the ‘Echo State Network’ approach,” Sankt Augustin, Tech. Rep., 2002.
- [14] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” in *A Field Guide to Dynamical Recurrent Neural Networks*, S. C. Kremer and J. F. Kolen, Eds. IEEE Press, 2001.
- [15] Y. Bengio, P. Smirard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Trans. on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
- [16] A. Graves, S. Fernandez, F. Gomes, and J. Schmidhuber, “Connectionist Temporal Classification: Labeling Unsegmented Sequence Data with Recurrent Neural Networks,” in *ICML*, Pennsylvania, USA, 2006, pp. 369–376.
- [17] S. F. Rashid, F. Shafait, and T. M. Breuel, “Scanning Neural Network for Text Line Recognition,” in *DAS*, Gold Coast, Australia, Mar. 2012.
- [18] A. Graves, “Offline Arabic Handwriting Recognition with Multidimensional Recurrent Neural Networks,” Springer, 2012, pp. 297–313.
- [19] —, “RNNLIB: A recurrent neural network library for sequence learning problems.” [Online]. Available: <http://sourceforge.net/projects/rnnl>
- [20] I. T. Phillips., “User’s reference manual for the UW English/Technical Document Image Database III,” Seattle University, Washington, USA, Tech. Rep., 1996.
- [21] R. Smith, “An Overview of the Tesseract OCR Engine,” in *ICDAR*, 2007, pp. 629–633.
- [22] “ABBY.” [Online]. Available: http://www.abby.com/recognition_server
- [23] G. E. Kopec and P. A. Chou, “Document Image Decoding Using Markov Source Models,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 602–617, 1994.
- [24] “OCROPUS - Open Source Document Analysis and OCR system.” [Online]. Available: <https://code.google.com/p/ocropus>