2018 IEEE 2nd International Workshop on Arabic and Derived Script Analysis and Recognition (ASAR)

How To Efficiently Increase Resolution in Neural OCR Models

Stephen Rawls, Huaigu Cao, Joe Mathai, Prem Natarajan
Information Sciences Institute
University of Southern California
Marina Del Rey, California 90007
Email: srawls@isi.edu, hcao@isi.edu, jmathai@isi.edu, pnataraj@isi.edu

***Abstract*—Modern CRNN OCR models require a fixed line height for all images, and it is known that, up to a point, increasing this input resolution improves recognition performance. However, doing so by simply increasing the line height of input images without changing the CRNN architecture has a large cost in memory and computation (they both scale $O(n^2)$ w.r.t. the input line height).**

We introduce a few very small convolutional and max pooling layers to a CRNN model to rapidly downsample high resolution images to a more manageable resolution before passing off to the "base" CRNN model. Doing this greatly improves recognition performance with a very modest increase in computation and memory requirements. We show a 33% relative improvement in WER, from 8.8% to 5.9% when increasing the input resolution from 30px line height to 240px line height on OpenHART/MADCAT Arabic handwriting data.

This is a new state of the art result on Arabic handwriting, and the large improvement from an already strong baseline shows the impact of this technique.

I. Introduction

In recent years, especially after the development of the CTC loss function by Graves [1], neural approaches to OCR and handwriting recognition have dominated the literature [2], [3], [4], [5], [6], with people either using a convolutional neural network (CNN) followed by a 1-D recurrent neural netowrk (RNN), i.e. a CRNN model, as in [2], [3] or using a 2-D Long Short Term Memory (LSTM) model as in [5], [6]. See [3] for a brief history on the evolution of neural approaches to OCR. Most systems from ICDAR17 used such neural models.

In the past when HMM systems were popular, some groups used resolution-independent percentile based hand-crafted features [7], [8], [9], [10]. These features could be extracted from images in their native resolution without needing to resample line images to a fixed line height. However with current neural approaches, primarily due to the fixed input size for 1-D LSTMs and fully connected layers (which are needed to transform an LSTM output into the dimension of the output alphabet), all input images must be transformed to a fixed line height. In [2] they chose to use a line height of 32px, while in [3] we chose a line height of 30px (but experimented with 40px and 60px as well), and a casual inspection of GitHub shows that most repositories (including Tesseract [11]) use a line height of 32px.

In this work we extend the system from [3] and report new state-of-the-art results for both Arabic handwriting recognition and English handwriting recognition. We do this by increasing the resolution of input images from a line height of 30px to 240px, an 8-fold increase.

We chose to focus on the resolution of line images after examining recognition performance on both the IAM English handwriting [12] and OpenHART/MADCAT Arabic handwriting [13] datasets, where we determined that a large number of recognition errors in our system were due to low quality input images. We further found that the dominant cause of low quality input images was degradations from downsampling a high resolution image down to a target line height of 30px. The IAM dataset has an average line height of 120px and the MADCAT dataset has an average line height of 230px, which shows that on both datasets our prior system was downsampling by more than a factor of 4 on average for IAM, and a factor 8 on average for MADCAT. While in most cases this loss of resolution yields a clear image that our CRNN system is able to accurately decode, in a non-trivial number of cases we found that standard downsampling techniques severely degrade image quality in a way difficult for a CRNN to recover from, especially as these degraded images appear very different from the majority of training samples.

Based on this, we wanted to experiment with line heights of 120px and 240px. At 120px we determined most images look fine even with downsampling, and at 240px, most images do not need to be downsampled at all. Naively increasing the input resolution of images without changing our base CRNN model from [3] would be computationally infeasible however. We explore this in closer detail in Section II, but roughly speaking because of the $O(n^2)$ scaling of both memory and time requirements, naively increasing line height by a factor of 4 or 8 would result in scaling memory and speed by roughly factors of 16 or 64, and given that the baseline model takes 2 GPUs and 2 days to train, increasing the input resolution quickly becomes infeasible.

II. Model

We take as our baseline model the CRNN model described in [3], and shown in Fig 1. To this, we prepend a sequence of small convolutional and pooling layers to quickly downsample the input image to a height of 30px. The convolutional layers are $3x3$ and have only 16 output filter maps, so they contribute only a small cost in memory and computation requirements.

Fig. 1: Our End-to-End OCR Model: With CNN and LSTM layers

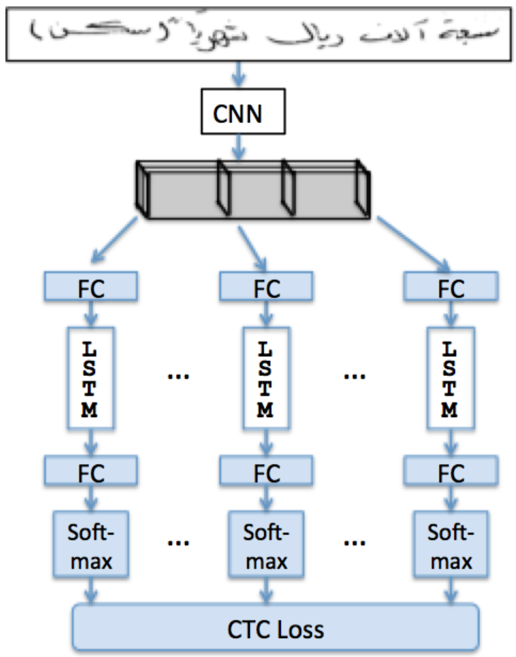


TABLE I: Layer-wise description of CRNN Model for 240px Input

Layer #	Type	Configuration
Start of rapid downsampling		
1	Convolution	#maps:16, k:3x3
2	Max Pooling	k:2x2, stride:2x2
3	Convolution	#maps:16, k:3x3
4	Max Pooling	k:2x2, stride:2x2
5	Convolution	#maps:16, k:3x3
6	Max Pooling	k:2x2, stride:2x2
Start of "base" CRNN		
7	Convolution	#maps:64, k:3x3
8	Convolution	#maps:64, k:3x3
9	Fractional Pooling	0.7x0.5
10	Convolution	#maps:128, k:3x3
11	Convolution	#maps:128, k:3x3
12	Fractional Pooling	0.7x0.5
13	Convolution	#maps:256, k:3x3
14	Convolution	#maps:256, k:3x3
15	Convolution	#maps:256, k:3x3

All convolution layers have a stride of 1x1 and padding of 1x1

See Table I for a layer-by-layer description of our model configured for a line height of 240px.

Below we briefly address memory and computational requirements of our model.

A. Memory Complexity

In general, memory in neural networks is consumed in the following places: 1) parameters; 2) activations; 3) gradients (for both parameters and activations); 4) potentially some gradient history for certain types of optimizers; and finally 5) overhead. While it is difficult to compute overhead exactly (overhead can come from extra memory needed to make convolution faster, memory used by cudnn to benchmark operations under various conditions, memory used by neural network frameworks for bookkeeping, etc), all the other types of memory use can be computed precisely.

Note that recurrent modules like LSTMs need to store activations and gradients at each timestep to support back propagation. Thus wide images have a ripple effect through the network, causing increased memory in both the CNN and LSTM layers.

We perform a careful accounting of the memory requirements for each layer in our network, and sum the memory for each layer to give total memory usage. See Table II for the results.

When first attempting to increase input resolution to our model we were running into memory limits, and we noticed that while 99% of images in MADCAT have a width of 600px or smaller when resized to a target height of 30px, there were a small number of excessively long line images, with the widest image being 1175px wide at a target line height of 30px. We noticed that we were in essence using twice as much memory as we otherwise would be, just to accommodate a small fraction (1%) of images. In later experiments we decided to ignore these large images in training to reduce overall memory consumption. In Table II we report our main numbers assuming a width of 600px given that is the maximum width of our reduced training set, but we also report parenthetically the memory required had we decided to process all images in MADCAT.

We note briefly that these 1% of very long images could be handled in other ways besides throwing them away. Conceivably we could break them in half, although we would need to use an automatic method to split the ground truth transcription appropriately. As the MADCAT training data is already very large, and we are only throwing out 1% of the data, we do not explore this possibility.

As can be seen in Table II, memory requirements for the naive upsampling approach quickly balloon out of control, requiring 112 GB of GPU memory for 240px high inputs. However, our rapid-downsampling method requires substantially less memory, requiring only 10.6 GB of GPU memory for the same 240px high inputs. We note that the naive approach wouldn't even fit on an 8-GPU machine with 12GB of memory per GPU (because 112 GB / 8 = 14 GB). Thus it is not even feasible to use the naive approach for high resolution images with our lab's resources.

We believe that if we had very high resolution images, we could perform even faster down sampling, for example by using strided and/or dilated convolutions [14] to reduce the input resolution by a factor greater than 2 at each step, or using more aggressive fractional max pooling [15]. However, given that the average line height in MADCAT is 230px, and we saw a very small improvement from 120px to 240px, we currently see no need to experiment with higher resolution inputs than 240px.

B. Time Complexity

Generally speaking, compute time for convolutional layers scales $O(hw)$ where h is the height of the input image and w is the width of the image, but in our work we assume that h is varying and w is set to keep a fixed aspect ratio, so we can consider time to scale as $O(h^2)$. Additionally, compute time for the LSTMs scale $O(T)$ where T is the number of time steps. For the naive upsampling case, T depends linearly on the input line height, so the LSTM compute scales $O(h)$ in relation to the input height. In the rapid-downsampling case, however, T is a model constant that does not depend on the input line height (only on the maximum image width in a given dataset, at a fixed model-specified target line height). The dominating factor, however, is the $O(h^2)$ dependency of the convolutional layers.

We would expect, therefore, that naively changing the input line height from 30px to 240px would cause the computational requirements for the CNN portion to increase roughly 64x, and the computational requirements for the LSTM portion to increase roughly 8x.

We can measure this empirically to find the total speed of our network as the input size varies. See Table III for the speed per iteration at varying input sizes. We find that as the line height increases from 30px to 240px (an 8x increase), the empirically measured time per iteration increases from $0.18s$ to $8.8s$, approximately a 50x increase, which fits with our understanding of a portion of the network increasing roughly 64x while another portion increases roughly 8x.

Since the baseline model from [3] takes 2 days of training time for a 30px input image, a 50x increase in computation time is not practical. Moreover, in the naive case, we cannot even fit a minibatch of 32 samples on 8 GPUs at once, so we would need to train with a smaller batch size, or accumulate gradients over two iterations and double the time to train, causing a 100x increase in total.

However, because the rapid downsampling layers are very small (they only have 16 output filters), and there is a 2x2 max pooling layer after each convolution, the extra time required for the rapid downsampling layers is almost negligible. When making the same increase to input height as before, the empirically measured iteration time for the rapid-downsampling method increses from $0.18s$ to $0.41s$, or a 2.3x increase. This can be completely hidden by using more GPUs, but also it is feasible to simply wait four days to train a model instead of two. Thus, while the naive method of increasing input resolution is infesible for both memory and time constraints, the rapid downsampling method is very practical.

TABLE II: Effect of Line Height on Memory

Input Resolution Line Height (px)	Naive Upsampling Memory (GB)	Rapid-Downsampling Memory (GB)
30	2.9 (5.6)	-
60	8.9 (17.4)	3.3 (6.3)
120	30.7 (59.9)	4.7 (9.2)
240	112.7 (220.6)	10.6 (20.8)

Calculation of memory use under different input sizes. Computed for batch size of 32. Parenthetical numbers represent memory usage for longest input sizes in MADCAT. See text for more details.

TABLE III: Effect of Line Height on Speed

Input Resolution Line Height (px)	Naive Upsampling Speed (s)	Rapid-Downsampling Speed (s)
30	0.18	-
60	0.56	0.20
120	2.0	0.27
240	8.8	0.41

Emprical measurement of speed per iteration, with batch size of 32. Computed using 2 NVidia 1080ti GPUs and averaged over 100 iterations. For large input size and naive umpsampling, we used a smaller batch size to fit into memory, and scaled the time up accordingly.

III. Data Sets

We show results on both Arabic and English handwriting datasets. A description of the datasets is below, along with examples of downsampled images. The examples shown use bicubic interpolation for downsamping, but results look similar for linear and Lanczos interpolation as well. For both IAM and MADCAT we specifically choose examples with a poor quality after downsampling to 30px line height. While a non-trivial number of images in each dataset exhibit this behavior, it should be noted that the majority of images look quite clear even after downsampling to 30px line height.

A. IAM English Handwriting

The IAM dataset consists of approximately 10k grayscale line images of English handwriting (6.1k in train, 1.8k in validation, 1.8k in test), containing 500 distinct writers with no writer overlap between train/validation/test sets [12].

The average line height for IAM images is 122px, and less than 0.1% of images have a line height greater than 300px. See Fig 2 to see the effect of downsampling on an example IAM image.

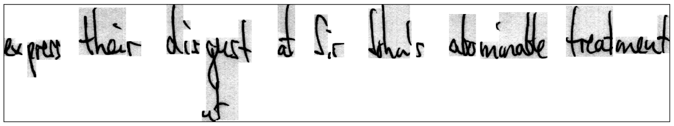
B. MADCAT Arabic Handwriting

The MADCAT/OpenHART dataset consists of training and evaluation data from multiple phases of the MADCAT DARPA program. The training data is available through the Linguistic Data Consortium (LDC) at the University of Pennsylvania, however the official evaluation data remains unavailable to the public.

The authors have approached NIST and LDC about obtaining the official MADCAT evaluation data, but as of the paper submission deadline have so far been unsuccessful.

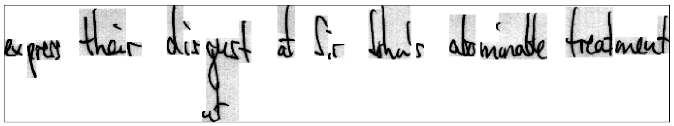
The official MADCAT train and evaluation split contains 41,747 page images in the training set, 470 page images in the development set, and 633 page images in the evaluation set [13]. With approximately 18 lines per page, this works out to roughly 750k line images in the training set, 8.4k line images in the development set, and 11.3k line images in the evaluation set.

Because the official MADCAT evaluation dataset remains unavailable, the authors were required to create their own



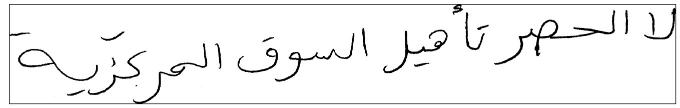
(a) Original image resolution (304px line height)

(b) Resized to 30px line height



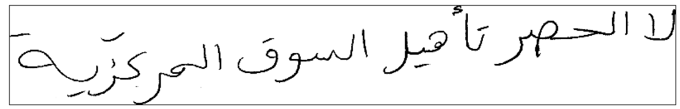
(c) Resized to 120px line height

Fig. 2: Example image from IAM showing an image in its original resolution, resized to 30px high, and resized to 120px high. The 30px high image is degraded while the 120px image is clear.



(a) Original image resolution (446px line height)

(b) Resized to 30px line height



(c) Resized to 120px line height

Fig. 3: Example image from MADCAT showing an image in its original resolution, resized to 30px high, and resized to 120px high. The 30px high image is degraded while the 120px image is clear.

train/dev/test split out of the MADCAT training data available through LDC. This was done by randomly choosing pages for train/dev/test, and enforcing a fair division by removing all copies of test-set passages written by other writers from being included in either the training or development sets. (This is necessary because each passage appears multiple times in the dataset, written by different authors; as it turned out, our final score was not affected by removing these passages). Our constructed test set contains approximately 667k line images of Arabic handwriting (648k in train, 9.1k in validation, 10.1k in test), containing 24 distinct writers. If we are unable to obtain a public version of the MADCAT evaluation set, then we will make public the page ID's for our train/dev/test split.

We note in passing that even though our results are not directly comparable to previously published work, since we lack the official evaluation set, they should still be considered strong. On the issue of comparing to past work, the difference in performance we show is large enough (5.9% vs 17.0%) to believe that our system would be state-of-the-art on the official evaluation set, as it is on IAM where we do have the official evaluation set. Additionally, the primary aim of this work is to show the improvement from using high-resolution input images, which can be clearly seen on our test set.

The line height statistics for MADCAT are that the average line height of images is 230px high, and 10% of images have a line height greater than 300px. See Fig 3 to see the effect of downsampling on an example MADCAT image.

IV. Experiments

For both IAM and MADCAT datasets we use the same CRNN architecture. The hyperparemeters are as follows: we use a CNN as described in Table I (with the Rapid-Downsampling layer modified to include only enough CNN/Max-pooling layers to downsample the input to 30px in height, depending on the input height), we use 3 bidirectional LSTM layers each of size 512 hidden units, we use a fully connected layer to reduce the dimension of each CNN feature slice to 128 dimensions prior to feeding it into the LSTM layers, and we train the model using the ADAM optimizer with an initial learning rate of 1e-3, reducing the learning rate to 1e-4 once training plateaus.

A. Baselines

We compare our current results to our previously reported results from [3], with a few small modifications.

For MADCAT, we report the result from [3], but also an improved baseline resulting from a slightly larger model (512 hidden LSTM units instead of 256) and from proper handling of embedded left-to-right regions within the right-to-left Arabic text (in our updated baseline, and later systems, we now undo the Unicode bidirectional algorithm so that the model is trained on visual-ordered text, as opposed to reading-ordered text). Our gains from increasing resolution should be compared against the updated baseline, as opposed to our previously published baseline. All our MADCAT numbers are using greedy argmax decoding without any language model. Finally, for MADCAT we use the NIST OpenHART evaluation tools [13] for computing CER/WER, which involve normalizing certain diacritics.

For IAM, we report our system from [3], but with updated error rates based on a bug fix associated with the WFST language model we used for [3]. All our IAM results are using a 5-gram word-based language model using data from English Gigaword and a sanitized version of the Brown and LOB Corpus [16].

TABLE IV: CNN+LSTM Final Results

Dataset	Model	CER (%)	WER (%)
MADCAT	RWTH OpenHart13 [17]	4.5	17.0
	Ours, from [3]	4.0	13.8
	Ours, 30px updated baseline	2.8	8.8
	Ours, 60px Rapid Downsampling	2.0	7.5
	Ours, 120px Rapid Downsampling	1.5	6.1
	Ours, 240px Rapid Downsampling	**1.5**	**5.9**
IAM	Doetsch et al [18]	5.0	13.2
	Ours, from [3]	5.3	12.7
	Ours, 120px Rapid Downsampling	**4.1**	**10.4**

B. Increased Resolution

For MADCAT we run experiments using an input resolution of 60px, 120px and 240px line height. Given that the average line height in MADCAT is 220px, and most images are less than 240px, we do not explore input resolutions higher than 240px.

For IAM we run experiments using an input resolution of 120px to show we also get new state-of-the-art results on English handwriting with this technique. We don't consider a 240px line height model because the average line height is only 122px, so we believe there is no point in artificially upsampling images further.

From Table IV it is clear that increasing the input resolution gives a dramatic improvement to recognition performance, giving us an overall improvement from 8.8% WER to 5.9% WER for MADCAT and from 12.7% WER to 10.4% WER for IAM.

We do not compare results against naively upsampling images without our Rapid Downsampling addition to our CRNN. While it is still computationally feasible to train such a model with 60px line height, it would take an 8 GPU machine over a week to complete it, compared to about a day and a half on 2 GPUs for all our Rapid Downsampling models. Also, it is not computationally feasible to run the experiment for higher resolutions, like 120px or 240px, meaning the biggest gains we show are only possible with our method.

V. Conclusion

We have shown that it is feasible to train CRNN models with very high input resolutions using a releatively simple trick, and that this results in a very healthy gain on already strong state-of-the-art systems, reducing error rates by an absolute 2.9% WER for MADCAT and an absolute 2.3% WER for IAM. Not only does this technique make such training feasible, but the added computational time is almost negligible, making it a clear win.

References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 369–376. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143891

[2] B. Shi, X. Bai, and C. Yao, "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*, vol. 39, no. 11, pp. 2298–2304, Nov 2017.

[3] S. Rawls, H. Cao, S. Kumar, and P. Natarajan, "Combining convolutional neural networks and lstms for segmentation-free ocr," in *Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition*, ser. ICDAR '17. Kyoto, Japan: IEEE Computer Society, 2017.

[4] S. Rawls, H. Cao, E. Sabir, and P. Natarajan, "Combining deep learning and language modeling for segmentation-free ocr from raw pixels," in *Proceedings of the 2017 1st IEEE International Workshop on Arabic Script Analysis and Recognition*, ser. ASAR '17. Nancy, France: IEEE Computer Society, 2017.

[5] T. Bluche, J. Louradour, M. Knibbe, B. Moysset, F. Benzeghiba, and C. Kermorvant, "The a2ia arabic handwritten text recognition system at the openhart2013 evaluation," in *International Workshop on Document Analysis Systems (DAS)*, 2014.

[6] B. Moysset, T. Bluche, M. Knibbe, M. F. Benzeghiba, R. Messina, J. Louradour, and C. Kermorvant, "The a2ia multi-lingual text recognition system at the maurdor evaluation," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014.

[7] P. NATARAJAN, Z. LU, R. SCHWARTZ, I. BAZZI, and J. MAKHOUL, "Multilingual machine printed ocr," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 15, no. 01, pp. 43–63, 2001. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0218001401000745

[8] J. Makhoul, R. Schwartz, C. LaPre, and I. Bazzi, "A script-independent methodology for optical character recognition," *Pattern Recognition*, vol. 31, no. 9, pp. 1285–1294, 1998.

[9] P. Natarajan, Z. Lu, R. Schwartz, I. Bazzi, and J. Makhoul, "Hidden markov models." River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2002, ch. Multilingual Machine Printed OCR, pp. 43–63. [Online]. Available: http://dl.acm.org/citation.cfm?id=505741.505744

[10] P. Natarajan, S. Saleem, R. Prasad, E. MacRostie, and K. Subramanian, "Multi-lingual offline handwriting recognition using hidden markov models: A script-independent approach," in *Proceedings of the 2006 Conference on Arabic and Chinese Handwriting Recognition*, ser. SACH'06. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 231–250. [Online]. Available: http://dl.acm.org/citation.cfm?id=1792262.1792276

[11] R. Smith and G. Inc, "An overview of the tesseract ocr engine," in *Proc. 9th IEEE Intl. Conf. on Document Analysis and Recognition (ICDAR*, 2007, pp. 629–633.

[12] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2002. [Online]. Available: http://dx.doi.org/10.1007/s100320200071

[13] A. Tong, "Nist openhart'13 evaluation: Overview and results," 2013.

[14] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015. [Online]. Available: http://arxiv.org/abs/1511.07122

[15] B. Graham, "Fractional max-pooling," *CoRR*, vol. abs/1412.6071, 2014. [Online]. Available: http://arxiv.org/abs/1412.6071

[16] A. D. Beale, "Grammatical analysis by computer of the lancaster-oslo/bergen (lob) corpus of british english texts," in *Proceedings of the 23rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '85. Stroudsburg, PA, USA: Association for Computational Linguistics, 1985, pp. 293–298. [Online]. Available: http://dx.doi.org/10.3115/981210.981246

[17] M. Hamdani, P. Doetsch, M. Kozielski, A. E.-D. Mousa, and H. Ney, "The rwth large vocabulary arabic handwriting recognition system," in *Proceedings of the 2013 27th Brazilian Symposium on Software Engineering*, ser. SBES '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 111–115. [Online]. Available: http://dx.doi.org/10.1109/DAS.2014.61

[18] M. Kozielski, P. Doetsch, and H. Ney, "Improvements in rwth's system for off-line handwriting recognition," in *2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28, 2013*, 2013, pp. 935–939. [Online]. Available: http://dx.doi.org/10.1109/ICDAR.2013.190