READ
RECOGNITION & ENRICHMENT
OF ARCHIVAL DOCUMENTS

D7.9
HTR Engine Based on NNs P3

Optimizing speed and performance - HTR+

Johannes Michael, Max Weidemann, Roger Labahn
URO

Distribution: http://read.transkribus.eu/

READ
H2020 Project 674943

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 674943

Project ref no.	H2020 674943
Project acronym	READ
Project full title	Recognition and Enrichment of Archival Documents
Instrument	H2020-EINFRA-2015-1
Thematic priority	EINFRA-9-2015 - e-Infrastructures for virtual research environments (VRE)
Start date/duration	01 January 2016 / 42 Months

Distribution	Public
Contract. date of delivery	31.12.2018
Actual date of delivery	03.12.2018
Date of last update	21.12.2018
Deliverable number	D7.9
Deliverable title	HTR Engine Based on NNs P3
Type	Demonstrator
Status & version	Final
Contributing WP(s)	WP7
Responsible beneficiary	URO
Other contributors	
Internal reviewers	Joan Andreu Sanchez (UPVLC)
Author(s)	Johannes Michael, Max Weidemann, Roger Labahn
EC project officer	Martin Majek
Keywords	BLSTM, CNN, TCN, GPU, competition, image-preprocessing, data augmentation, training strategies

Contents

Executive Summary

The third year's deliverable describes that the performance of a handwritten text recognition (HTR) model is not only affected by its structure but also by many other factors. This includes, among others, the quality, variety and amount of input images and the use of hardware-optimized software. Those changes led to a faster and better HTR model measured in terms of character error rate (CER) compared to the architecture presented in deliverable D7.8 of year two.

1 Introduction

Last year we introduced Google's Deep Learning library TensorFlow [1] which enabled us to build deep neural networks (NNs) fast and easily. The proposed architecture resulted in a performance gain of more than 50 % compared to the sequential processing recurrent neural network (SPRNN) architecture of year one. Further adaptations to the NN, a simplified preprocessing pipeline, diversified data augmentation and modified training strategies have brought additional improvements and are described in section 2. The HTR models reach new peak accuracies in terms of CER, while simultaneously running faster due to optimized GPU utilization. We extended the performance gain up to 80 % compared to year one (CER) and the training of an NN is up to 10 times faster than before. Experiments were evaluated on three different datasets: Staatsarchiv des Kantons Zürich (StAZH) collection, Bozen and IAM (see section 3). The new HTR has been integrated in Transkribus and can be used now. URO, UIBK and StAZH have also organized the *ICFHR2018 Competition on Automated Text Recognition on a READ Dataset*, tackling the task of writer adaptation described in section 4. The underlying task is formulated as followed.

Task 7.3 – Neural Network based HTR

The following is copied from the Grant Agreement:
This task comprises all investigations concerning Neural Network (NN) based HTR technology. It focuses on the NN training based upon a sufficiently large number of line images along with their textual transcripts, and how the NN output can be effectively decoded into word graphs for further processing in the overall workflow. This covers both training and decoding rely on well-known machine learning techniques that were extensively used and extended by the UPVLC and URO. Further flexibility, improvement and particular issues require ongoing research: maintaining stability of the training process for avoiding degenerated NNs; speeding up the training process for advanced flexibility; adapting to specific data for effective extendibility; researching on word-based graphs and character-based graph generated from NN-based decoders for interactive transcription and KWS; incorporating advanced language models into the decoding procedures for improved overall performance in transcription and search tasks. Furthermore the NN based HTR engine available at URO will be customized for use in the project workflow. From a research or an implementation point of view, we will then investigate the inter-

actions of the algorithms or the engine, respectively, across the interfaces to adjacent workflow modules.

2 HTR improvements

In year three we improved upon the new HTR baseline that was established in deliverable D7.8 both in terms of performance and speed. Advancements include an optimized network architecture, better GPU utilization, a simplified preprocessing pipeline, diversified data augmentation and modified training strategies. Additionally, first experiments with recurrence-free models were carried out to investigate fundamental alternatives to the current implementations.

2.1 HTR+ architecture

HTR+ is an adaptation of year two's HTR architecture. We are still using a deep model mainly consisting of nested convolutional layers, bidirectional long short-term memorys (LSTMs) and interweaved pooling layers, but with slight modifications. First of all the basic layer structure got rearranged into stacked bidirectional LSTM layers ontop of stacked convolutional layers, instead of a more loosely alternation. The convolutional stack as the first part of the NN focuses on 2D-feature extraction. Implicit and explicit downsampling in the form of convolutional strides in the first layer or pooling after the second and third layer reduce the spatial dimensions of the input image, thus lowering the computational cost and combatting overfitting. Addtionally the convolutional blocks from year two got simplified by excluding any utility layers like batch normalization or local response normalization and their filters were finetuned according to demands. The bidirectional LSTM stack as the second part of the NN introduces dynamic temporal behaviour by means of recurrent connections and allows for segmentation free and context sensitive processing. Overall we ended up with a simpified but deeper architecture than before. We also integrated GPU-optimized LSTM cells, which were the main bottleneck when profiling training and inference runs of our models, to take advantage of Transkribus' new hardware in the form of multiple GPU servers. The output layer of the model still provides the known confidence matrix (ConfMat) and decoding can be applied as before (see [9]). A graphical representation of the HTR+ architecture is depicted in figure 1.

2.2 Preprocessing and data augmentation

The input of an HTR model is a set of text line images extracted by the baseline detection algorithm. Quality and quantity of images play an important role for the following training algorithm. On the one hand, if inputting the raw grey textline images that we get from the digitized document we would end up getting low accuracy values. Therefore the preprocessing of images is an important step in the HTR pipeline. We simplified the previous preprocessing and are now four times faster than before. The main components are the size normalization to a height of 64 pixels, a contrast normalization without any

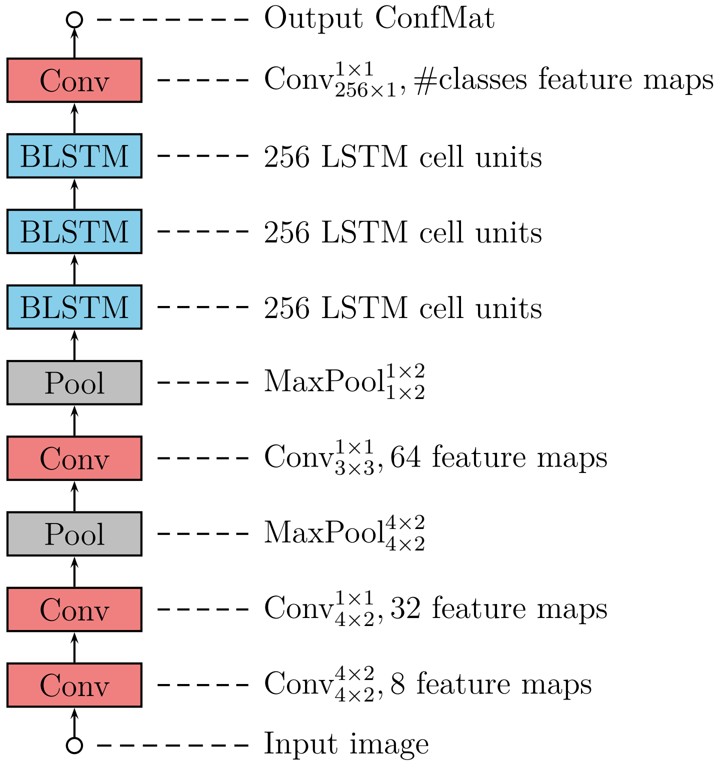

Output ConfMat
Conv
$\mathrm{Conv}_{256\times1}^{1\times1}$, #classes feature maps
BLSTM
256 LSTM cell units
BLSTM
256 LSTM cell units
BLSTM
256 LSTM cell units
Pool
$\mathrm{MaxPool}_{1\times2}^{1\times2}$
Conv
$\mathrm{Conv}_{3\times3}^{1\times1}$, 64 feature maps
Pool
$\mathrm{MaxPool}_{4\times2}^{4\times2}$
Conv
$\mathrm{Conv}_{4\times2}^{1\times1}$, 32 feature maps
Conv
$\mathrm{Conv}_{4\times2}^{4\times2}$, 8 feature maps
Input image

Figure 1: The proposed HTR+ model: A stacked Conv-BLSTM architecture with interweaved pooling layers and a ConfMat output. The sub- and superscript in $\mathrm{Conv/MaxPool}_{k_y\times k_x}^{s_y\times s_x}$ desribes the size of the kernel and the strides along both dimensions respectively. Furthermore, #classes is the number of character channels present in the ConfMat.

binarization of the image, a skew correction and a slant normalizer using the centre-of-gravity-line of a text line.
On the other hand, deep learning networks typically require large amounts of data to learn and generalize. However, for many datasets, especially historical documents, not a lot of ground truth (GT) data is available. One way to artificially increase this amount is data augmentation, where minor alterations are made to the existing training data. To simulate naturally occuring variations in handwritten text line images, we combine different methods, namely affine transformations, dilation and erosion, aswell as elastic and grid-like distortions. These methods are applied to the original line image randomly with an independent probability of 50%. Examples of affine transformations include translation, scaling, rotation, shear mapping and compositions of them in any combination and sequence. Dilation and erosion are two fundamental operations in morphological image processing, which basically function like image filters to enlarge or decrease relevant parts of an image, based on structuring elements. Elastic distortions deform the image by generating random displacement fields, convolving those with a Gaussian, and applying the displacements to the pixel values of the image [8]. For Gaussians with intermediate standard deviation, the displacement fields look like elastic deformations. Grid-like distortions try to capture slight differences in scale and slant

from character to character, instead of applying uniform deformations to the entire text line. They work by placing control points on a regular grid such that they align to the baseline, randomly perturbing those points and warping the image according to the perturbed grid [11]. Figure 2 shows some examples for seperately applied augmentation techniques to the original line image. We found a random combination of dilation, erosion and grid-like distortions to achieve the best overall results.

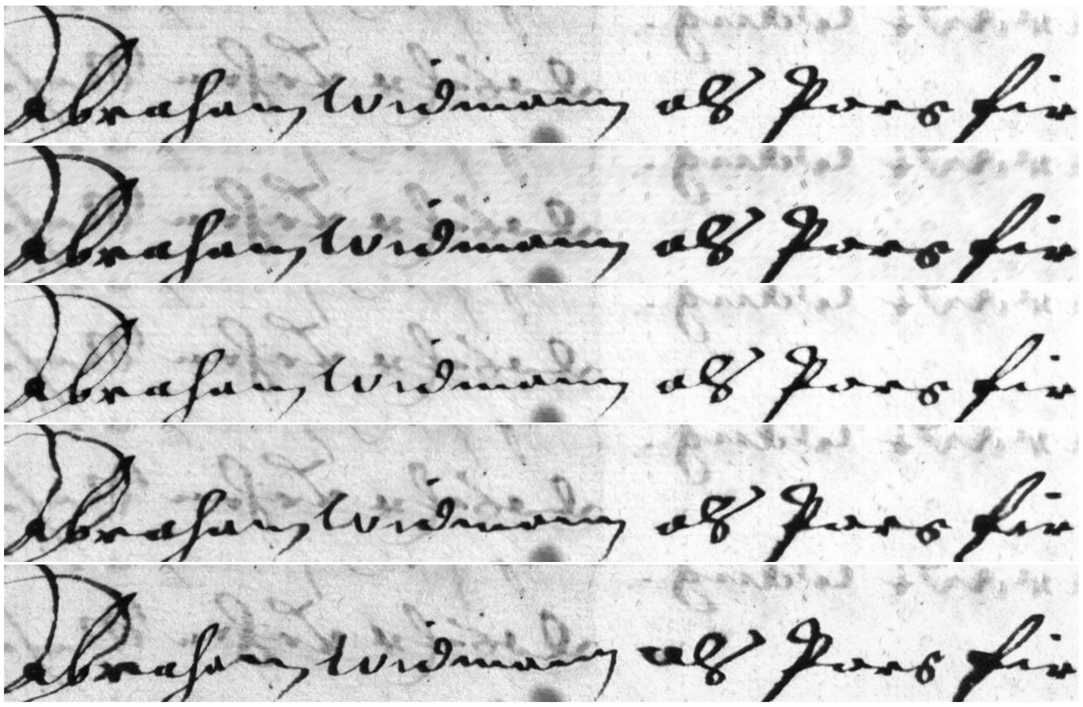


Figure 2: Line image and seperately applied augmentation techniques. From top to bottom: original, dilation, erosion, elastic, grid-like.

2.3 Modified training strategies

The HTR+ network is entirely differentiable and can be trained end-to-end in a supervised manner via the backpropagation algorithm. Using TensorFlows integrated concept of optimizers, we were able to experiment with a wide variety of optimization algorithms, like stochastic gradient descent, ADAM or RMSPROP, to name a few. One of the most important hyperparameters that ensures a robust and stable training is the learning rate, controlling how much the network parameters are adjusted. It's independent of the chosen optimizer and amongst other things, it's responsible for a smooth convergence of the training and can improve the final model performance. If the learning rate is too small, the training will take too long, since the network adapts very slowly. If the learning rate is too big, the network may fail to converge or even diverge. In year three we experimented with different learning rate schemes, to improve upon the constant learning rate baseline in year two. A simple exponential decay of the learning rate over the course of the training helps the network to adapt quickly in the early stages of

training and to finetune its parameters in the later stages of training. Another experimental approach was to lock the learning rate to a constant value in the beginning and to apply a cosine-like decay at the end. Overall the network converges faster with the exponential decay, but reaches better final performance with the latter approach, which is why we chose that setup in combination with the ADAM optimization algorithm as the new baseline for HTR+ training.
We also experimented with the concept of reinitialization strategies. The core idea is, to fully retrain an already pretrained model after reinitializing one or more specific layers of the network. The standard case comprises a full training run, after which a layer gets randomly reinitialized, the learning rate gets reset and a second full training run is executed. Theoretically there are no constrains on how many layers can be reinitialized in this manner or how often one can repeat the training process, but even with just one reinitialization loop, we observed a gain of about 10 % CER performance when applied on the output or last bidirectional LSTM layers. Doing multiple loops can be expensive, because you have to fully retrain the model each time, but exploiting the new GPU capabilities it seems feasible when one wants to really optimize a particular model.

2.4 Temporal Convolutional Networks

In addition to (character) accuracy, the time it takes to train and use a model is also important when judging the performance of a system. So we searched for a faster, and maybe better, alternative to the LSTM cells that are used in the current architecture. The temporal convolutional networks (TCNs) do just that by substituting recurrent neural network (RNN) layers through convolutional neural network (CNN) layers. The recurrent connections in an RNN, i.e., the property to look back in time, are imitated by dilated and stacked 1-D convolutions that are applied to an input sequence. The dilated convolutions make it possible to look back in history in a nonlinear fashion, so longer sequences can be handled. Furthermore, the stacked 1-D convolutions are used to extract information from the past and to output a sequence that has the same length as the input sequence. We followed the description in [2] and adapted the architecture so that we are able not only to look back in time, but also take future elements in the sequence into account. In other words, to mimic the bidirectional behaviour of bidirectional RNNs. First experiments showed that the TCNs are slower and not as good as the TensorFlow GPU-optimized bidirectional LSTM counterparts. Because of that there was no need to integrate this architecture into Transkribus.

3 Experiments

Experiments on various datasets demonstrate minor performance gains in terms of CER compared to year two's architecture. We also achieved a major speedup of the models, primarily for the training and convergence times, but also for inference when the trained models get deployed. We briefly present the data sets for the evaluation in the following and report on the corresponding results afterwards.

3.1 Data sets

Staatsarchiv des Kantons Zürich collection

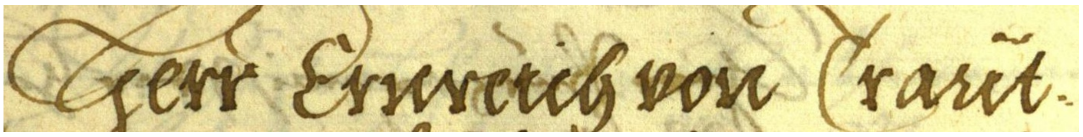
We refer to deliverable D7.8 (section 4) for the StAZH[1] collection, as it is presented in detail there.

Bozen

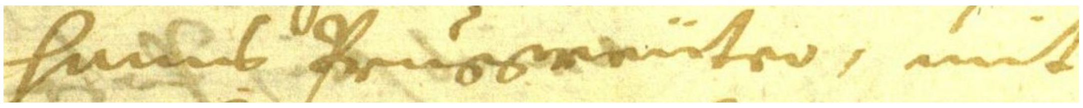
The Bozen dataset consists of a subset of documents from the Ratsprotokolle collection composed of minutes of the council meetings held from 1470 to 1805. This dataset is written in Early Modern German and the number of writers is unknown [10]. The data is divided into three different subsets with the following statistics:

- Train: 8366 lines.
- Validation: 1040 lines.
- Test: 1138 lines.

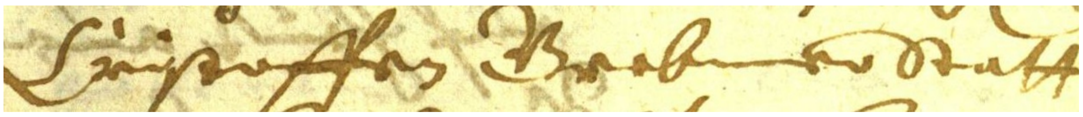
Some example images from the given subsets are depicted in Figure 3.



(a) sample from train



(b) sample from val



(c) sample from test

Figure 3: Sample lines from the three Bozen subsets.

IAM Handwriting Database

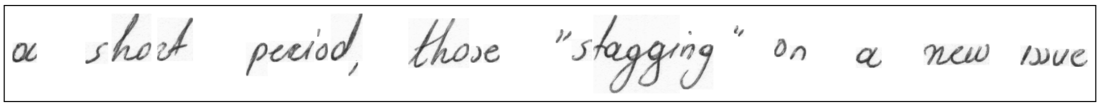
The IAM Handwriting Database contains forms of unconstrained handwritten text for off-line handwriting recognition. The database was first published in [5] at the ICDAR1999 and was expanded in October 2002 as described in [6]. The database is based on the Lancaster-Oslo/Bergen (LOB) corpus [3], which is a collection of texts that were

[1]The Staatsarchiv Zürich is a large scale demonstrator of the READ project.

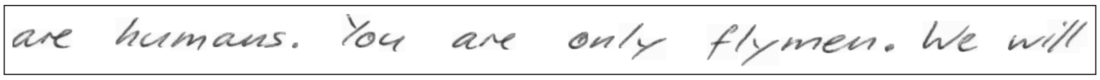
used to generate forms, which subsequently were filled out by persons with their handwriting.
The subsets and ground-truth processing are taken from [7]. The so called Aachen's partition of the dataset contains three different subsets with the following statistics:

- Train: 6161 lines from 747 forms.
- Validation: 966 lines from 115 forms.
- Test: 2915 lines from 336 forms.

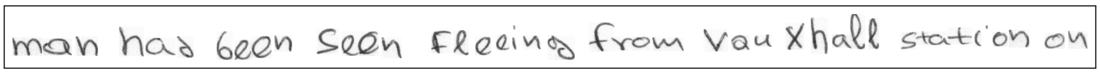
This is not the official partition of the dataset, but it is widely used for HTR experiments (in other applications, like KWS, other partitions are used). Some example images from the given subsets are depicted in Figure 4. The ground-truth is processed in several ways to fix some of its irregularities. Firstly, some characters that were originally separated by white spaces are put together again to form words (e.g. "B B C" → "BBC"). Secondly, the original data was unconsistently tokenized by separating contractions from the words. The contractions are attached to the word again (e.g. "We 've" → "We've").


a short period, those "stagging" on a new issue

(a) sample from train


are humans. You are only flymen. We will

(b) sample from val


man has been seen fleeing from Vauxhall station on

(c) sample from test

Figure 4: Sample lines from the three IAM subsets.

3.2 Results

We present comparative experimental results for the evolution of our HTR architecture since the beginning of the READ project, using the listed datasets. The architectures include the SPRNN [4] used in year one, the new baseline HTR implemented in TensorFlow used in year two and the development of said architecture over the course of year three. We observe a performance boost of about $5 - 10\,\%$ in terms of CER compared to the state of HTR+ in the beginning of 2018 and an overall performance gain of up to $80\,\%$ compared to year one's results (see Table 1).
By finetuning the overall network architecture, optimizing our models for better GPU utilization and simplifying the preprocessing pipeline, we were able to reach significant speed improvements in comparison to our previous models. As Table 2 shows, a full training run from scratch, using a single GTX 1080Ti GPU, only takes 1/10th of the time

Table 1: Comparison of different HTR architectures with regard to the CER on various datasets, evaluated on the corresponding test set.

	CER [%]			
	SPRNN	HTR+(2017)	HTR+(2018)	HTR+(e2018)
StAZH	14.48	4.45	3.21	**2.97**
IAM	–	8.06	5.97	**5.46**
Bozen	–	6.70	5.10	**4.89**

that was previously necessary. We also observed faster convergence times for network trainings, i.e., it takes less training epochs for new models to reach satisfactory error rates. In inference time when the trained models get employed by users and no GPU hardware is available, we still managed to speed up single core CPU performance by a factor of two.

Table 2: Comparison of different HTR architectures with regard to the training time on various datasets, using a single GTX 1080Ti GPU. The models are trained for 150 epochs with 8192 line images each.

	Train time [h]			
	SPRNN	HTR+(2017)	HTR+(2018)	HTR+(e2018)
StAZH	~72	54	73	**7**
IAM	~72	63	79	**7**
Bozen	~72	25	33	**4**

4 Writer adaptation – ICFHR18 competition

Normally an HTR model is trained on homogeneous data to achieve the best performance for similar writing styles or documents. In practical applications however, the amount of GT for one particular document to be transcribed is limited or, in the worst case, not available. On the other hand, many text corpora have already been transcribed and published and can be used to pretrain an HTR model. The question that now arises is how much additional training material of a specific document is necessary to adapt a pretrained HTR system in order to produce reasonably low error rates on the respective document. The *ICFHR18 Competition on Automated Text Recognition on a READ Dataset* also dealt with this question. The competition was organized by URO, UIBK and StAZH and hosted by the ScriptNet platform. For more information we refer to the competition website[2] and just present the submission results along with the HTR+(e2018) results in Table 3 where the quality of a submitted model is measured in terms of CER.
The competition has shown the importance of selection and amount of training data for an HTR system. If no GT is available, a pretrained model is a good starting point for

[2]`https://scriptnet.iit.demokritos.gr/competitions/10/`

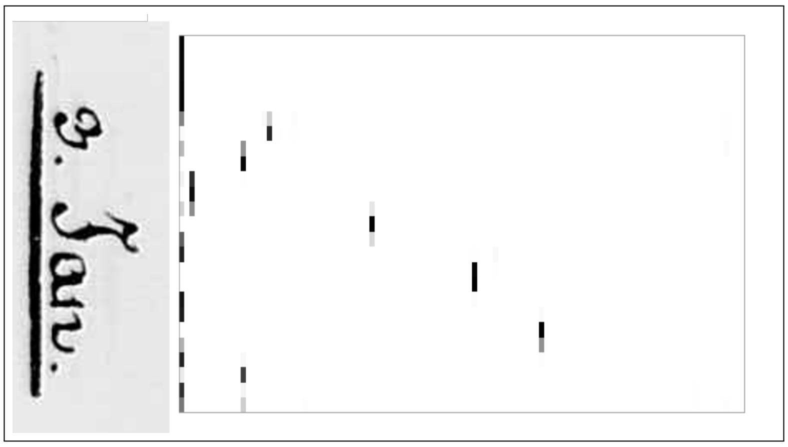
Table 3: Detailed results of the submissions. It describes the average CER for a certain number of additional training pages that where used to train on the HTR model.

	CER per additional specific training pages (in %)				
	0	1	4	16	total error
OSU	31.3987	**17.7344**	**13.2672**	**9.0238**	**17.8560**
ParisTech	32.2516	19.7981	16.9794	14.7213	20.9376
LITIS	35.2940	22.5078	16.8871	11.3448	21.5084
PRHLT	32.7927	22.1470	17.8952	13.3288	21.5409
RPPDI	**30.8045**	28.4038	27.2462	22.8461	27.3252
HTR+(e2018)	28.9226	16.3390	11.7684	7.1055	16.0339

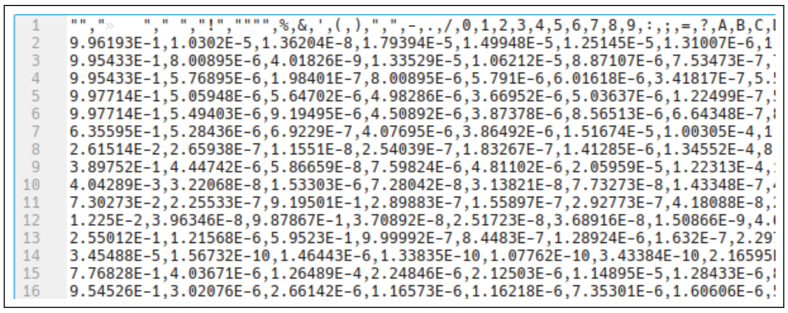
transcribing and a following retraining with new GT data. The more training material for one scenario is accessible, the better the model performs. One can also see that the HTR+ is outperforming all other competitors that submitted their results and thus yielding state-of-the-art results.

5 Additional Tools - ConfMat Exporter

There are many tasks working on raw output data like text, e.g., named entity recognition or topic modeling. When applying these models to handwritten text, i.e., to the (text) output of the HTR model, it can be useful to use the confidence values stored in the ConfMat. To use the ConfMats externally we want to store them in a suitable format. This is achieved by saving them in CSV-files following the RFC-4180 specification using the UTF-8 encoding scheme. To get a clear assignment between ConfMats and text lines the filenames are given by “line_id.csv”. To handle the decimal separator as a dot the en-US format is used. Futhermore the confidences are given in exponential notation with a significand-length of 6. In Figure 5 an example is given. The tool is not yet integrated in Transkribus but open source at `https://github.com/CITlabRostock/CITlabConfMat`.



(a) Line of handwritten text: “3. Jan.” used as input to the HTR model and the corresponding output of the NN. Black means high confidence, white means very unlikely. Rows correspond to positions in the image, columns correspond to classes (i.e. characters).



```
"","»    "," ","!","""",%,&,',(,),",",-,.,/,0,1,2,3,4,5,6,7,8,9,:,;,=,?,A,B,C,
9.96193E-1,1.0302E-5,1.36204E-8,1.79394E-5,1.49948E-5,1.25145E-5,1.31007E-6,1
9.95433E-1,8.00895E-6,4.01826E-9,1.33529E-5,1.06212E-5,8.87107E-6,7.53473E-7,
9.95433E-1,5.76895E-6,1.98401E-7,8.00895E-6,5.791E-6,6.01618E-6,3.41817E-7,5.
9.97714E-1,5.05948E-6,5.64702E-6,4.98286E-6,3.66952E-6,5.03637E-6,1.22499E-7,
9.97714E-1,5.49403E-6,9.19495E-6,4.50892E-6,3.87378E-6,8.56513E-6,6.64348E-7,
6.35595E-1,5.28436E-6,6.9229E-7,4.07695E-6,3.86492E-6,1.51674E-5,1.00305E-4,1
2.61514E-2,2.65938E-7,1.1551E-8,2.54039E-7,1.83267E-7,1.41285E-6,1.34552E-4,8
3.89752E-1,4.44742E-6,5.86659E-8,7.59824E-6,4.81102E-6,2.05959E-5,1.22313E-4,
4.04289E-3,3.22068E-8,1.53303E-6,7.28042E-8,3.13821E-8,7.73273E-8,1.43348E-7,
7.30273E-2,2.25533E-7,9.19501E-1,2.89883E-7,1.55897E-7,2.92773E-7,4.18088E-8,
1.225E-2,3.96346E-8,9.87867E-1,3.70892E-8,2.51723E-8,3.68916E-8,1.50866E-9,4.
2.55012E-1,1.21568E-6,5.9523E-1,9.99992E-7,8.4483E-7,1.28924E-6,1.632E-7,2.29
3.45488E-5,1.56732E-10,1.46443E-6,1.33835E-10,1.07762E-10,3.43384E-10,2.16595
7.76828E-1,4.03671E-6,1.26489E-4,2.24846E-6,2.12503E-6,1.14895E-5,1.28433E-6,
9.54526E-1,3.02076E-6,2.66142E-6,1.16573E-6,1.16218E-6,7.35301E-6,1.60606E-6,
```

(b) A snippet of the corresponding CSV-file, where the first row represents the channels. The other rows depict the confidences of the channels for a given timestep in the image.

Figure 5: Example of a ConfMat and the corresponding export CSV-format.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of

generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[3] S. Johansson, G. Leech, and H. Goodluck. *Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers.* Department of English, University of Oslo, Norway, 1978.

[4] Gundram Leifert, Tobias Strauß, Tobias Grüning, Welf Wustlich, and Roger Labahn. Cells in multidimensional recurrent neural networks. *The Journal of Machine Learning Research*, 17(1):3313–3349, 2016.

[5] U.-V. Marti and H. Bunke. A full english sentence database for off-line handwriting recognition. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR '99 (Cat. No.PR00318)*, pages 705–708, Sept 1999.

[6] U.-V. Marti and H. Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5(1):39–46, Nov 2002.

[7] Joan Puigcerver. Step-by-step training guide using iam database. `https://github.com/jpuigcerver/Laia/tree/master/egs/iam`, 2017.

[8] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, pages 958–963, Aug 2003.

[9] Tobias Strauß, Tobias Grüning, Gundram Leifert, and Roger Labahn. Citlab argus for keyword search in historical handwritten documents-description of citlab's system for the imageclef 2016 handwritten scanned document retrieval task. In *CLEF (Working Notes)*, pages 399–412, 2016.

[10] Joan Andreu Sánchez, Verónica Romero, Alejandro H. Toselli, and Enrique Vidal. Read dataset bozen, dec 2016.

[11] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price, and S. Cohen. Data augmentation for recognition of handwritten words and lines using a cnn-lstm network. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 639–645, Nov 2017.