READ

RECOGNITION & ENRICHMENT
OF ARCHIVAL DOCUMENTS

D7.8
HTR Engine Based on NNs P2

Building deep architectures with TensorFlow

Max Weidemann, Johannes Michael, Tobias Grüning, 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	31.12.2017
Actual date of delivery	01.12.2017
Date of last update	05.12.2017
Deliverable number	D7.8
Deliverable title	HTR Engine Based on NNs P2
Type	Demonstrator
Status & version	Final
Contributing WP(s)	WP7
Responsible beneficiary	URO
Other contributors	
Internal reviewers	Joan Andreu Sanchez (UPVLC)
Author(s)	Max Weidemann, Johannes Michael, Tobias Grüning, Roger Labahn
EC project officer	Martin Majek
Keywords	RNN, LSTM, CNN, segmentation free, TensorFlow

Contents

Executive Summary

The second year's deliverable describes the importance of state-of-the-art Neural Network based HTR models that yield a higher accuracy in terms of CER in comparison to architectures currently used in Transkribus. To build such models it is necessary to understand the underlying key components from the field of deep learning like the Convolutional Neural Network. Finally, a tool is needed to implement new and more complex architectures easily. Hence, the main focus of year two was research on the one hand and finding and familiarizing with a suitable tool on the other hand.

1 Introduction

Deep learning is currently the fastest-growing field in machine learning. It covers algorithms that use deep hierarchies of concepts to learn more complicated ones. These include deep Neural Networks (NN), Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) which are used in fields like speech recognition, computer vision, machine translation or natural language processing. But also fields like Handwritten Text Recognition (HTR) or Keword Spotting (KWS) benefit from such deep structures. Thus, deep learning, especially the CNN, was the main topic at the ICDAR2017, an international conference about HTR and document analysis. For this reason, one of the main tasks of year two was to take a closer look at the topics of deep learning and find a way to integrate them into Transkribus. TensorFlow, a machine learning library that is introduced in the following section offers a great opportunity to realize this. 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 interactions of the algorithms or the engine, respectively, across the interfaces to adjacent workflow modules.

2 TensorFlow – A C++ based Machine Learning Library

TensorFlow [1] is an open-source software library of Google for numerical computations, mainly used for machine learning applications. It was released at the end of 2015 and left its beta state in February 2017. TensorFlow computations are depicted using dataflow graphs, where the edges represent multidimensional data arrays (tensors, hence the name) that communicate via mathematical operations (ops) represented by the nodes.

TensorFlow has many advantages. There is a big community behind TensorFlow and has been declared the number one repository in machine learning on Github. In research and production, it is currently used by various teams in commercial Google products such as speech recognition, Gmail, Google Photos or the Google search. Because TensorFlow is under constant development there is no need to implement most of the parts that are being used in a model by yourself. Therefore it is possible to recreate many state-of-the-art architectures quickly and easily. This includes Feedforward NNs, RNNs, CNNs and more. E.g. it takes only a few lines of code to build a multi-layer NN. Besides, one can use the computing power of one or more GPUs for tasks like matrix multiplications to speed up the overall training process.

3 New architectures

A study [5] showed that BLSTM-based HTR recognizers can perform as well as MDLSTM-based HTR systems in terms of recognition accuracy and outperforms them in terms of memory usage and running time. Therefore, the new models are still using RNNs but the 2-D LSTM cells are replaced by BLSTMs. The concepts of (Bidirectional) Long Short-Term Memory ((B)LSTM) networks and Multidimensional LSTM (MDLSTM) networks are beyond the scope of this deliverable. For more details we refer to [2, 6]. Still, a brief introduction is given in section 3.1.
It should be noted here that the output format as introduced in the first year deliverable D7.7 is still supported in the new architectures and decoding can be applied on the confidence matrices (ConfMats) as before (see [7]). The new architectures are segmentation free and support context sensitive processing as well.

3.1 Long Short-Term Memory

Basically, an LSTM network is an RNN with a special kind of cell that is able to store information for a longer time period while an MDLSTM network introduces a recurrence along multiple axes (for HTR the x-axis and y-axis of an image). In the bidirectional case (BLSTM) the amount of input information is increased by introducing two RNN layers with different directions, one for positive time direction and another for negative time direction (i.e. along the x-axis). The idea is visualized in Figure 1.

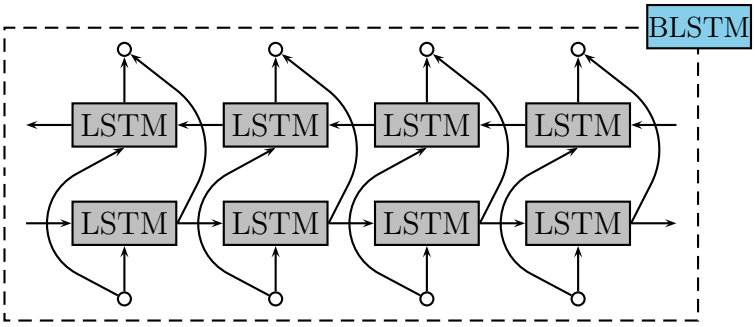


Figure 1: General structure of a bidirectional RNN block using LSTM cells.

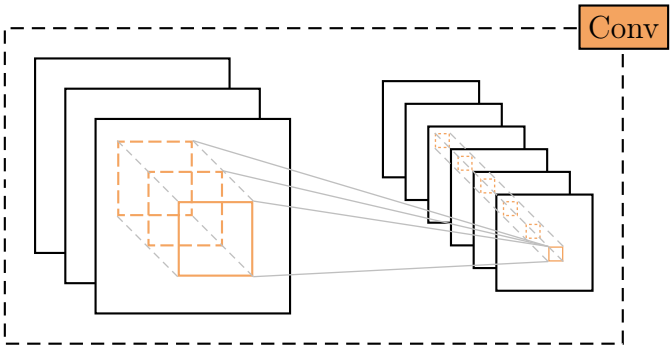


Figure 2: General structure of a convolutional block.

3.2 Convolutional Neural Networks

CNNs as introduced in [3] are a specialized kind of neural network for processing data that has a known grid-like topology, e.g. images which can be viewed as a 2-D grid of pixels. As the name indicates the network employs a linear operation called convolution. In the field of machine learning we apply a discrete convolution

$$\begin{aligned}(I * K)_{i,j} &= \sum_m \sum_n I_{m,n} K_{i-m,j-n} \\ &= \sum_m \sum_n I_{i-m,j-n} K_{m,n}\end{aligned}$$

where the input (the image) I and the kernel K are 2-D arrays where K holds the trainable parameters. The kernel can also be viewed as a sliding window that is shifted over the image I (with a certain stride) to detect local features. The output of the convolution is often referred to as feature map. The concept is visualized in Figure 2. Sparse (or local) connectivity and parameter sharing are two major advantages of convolution if used in machine learning. In traditional NN layers every output unit is connected with every input unit, such that we have seperate parameters for describing the interaction between these units. In CNNs, however, this number of paramaters depends only on the size of the kernel K which is usually significantly smaller than the input image. The kernel makes it possible to detect small features like edges with much less parameters yielding lower storage costs, hence reducing the memory requirements

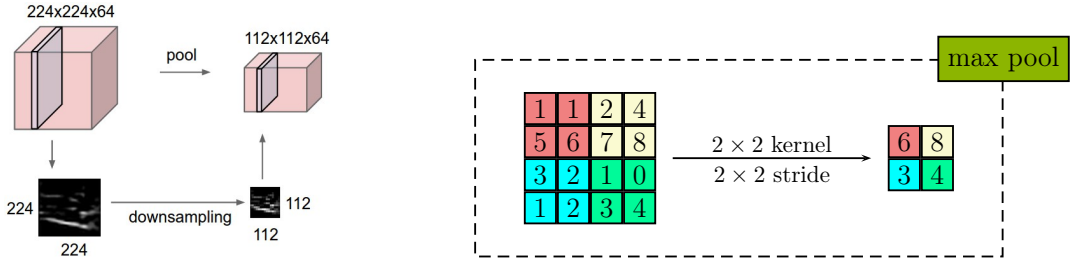


Figure 3: Max pooling operator used to downsample a layer output and improve generalisation.[1]

of the model. Parameter sharing is also used to control the number of parameters in a model. It follows the idea that if a feature is useful at some position in an image, then it should also be useful at a different position. In other words, parameter sharing easily enables the system to detect similar structures at different locations of the image.

3.3 Pooling Layer

Pooling can be used to lower the computational costs even more with non-linear downsampling. It reduces the spatial size of the representation to decrease the amount of parameters and to create high-level features out of low-level ones. Max pooling is a method that is used most frequently where a max filter is applied to rectangular (usually) non-overlapping subregions of a layers output as depicted in Figure 3.

3.4 Construction of the new architecture

A brief introduction into the new competing architecture should be given here. We are using a deep architecture mainly consisting of nested CNN and RNN layers with optional Pooling layers. An example can be seen in Figure 4. As already mentioned, the CNN layers do a 2D-feature extraction. Explicit downsampling is achieved by the pooling layers. Once a certain feature is present it suffices to know about its relative location in the image, so you don't necessarily lose (a lot of) information during this process. Generally, subsampling reduces the spatial dimension of the input, which in turn results in less computational cost.
The RNN layers introduce dynamic temporal behaviour by means of recurrent connections and allow context sensitive processing as can be seen in D7.7 section 3.2. Instead of using multidimensional LSTM cells we now employ bidirectional LSTM cells which leads to faster and more efficient models with negligable changes in performance. RNNs can process arbitrary sequences of inputs which makes them applicable to tasks such as unsegmented HTR as we have already seen in D7.7 section 3.1. Finally, the RNN output in terms of ConfMats is unchanged in comparison to previous HTR models (see D7.7 section 3.3).

[1] `http://cs231n.github.io/convolutional-networks/`

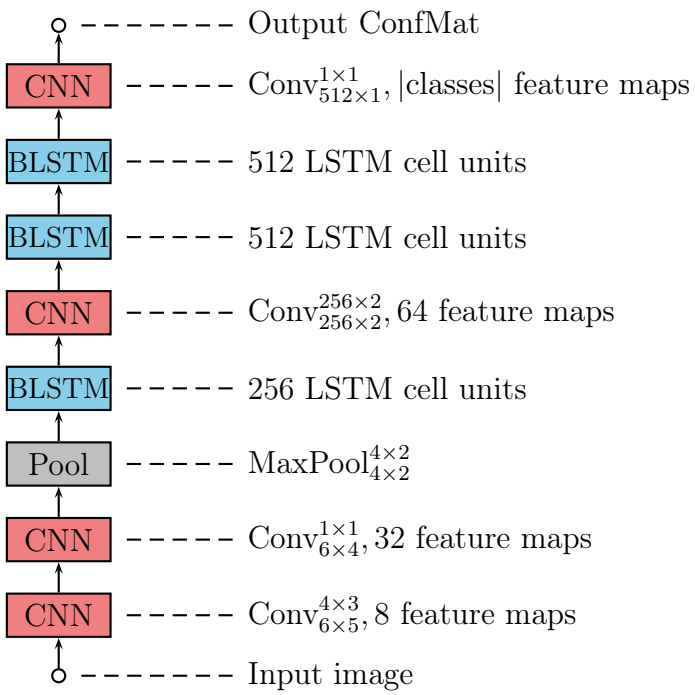


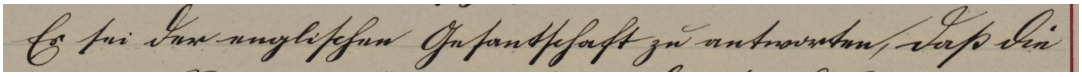
Figure 4: The proposed HTR model: A combination of various CNN and BLSTM blocks with one additional pooling layer. The subscript in $\text{Conv/MaxPool}^{m\times n}_{\tilde{m}\times\tilde{n}}$ describes the size of the kernel and the superscript defines the stride along both dimensions. Basically, |classes| is the number of characters present in the ConfMat.

The network is entirely differentiable and can be trained end-to-end in a supervised manner via the backpropagation algorithm. TensorFlow uses a technique called reverse mode automatic differentiation which basically means that it provides functions to compute symbolic (partial) derivatives for a given computation graph automatically. This allows us to create and experiment with complex architectures where we do not have to worry about the intricacy of propagating errors through various layers and functions. However, deep and more complex models require even more computing power than before. The TensorFlow API supports Multi-GPU training in synchronous and asynchronous settings, enabling us to train and test our proposed architectures and to get results in a reasonable timeframe given the necessary hardware.

4 Experiments

First tests demonstrate a performance boost of up to 50 % in terms of Character Error Rate (CER) compared to the Sequential Processing RNN (SPRNN) [4] used in year one. The experiments were performed on the StAZH data set that is described in the following.

(a) sample from val_a



(b) sample from val_b

Figure 5: Sample lines from the two different validation sets.

Data set

The data set for the experiments was chosen from 800 pages of the Staatsarchiv des Kantons Zürich[2] (StAZH) collection. This set consists of *Kantonsratsprotokolle* and *Regierungsratsbeschlüsse* whereas the former contains less than 10 % of the overall amount of text.
The texts consist of resolution/enactment of the cabinet as well as the parliament of the state of Zurich (starting in 1848 "canton"). The first documents were written in 1803, the last in 1882. The script is a very well-formed and highly trained German current. Different involved scribes are from a paleographic point of view quite similar but still distinguishable.
A subset of this collection was used for training the neural network: 4 pages from every year, i.e., 320 pages. For validation of the models two separate validation sets were used (see Figure 5):

val_a contains 1 page of every second year from the above mentioned 800 pages, i.e., in total 40 pages.

val_b consists of 20 pages of minutes parliament of the state of Zurich from the year 1882 (document id 13709). The writers are unknown to the neural network.

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

Table 1: Comparison of the SPRNN and the TensorFlow network on the StAZH dataset in terms of CER.

Validationset	SPRNN	TensorFlow
Same writer in train and val set (val_a)	14.48 %	7.18 %
Different writer / same domain (val_b)	22.61 %	16.38 %

Some results on the StAZH dataset are shown in Table 1. The comparative model consists of several CNN layers followed by RNN layers as introduced in section 3 and D7.7.
From the results one can conclude that the new HTR causes a vast drop in error rate if we use the trained network for similar fonts/writing styles. It also becomes clear that the performance of the network depends heavily on the choice of training data. One can not assume that a model trained on a particular dataset means that it can be applied to any other dataset without loss in HTR accuracy. By now the models are generated in Python and can then be used in Java by means of a wrapper.

5 Outlook

The experiments show that deeper architectures using CNNs and RNNs lead to a significant improvement in HTR accuracy. Therefore, one of the main goals in year three will be the integration of TensorFlow models into Transkribus. Finally, writer adaptation is another task that will be tackled. This involves adapting a pre-trained writer-independent network to a specific writer with a small amount of training data and comparing it with the original pre-trained HTR model which is based on large amounts of data. To cite deliverable D8.4 (section 2.2.2) the question is "[H]ow many writers can be fitted into one model with CER below 10 % and whether it is possible to build models broad enough to recognize writers/hands unknown."

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] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[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] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 67–72, 2017.

[6] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[7] 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.