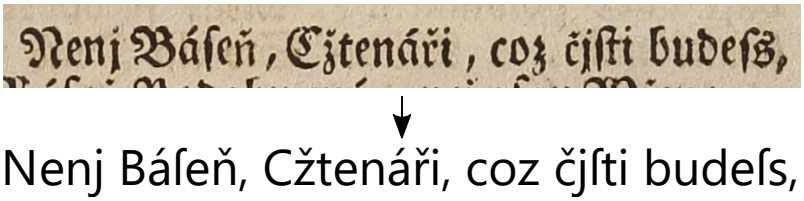
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2018
http://excel.fit.vutbr.cz

Convolutional Networks for Historic Text Recognition

Martin Kišš*



Abstract
The aim of this work is to create a tool for automatic transcription of historical documents. The work is mainly focused on the recognition of texts from the period of early modern times written primarily using font called Fraktur. The problem is solved using convolutional neural networks with addition of Spatial Transformer Network. The solution also includes implemented generator of artificial historical texts. The proposed neural network was trained on a dataset created by this generator and for evaluation real historical texts were used. On the real historical dataset, the network achieved 81.8 % of correctly recognized characters. The benefit of this work is the finding that it is possible to train the neural network on artificial data to be able to recognize real historical texts.

Keywords: Historical text — OCR — Convolutional Neural Networks

Supplementary Material: *N/A*

*xkissm00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

Libraries and archives worldwide have digitized large number of historical documents. However, these documents are mostly inaccessible as they lack transcriptions and searching for specific information is time consuming. The aim of this work is to create a tool capable of automatic text transcription of early printed documents. The transcriptions will allow historical documents to be searched using standard full text queries which will improve efficiency of historical research and which will make the documents accessible to general public.

This work is mainly focused on transcription of texts from early modern time (approximately 16–18th century) written using font called *Fraktur* [1]. During 19th century, this font (commonly Blackletter typeface) was gradually replaced by humanistic fonts, such as Antiqua [2]. Definitive end of general usage of Fraktur is dated to Second World War, when Nazi Germany, as a last country, stopped using this font. More detailed description of this font is in section 2.

The approach of proposed solution in this paper is based on convolutional neural networks which transcribes single line of a text. During the transcription network produces two output vectors. First vector consists of values representing probability of given character in the image. Second vector represents pixel distance between currently recognized character and the next character. The resulting transcription is created by producing character outputs of the neural network while moving through the image. This article is focused on transcribing English written texts, but the trained solution should also handle other languages, except some language-specific characters.

The two main contributions of this paper are as follows. Firstly, it is a novel approach to historical text lines recognition using convolutional neural networks. Secondly, a generator of artificial historic texts is implemented. The main benefit of this generator is the ability to create training and test data with ground-truth annotations. This generator creates text pages with large variations in its content, such as different fonts and their sizes, backgrounds and others. Beside this main image output, the generator stores annotation files which could also be used for other machine learning tasks (e.g. semantic segmentation).

2. Blackletter

Blackletter is a name for typeface used since 12th century to first half of 20th century. One of the most used fonts was Fraktur [1] (Figure 2c). It was largely used in early modern time and it is based on middle age font called Schwabacher (shown in Figure 2b). Fraktur, in comparison with Schwabacher, is narrower and therefore more space-economical.

In modern time typeface called *Antiqua* (shown in Figure 2a) was also used. Besides the appearance, the main difference was also in the usage of the font. While Antiqua was used for texts written in Latin, Blackletter was used for native language texts. Hence using Blackletter (Fraktur) also had an important nationalistic meaning, each nation had its own, little bit different, type of this font [3].

The name Fraktur is derived from Latin word *frangere* which is translated as *"to broke"* [1]. Shape of its letters has a lot more angles when compared to Antiqua and therefore this font is not very suitable for handwriting but it is usually used for printing. Illustration of Antiqua, Schwabacher and Fraktur is shown in Figure 2.

Recognition of texts written using Blackletter is complicated by the high variability of the content. Primarily, various Fraktur font variants exist, text lines are not always straight, quality of document printing is different, scanned documents can have some paper defects, etc. Example of such page is shown in Figure 1.

3. Related work

Classical approach to text recognition relies on progressive steps of image binarization, character segmentation, character classification and optional dictionary text verification [6]. However, such process is fragile as if any step fails or looses important information, it is hard for the subsequent steps to correctly

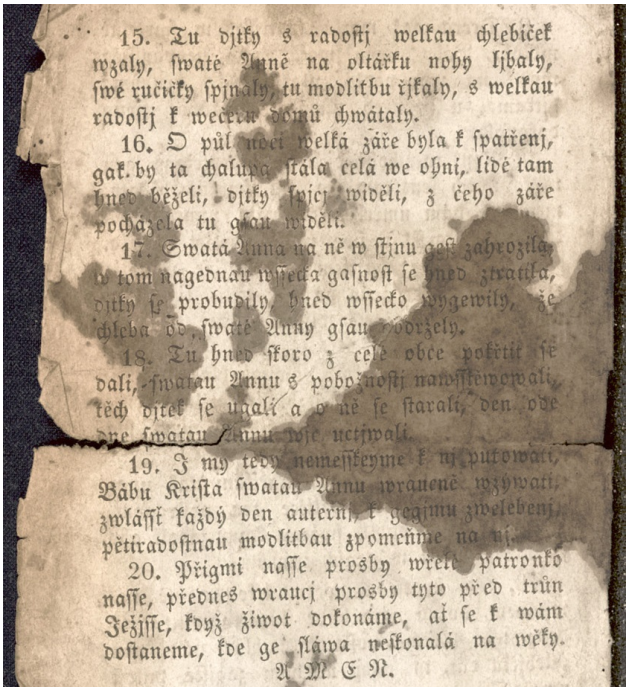


Figure 1. Illustration of scanned historical document page. Taken from [4], modified.

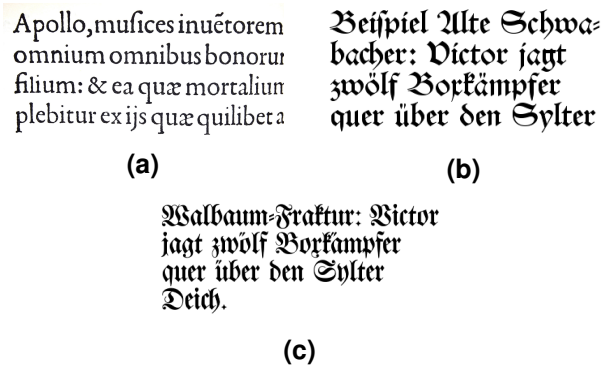


Figure 2. Illustration of fonts Antiqua (2a), Schwabacher (2b) and Fraktur (2c). Taken from [2, 5, 1], modified.

recover. On the other hand, modern machine learning approaches, e.g. convolutional neural networks and recurrent neural networks, are powerful enough that the whole process of text transcription can be handled as single end-to-end learning task. A single neural network can incorporate character localization, classification and even a language model and learn to transform images directly to text representation [7].

Processing sequential data, such as text, sound, etc, often requires context as the input can contain dependencies within entire sequence. The context could not be achieved by standard convolutional neural networks as it process the input without any other knowledge, but it could be achieved by *recurrent neural networks* (*RNN*). RNNs are similar to the standard convolutional neural networks, but they contain special layers devel-

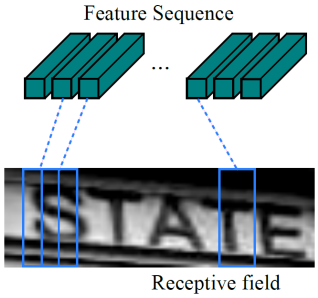


Figure 3. Illustration of CTC. Taken from [11].

oped for storing the context. These layers are called *Long Short-Term memory* (*LSTM*) [8] and *Gated Recurrent Unit* (*GRU*) [9].

Current solutions of text recognition are mainly focused on recognition of scene text and handwritten text. Scene text is such text which appears in some common situation, for example it could be photographed text on an advertisement banner. In these situations, it is common that the conditions when the image is captured are not always the same. Therefore recognition systems have to deal with a lot of variability in images (different illumination, location and rotation of the text, etc.).

One of the approaches to scene text recognition is based on *histogram of oriented gradients* (*HOG*) [10]. The principle of this method is in calculation of gradients in each 3×3 pixel cutout of original image. For every column of gradients sum of those is made. This results in vector of histograms of oriented gradients which is subsequently used for text recognition using recurrent neural network. Output of this network is processed using technique called *Connectionist Temporal Classification* (*CTC*.

Connectionist Temporal Classification CTC is a loss function used for training recurrent neural networks that output sequences. The key part of using this method is that the input image is processed with a constant step (see Figure 3).

In another approach to recognize scene text is also used CTC, but instead of histogram of oriented gradients convolutional layers are used to extract features from the input image [11]. In other work, *Spatial Transformer Network* (*STN*) is combined with RNN [12].

Spatial Transformer Network STN is a special module used as a layer in neural network which learns to perform geometric transformations on given input matrix (e.g. image) to focus on some object of interest.

Approaches to handwritten text recognition do not much differ from scene text recognition approaches. Therefore same techniques, such as CTC, are also used [7, 13].

Different approach to text recognition used *Springmann et al.* [14] who are not using any method to extract features from the input image. Instead of features, their input to recurrent neural network is directly a column of pixels of an input image. This method was used for recognition of historical text lines.

4. Proposed model

The proposed solution is a convolution neural network consisting of two branches. The first branch is responsible for classification of the character of the current input, the second branch estimates distance to the next character (*positioning branch*).

During transcription of an input image, cutouts of this image are made. Character branch outputs a vector which is probability distribution over possible characters. Positioning branch outputs also a vector of probabilities where each value represents probability of a given distance to the next character with resolution one pixel. At the most probable predicted position next cutout is made and the whole process is repeated. Illustration of the process is shown in Figure 5. The recognition process is terminated by a special character representing end of line.

Both branches of the network are trained for classification, therefore the architecture of the network is based on common architecture of classification convolutional neural networks [15]. Beside traditional layers (convolutional, max-pooling, fully connected, etc) the *character branch* uses the STN to learn to focus exactly on the character. The STN is not used in the *positioning branch*, as the changing geometric transformations could interfere with the task of estimating the relative position.

The described network processes each character independently, which is not optimal as context is often necessary to distinguish ambiguous characters. To add context, the network may be augmented by recurrent layers (e.g. LSTM), which would replace the fully connected layers.

In the first phase of training, the proposed network is trained on inputs where the character and the distance to the next character is exactly known. Before this training, a noise is also added to the positions of characters in training data, so the trained network should deal with inaccurate classifications from previous steps while it is evaluated.

In next phases, the network could also be trained on data where ground-truth information is missing (e.g. historical data). This data could be automatically

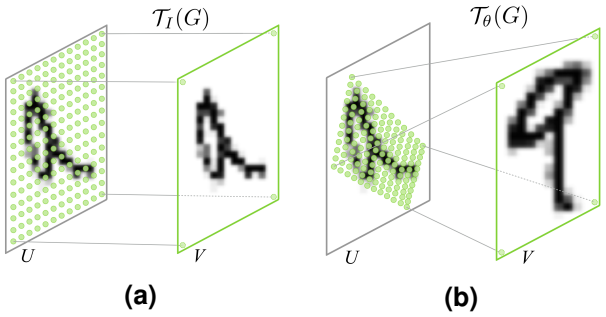


Figure 4. Illustration of application of generated matrix of input U resulting in output V. Figure 4a shows application of matrix $\mathscr{T}_I$ which is identity matrix. In figure 4b, application of affine transformation matrix $\mathscr{T}_\theta$ is shown. Taken from [17].

aligned for training as follows. Firstly, it is needed to use previously trained network and transcribe the input to produce several outputs. From these transcriptions and, if available, ground-truth information, it is possible to determine correctly and incorrectly classified characters and positions. Further, it is possible to align these outputs, while characters and their positions, which reaches given certainty of correctness, could be added to new dataset. For example, the alignment could be done using Levenshtein distance [16].

Spatial Transformer Network Spatial Transformer Network is a module in neural network which performs geometric transformation on given input [17]. It can apply translation, rotation, scaling, etc. on the input tensor (for example image). Entire mechanism compounds of three parts - *localization network*, *matrix generator* and *sampler*.

The localization network is neural network whose input is a matrix $U \in \mathbb{R}^{H \times W \times C}$ where H is a number of rows of the matrix, W is a number of columns of the matrix and C is a number of channels of the matrix. Output of this network is a vector of parameters θ, which defines applied transformation. From those parameters, matrix $\mathscr{T}_\theta$ is generated.

In the next step of the STN, original input is sampled using transformation matrix $\mathscr{T}_\theta$, obtained in previous step, and also using some interpolation (e.g. bilinear). This interpolation must be derivable to be able to compute derivatives for backpropagation algorithm. Visualization of how STN works is shown in figure 4

5. Dataset generator

Significant part of this work is a generator of artificial historical texts. There were several reasons to create this generator. First of them was that with this tool it is possible to have really large dataset containing thousands of training examples. Another reason was

Figure 5. Example of work of proposed solution. In Figure 5a cutouts in step t (red rectangle) and in step $t+1$ (green rectangle) are shown. In Figure 5b predicted positions are shown. The red line represents predicted position for step t and the green line represents predicted position for step $t+1$.

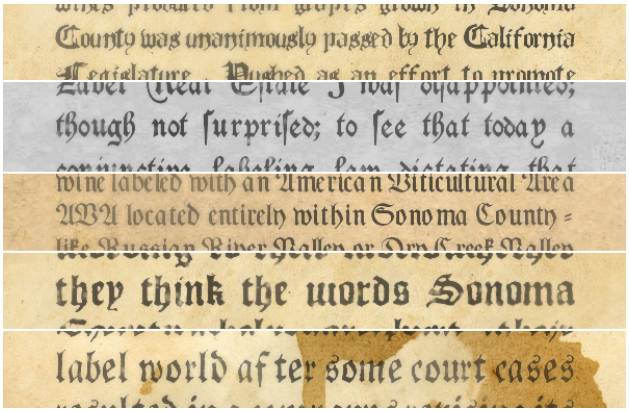


Figure 6. Examples of generator output.

that this generator is able to create detailed annotations about generated image. Those annotations consist of exact bounding box of each character in the text and its location.

Probably the most important requirement for this generator is to generate images which are as much as possible similar to real historic texts. To achieve this, so called *effects* are responsible for decreasing quality of rendered text to make resulting images look more realistic.

Entire process of generating image is as follows. At first, selected text is rendered. During the rendering process, characters are randomly shifted in horizontal axis and slightly rotated. Also size of the font is randomly chosen with respect to set minimal and maximal size. This creates grayscale image containing one page of text aligned to block and to which effects are applied. Example of output of the generator is shown in Figure 6.

5.1 Effects

As mentioned before, effects should modify rendered text to achieve more realistic output. First effect of the pipeline adds printing imperfections to the original rendered text. Essentially it is adjustment of alpha channel of the rendered text. The alpha channel is updated in two ways. Firstly, it is multiplied with generated Perlin noise map. Thus the text is less significant in some places, when it is placed on a background. Secondly,

random spots are generated. These spots have alpha channel lowered as much as it looks like the text was not printed at all in that locations.

The next effect in the sequence is effect that adds another text to the image. This second text is mirrored and it should imitate text that is on the other side of the paper and which shines through the paper. Or it can be viewed as a text which is printed on the next paper and a little bit of the ink was accidentally also applied to this page. To this text same effects as to the main one are applied, but this text is more transparent, so it looks brighter and not so significant in the resulting image.

Another effect places so far modified image on a random place in one of input backgrounds. This is done using alpha blending. In the last step, Gaussian blur is added to the resulting image. This is done to lower the sharpness of characters as it is common in historical documents.

6. Experiments

This section describes two conducted experiments on which neural network was evaluated. The first experiment focuses on network architecture and shows whether the transcription success rate depends on the size of the network. The second experiment aims to show whether there are any advantages of using STN in the network.

Datasets In both experiments networks are trained on the same artificial dataset created by the generator. In total, it contains 481 250 samples for training and 86 130 samples used for validation during the training. For the evaluation there are two datasets. First consists of 305 generated artificial text lines that contain 13 371 characters. The second dataset is hand-annotated dataset of 82 text lines (3 023 characters in total) extracted from the Impact database [18]. All datasets contain texts written in English.

Evaluation The evaluation is performed by calculating the Levenshtein distance between the obtained transcription and the ground-truth transcription. Total success rate is calculated as a ratio of count of successfully transcribed characters to count of all characters in the dataset.

6.1 Size of the network

As mentioned above, this experiments focuses on size of the network. Architectures of both networks are shown in 1. There is also used STN after second max-pooling layer in the character branch. The architecture schema of the networks is shown in Figure 7.

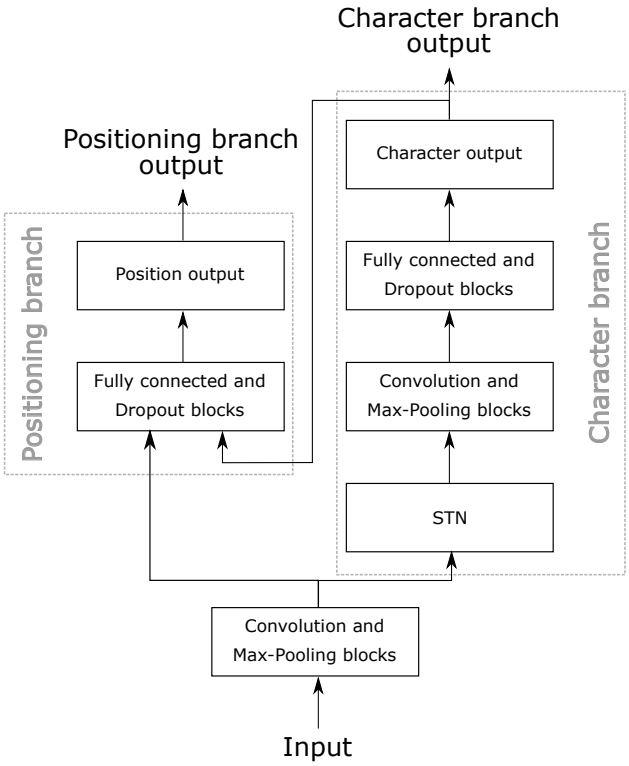


Figure 7. Architecture schema.

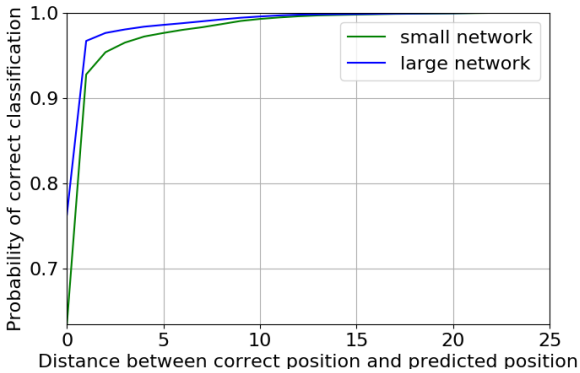


Figure 8. Results of positioning branches from experiment 1.

In both networks, same localization network is used for STN. This network consists of four convolutional layers with kernel size 2×2, two max-pooling layers with size 2×2, two fully connected layers with size 256 and two layers performing dropout regularization with probability of 0.3. Output of the localization network is a vector containing three values defining geometric transformation which allows cropping, translation and isotropic scaling [17]. The sampler of the STN layer calculates resulting matrix using bilinear interpolation.

Results are shown in table 2. Figure 8 shows that larger network predicts position of the next character with higher accuracy and from Table 2 it can also be seen that this network's character success rate is higher.

Table 1. Architectures of networks from experiment 1. In this table, *C* represents number of convolutional layers and their kernel sizes, *MP* represents number of max-pooling layers and their filter sizes, *FC* represents number of fully connected layers and their sizes.

Network	C	MP	FC
Small	$3\times(2\times2)$	$3\times(2\times2)$	$2\times(256)$
Large	$9\times(2\times2)$	$3\times(2\times2)$	$2\times(1024)$

Table 2. Results of transcriptions from experiment 1. Values represent character success rates.

Network	Artificial dataset	Historical dataset
Small net	83.2 %	75.8 %
Large net	91.1 %	81.8 %

6.2 Enhancements with STN

In this experiment, the larger network from previous experiment is trained and evaluated also without STN layer. Results of both networks are shown in table 3. From these table, it can be seen that network with STN layer has a little bit better performance.

Table 3. Results of transcriptions from experiment 2. Values represent character success rates.

Network	Artificial dataset	Historical dataset
With STN	91.1 %	81.8 %
Without STN	89.4 %	79.4 %

6.3 Result discussion

From the performed experiments it can be seen, that it is better to use larger network and also usage of STN layer improves its success rate.

Manually checking the results of experiments has shown that transcriptions often contain mistakes that could be eliminated when recurrent layers are used. For example, common mistakes included a substitution of a lowercase and uppercase form of the letter, substitution of a zero and character ”o”, or a bad insertion of spaces.

7. Conclusions

In this work, a novel approach to historical documents recognition is proposed. This approach contains convolutional neural network which consists of two branches. The first branch is used for classification of current character in the input image, the second branch is responsible for prediction of distance to next character in the sequence. Part of this work is also implemented generator of artificial historical texts. Using this generator, training dataset for the proposed network is created.

The experiments demonstrate that it is possible to train the proposed neural network on artificial historical texts and then use it to transcribe real historical documents. The experiments also show that for higher success rate of the transcription it is better to use larger networks with STN layers. The best trained network achieved 81.8 % of success rate on the real historical dataset.

Future work includes adding recurrent blocks to the proposed network and also retraining the network on aligned real historical data.

Acknowledgements

I would like to thank my supervisor Ing. Michal Hradiš, Ph.D. for his help.

References

[1] Wikipedia contributors. Fraktur.

[2] Wikipedia contributors. Antikva.

[3] J. Kašpar. *Soubor statí o novověkém písmu*. Univerzita Karlova, 1993.

[4] Špalíček. Digitální knihovna kramářských tisků.

[5] Wikipedia contributors. Schwabacher.

[6] Kai Wang and Serge Belongie. Word spotting in the wild. In *Proceedings of the 11th European Conference on Computer Vision: Part I*, ECCV'10, pages 591–604, Berlin, Heidelberg, 2010. Springer-Verlag.

[7] J. Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 67–72, Nov 2017.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

[10] Bolan Su and Shijian Lu. Accurate recognition of words in scenes without character segmentation using recurrent neural network. *Pattern Recognition*, 63(Supplement C):397 – 405, 2017.

[11] 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*, 39(11):2298–2304, Nov 2017.

[12] Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Robust scene text recognition with automatic rectification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] Vu Pham, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[14] Uwe Springmann and Anke Lüdeling. *OCR of historical printings with an application to building diachronic corpora: A case study using the RIDGES herbal corpus*, February 2017.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Rishin Haldar and Debajyoti Mukhopadhyay. Levenshtein distance technique in dictionary lookup methods: An improved approach. *CoRR*, 2011.

[17] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2017–2025. Curran Associates, Inc., 2015.

[18] Digitisation.eu. Impact centre of competence dataset, 2018.