Deep BLSTM Neural Networks for Unconstrained Continuous Handwritten Text Recognition

Volkmar Frinken
Faculty of Information Science
and Electrical Engineering
Kyushu University, Japan
vfrinken@ait.kyushu-u.ac.jp

Seiichi Uchida
Faculty of Information Science
and Electrical Engineering
Kyushu University, Japan
uchida@ait.kyushu-u.ac.jp

Abstract—**Recently, two different trends in neural network-based machine learning could be observed. The first one are the introduction of Bidirectional Long Short-Term Memory (BLSTM) neural networks (NN) which made sequences with long-distant dependencies amenable for neural network-based processing. The second one are deep learning techniques, which greatly increased the performance of neural networks, by making use of many hidden layers. In this paper, we propose to combine these two ideas for the task of unconstrained handwriting recognition. Extensive experimental evaluation on the IAM database demonstrate an increase of the recognition performance when using deep learning approaches over commonly used BLSTM neural networks, as well as insight into how different types of hidden layers affect the recognition accuracy.**

I. Introduction

The automatic unconstrained recognition of continuous, writer independent handwritten text is a challenging task and has been the focus of research for many decades [5], [21]. Even though important progress has been achieved, the task still remains challenging and can not yet be considered as solved.

A big step forward in reducing the overall error rate has been the introduction of bidirectional long short-term memory neural networks, a recurrent neural network architecture able to deal with long-term dependencies. The output of the network treated directly as character posterior probabilities [10] or input features for hidden Markov models [14], gave a significant boost to state-of-the-art recognition performances.

In this paper we investigate the impact of a extension to neural networks, namely deep learning. Recent advances in neural network research proposed strategies to train networks with several hidden layers, a problems that previously seemed unfeasible [3]. Existing deep learning approaches focus on feed-forward neural networks, and very often convolutional neural networks. In contrast, we focus on recurrent BLSTM neural networks. This way, we can make use of long-term sequence dependencies as well as the improved pattern classification power of deep learning architectures.

While specific multi-layer BLSTM neural networks have been proposed before [9], [15] for the task of speech recognition, we explore the applicability of using the ideas of deep learning for the task of handwriting recognition. Instead of suggesting one specific topology, we aim to give a broader overview of how BLSTM neural networks can be extended and the impact on the word error rate.

The rest of the paper is organized as follows. In Section II a brief overview of related work is given. Section III introduces deep BLSTM neural networks. An experimental evaluation is presented in Section IV and conclusions are drawn in Section V.

II. Related Works

Neural network-based pattern recognition has been given a large amount of attention in the last few years. With increasing computational resources and new learning strategies, large neural networks can be build to better and better approximate a good mapping. Without doubt the advent of deep learning is a paradigm that already profound impact. In [1], [2], [24], a thorough overview of the main ideas and progress is given. Particularly for convolutional neural networks, has been done or years [18].

Recurrent neutral networks, also a form of deep learning, with long-term memory cells have been successfully used for phoneme recognition [12] and with more layers also for more complete speech recognition [11].

In fact, deep BLSTM networks for speech recognition has enjoyed an increased attention as a research direction on its own [9], [13], [15]. Apart from speech, multi-layer LSTM networks have also occurred for different applications, such as non-verbal, social signal classification [4], [7].

III. Neural Networks

Historically, neural networks were restricted to only 1 or maybe 2 hidden layers while recurrent neural networks have been considered unable to learn long-term dependencies. The reason for both is the so-called vanishing gradient problem [16]. This problem describes the limits of back-propagation learning [22]. In short, the error gradient decreases exponentially by each layer through which the error is propagated. Hence, after a few layers, or time-steps in case of recurrent neural networks, the gradient is nearly gone and inaccuracies in the numerical representation of a number in a computer can significant noise.

Several solutions to this problem have been proposed. For sequence processing, long short-term memory (LSTM) units [17] can act as a form of a memory cell. For multiple-layered networks, deep learning approaches have been successfully applied for many applications.

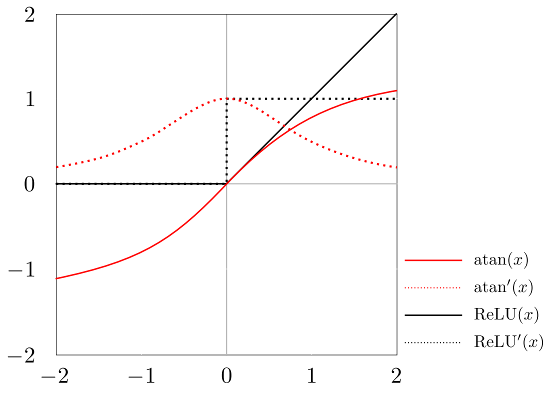

Fig. 1. A plot of the atan and ReLU activation function and their derivative.

A. *Deep Neural Networks*

Deep neural networks describe network architectures consisting of several hidden layers. To overcome the vanishing gradient problem and make use of multiple hidden layers, two approaches are possible. The first one, also called *deep believe networks* is to build the network from input-to-output, one layer at a time [3]. In each step, a new layer is trained via unsupervised learning as a restricted Bolzmann machine. This initialization phase places the weights very close to a local optimum and supervised back-propagation steps can be used to finalize the learning.

The other approach is to change the activation function in the nodes of the neural network. Using a sigmoid activation function, particularly the logistic function, has been the historical choice for network nodes, mainly because of three reasons. It runs asymptotically from 0 (or -1) to 1 to simulate a binary on/off state, the function is differentiable over the entire domain, and the derivative is fairly easy to compute. However, the derivative is close to zero for all but a few values, which is one of the reasons that the error gradient vanishes as it is propagated from the output layer to the input layer. A surprisingly easy solution to that problem in the form of rectified linear (ReLU) activation functions has recently become very popular.

B. *Rectified Linear Unit*

A rectified linear function is basically a hinge function which is zero for negative input values and the identity function otherwise as depicted in Fig. 1. The function is extremely fast to compute and has a simple derivative, 0 for negative input values and otherwise 1. Usually the function is also clipped to not exceed a large value, e.g., 20. The lack of differentiability at $x = 0$ is not an important issue, since it can be set arbitrarily to 0 or 1.

In [8] the authors argue that this simple activation function can successfully be used to train deep neural networks without any unsupervised pre-training. The 2014 ImageNet Large Scale Visual Recognition Challenge [23], for example, was won by such a a convolutional neural network with rectified linear units (ReLU) in nearly 40 layers [25].

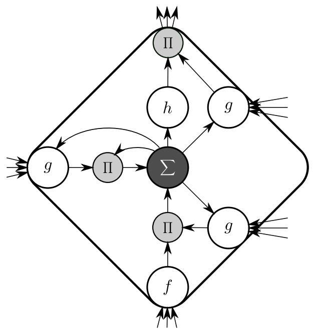

Fig. 2. Gate functions are labeled as g, input activation as f, output activation as h, multiplication nodes as $\prod$ and the core node, which realizes a summation, as $\sum$.

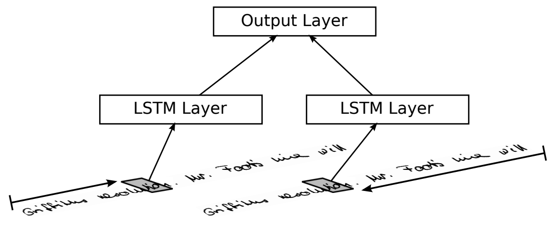

Fig. 3. The overview of the BLSTM neural network used as a baseline system.

C. *Long Short-Term memory cells*

Whereas layer-wise, unsupervised pre-training or ReLU units help to deal with many hidden layers, the vanishing gradient problem for recurrent neural networks can be successfully avoided with an LSTM architecture [17]. An LSTM unit is a second-order recurrent node, i.e. the weight of the recurrent connection, the connections toward the node and the connections leaving the nodes are not fixed but depend upon the activation of dedicated gate nodes. Such a LSTM node is depicted in Fig. 2. The core of the LSTM node, $\sum$, is a summation node with a recurrent connection of weight 1. The input from the network after being squashed by node f, the recurrent connection, as well as the squashed core value are each multiplied by the activation of the gate functions g, usually a logistic function between 0 and 1. This turns an LSTM cell basically into an memory cell, not unlike a computer memory, which can be set or reset, and is left unchanged otherwise.

A further extension is to process the sequence from both sides and aggregating the results for each position in the output layer. These kind of networks [10] deliver state-of-the-art performance for handwriting recognition and are used as a baseline system in this paper.

D. *Deep BLSTM NN*

In this paper we investigate the combination of ReLU layers and BLSTM layers for the task of continuous handwritten text recognition. Starting from the classical BLSTM neural network, we stack several feed-forward ReLU layers as well as BLSTM layers and observe training behavior and recognition accuracy. For the sake of a simpler notation, we describe a network by the type of layer in the direction from the input

TABLE I. An overview of the different neural network architectures investigated.

ID	Network name	LSTM cells	ReLU nodes	Weights (times 10^5)
baseline	I^2M^2O	200	0	1.05
A1	$I^2L^2M^2O$	200	100	1.39
A2	$I^2L^2L^2M^2O$	200	200	1.44
A3	$I^2L^2L^2L^2M^2O$	200	300	1.49
A3	$I^2L^2L^2L^2L^2M^2O$	200	400	1.54
B1	IM^2LO	200	50	1.03
B2	IM^2LLO	200	100	1.05
B3	IM^2LLLO	200	150	1.08
B3	IM^2LLLLO	200	200	1.11
C1	IM^2M^2O	400	0	2.67
C2	$IM^2L^2M^2O$	400	100	2.37
C3	$IM^2L^2L^2M^2O$	400	200	2.42
C4	$IM^2L^2L^2L^2M^2O$	400	300	2.47

to the output. Let I indicate the input layer, M (for memory) a recurrent BLSTM layer, L (for linear) a ReLU layer and O the output layer and a upper index ”2” indicates that this layer exists twice, for forward and backward processing. The standard BLSTM neural network is then described as I^2M^2O.

Note that all similar recurrent neural network can be split into a *directional* part and a *stationary* part. The directional part is the part of the network in which the processing direction of the sequence plays a role, such as the input layer and BLSTM layer. The stationary part, on the other hand, aggregates the inputs of the forward and backward sequence processing, i.e., the output layer, and does not inhibit a processing direction in itself. The network shown in Fig. 4 is written in our notation as $I^2M^2L^2L^2M^2LO$. This network contains four directional layers after the input. Once the node activations at each time step in the highest directional layer in computed separately for the forward and backward direction, the data is passed through a (stationary) ReLU layer and finally to the output layer.

Obviously, the number of possibilities to extend a recurrent deep neural network with further layers grows exponentially with the depth of the network. We focus therefore on three types of extensions. For ease of notation, we have separated the types of extensions into three groups, A, B, and C. The first one, type A, is done by adding directional ReLU layers between the input layer and the BLSTM layer. The second type of extension, type B, is to add stationary ReLU layers before the output layer. Finally, in the last tested type of extensions, type C, we add directional ReLU layers between multiple BLSTM layers.

An overview of the different deep neural network extensions, sorted according to the extension type, can be seen in Table I.

IV. Experimental Evaluation

A. Setup

The impact on using several ReLU and BLSTM layers is evaluated on continuous text lines of the IAM database [20][1].

[1]http://www.iam.unibe.ch/fki/databases

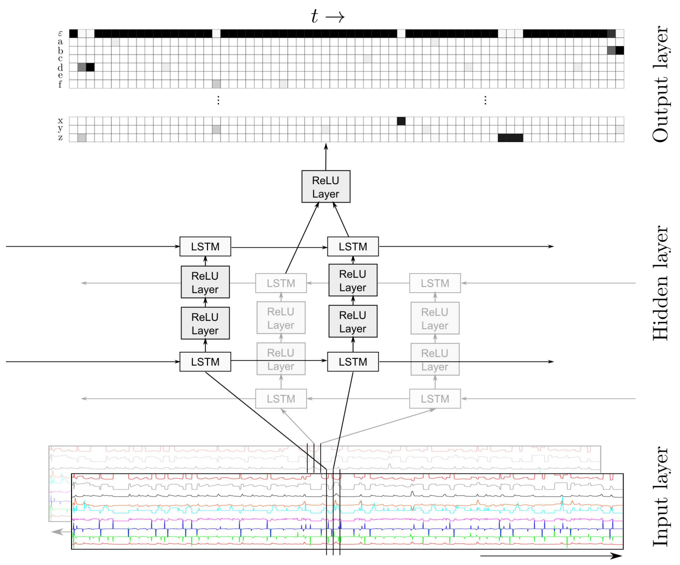

Fig. 4. A deep $IM^2L^2L^2M^L2O$ network for handwriting recognition, layers are drawn for time step t.

The database is split up into a training set of 6,161 text lines, a validation set of 920 text lines and a writer independent test set of 3421 text lines. The three sets are writer disjunct, i.e., a person who has contributed to any of the three sets did not contribute to any of the other sets.

To focus on the text line recognition, we omit all processing steps up to text line extraction. Once extracted, the text lines are normalized in order to cope with different writing styles. Finally, a sequence of feature vectors is extracted by a sliding window of one pixel width moving from left to right over the text line image. At each position of the sliding window, nine geometrical features are extracted. Three global features capture the fraction of black pixels, the center of gravity, and the second order moment. The remaining six local features consist of the position of the upper and lower contour, the gradient of the upper and lower contour, the number of black-white transitions, and the fraction of black pixels between the contours. For more details on the text line normalization operations and feature extraction, we refer to [19], [20].

For each setup, we trained 5 different, randomly initialized neural networks. Training stopped when over the course of 5 iteration no further improvement on of the character error rate the validation set was observed. The final evaluation was done regarding the word error rate on the test set using a bi-gram language model with 20 000 words.

B. Results

In Fig. 5 (and Table II) the average performances and the standard deviation for the different network topologies are shown in comparison with the reference system, with detailed numbers given in Table II. Note that the baseline, although the same system as in [10] or mentioned in [6], is given with a different word error rate (25.49%). In that setup, several neural networks were trained and the test set performance of the best single network (according to a validation set) was chosen. This article, on the other hand, is more concerned with

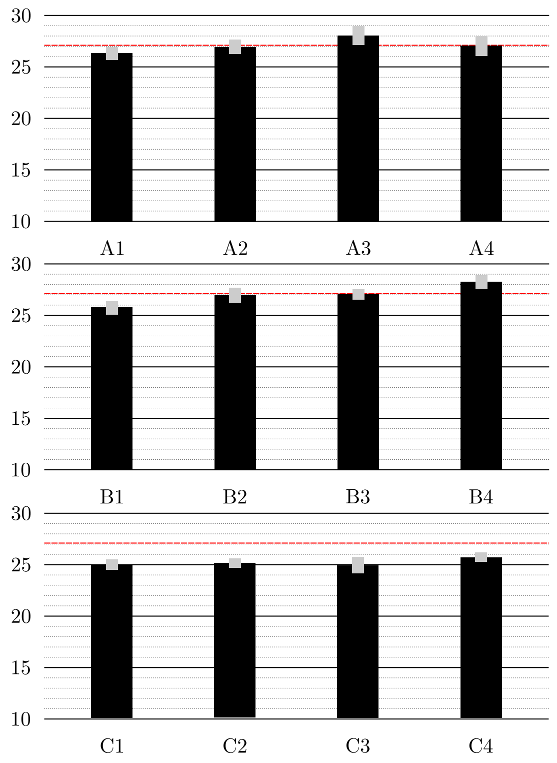

Fig. 5. Average word error rates of the test set for the networks. The baseline system in indicated by the red line.

the average performance of different topologies. Following the same strategy, it is also possible to train many networks and select the best one, yet the comparison seems more difficult.

Several observations can be made. First of all, most of the tested systems perform better than the reference system, but not all. Particularly for topology types A and B, i.e., having several ReLU layer before the BLSTM layer (type A) or before the output layer (type B) seem to get worse as the number of hidden layers increases. For the type C topology, i.e. stacking ReLU layers between BLSTM layers, the performance does not change significantly. Overall it can be seen that type C topologies perform better than the other types.

Just the number of weights, or trainable parameters, which is twice as high for type C systems, can not be the reason for that behavior, since this number or parameters increase also from A1 to A4 and B1 to B2 with degrading performance. This is further underlined by the best character error rate on the validation set during training and the final word error rate on the test set, an indication of how well the network can generalize the trained mapping. These error rates are given in Table II. Again, networks of type C perform better than networks of type A, B, or the baseline system. Networks of configuration C have many more weights, thus the networks can be better trained without losing the capability to generalize from the training set to the testing set. Finally, it seems like the number of ReLU layers – which increase from 0 to 3 in all three configurations A, B, and C – has less effect than the number of LSTM layers. Configuration A and B have one LSTM layer, while configuration C has 2 LSTM layers.

The other difference that distinguishes type C systems among these systems is the number of recurrent layers. Type A and B only have one BLSTM layer, whereas type C networks have two. From these results, this seems to be a consistent behavior.

TABLE II. An overview of the error rates of the different neural network architectures investigated.

ID	CER (val. set)	Min CER (val. set)	WER (test set)
baseline	14.28	13.48	27.11
A1	14.17	13.28	26.41
A2	14.56	13.90	27.04
A3	15.70	14.33	28.11
A4	14.73	13.90	27.10
B1	13.69	13.15	25.84
B2	14.02	13.45	27.17
B3	13.69	13.45	27.14
B4	13.92	13.47	28.29
C1	12.85	12.32	25.07
C2	13.11	12.60	25.25
C3	14.26	12.26	24.99
C4	13.31	12.56	25.77

V. Conclusion

Deep Learning has become an important trend in machine learning, backed up by astonishing results in many different areas, due to finding a workaround of the vanishing gradient problem, which renders back-propagation training very difficult for many hidden layers. One successful strategy to overcome this problem is to use ReLU units, which have a derivative of either 0 or 1 and therefore maintain the error gradient, or set it completely to 0.

On the other hand, a recent recurrent neural network architecture has overcome the vanishing gradient problem by creating completely differentiable memory cells through a composition of simple nodes, implementing sigmoid activation functions, addition, and multiplication.

In this paper we investigate the combination of both ideas for the task of handwriting recognition. The investigated systems are BLSTM layers to deal with long-term dependencies encountered in handwritten text. This system is extended by adding layers between input and output layer. Depending upon where, and what kind of layer, has been added, we tested three different types of networks.

The main finding is that we can decrease the error rate of this high-performing system by adding several layers. However, it appears that simply adding layers to increase the number of weights is not directly correlated with the recognition rate. Instead, the type of layer seems to have a big impact, with recurrent layers being more successful in decreasing the error rate than feed-forwards ReLU layers.

In the future we will further extensions and also more complex topologies. With an increase of parameters, the need for further training data increases as well, but also the potential for generalization. Hence, experiments on training one deep network across several databases seem also to be a natural way to go from here.

Acknowledgment

This work is supported in part by the CREST project from Japan Society for the Promotion of Science (JSPS).

The computation was mainly carried out using the computer facilities at the Research Institute for Information Technology, Kyushu University, Japan.

References

[1] Itamar Arel, Derek C. Rose, and Thomas P. Karnowski. Deep Machine Learning – A New Frontier in Artificial Intelligence Research. *Computational Intelligence Magazine, IEEE*, 5(4):13–18, 2010.

[2] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. Deep Learning. Book in preparation for MIT Press, 2014.

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-wise Training of Deep Networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in neural information processing systems 19*, pages 153–160. MIT Press, Cambridge, MA, 2007.

[4] Raymond Brueckner and Björn Schuller. Social Signal Classification using Deep BLSTM Recurrent Neural Networks. In *Int'l Conf. on Acoustics, Speech and Signal Processing*, pages 4823–4827, 2014.

[5] Horst Bunke. Recognition of Cursive Roman Handwriting - Past, Present and Future. In *Proc. 7th Int'l Conf. on Document Analysis and Recognition*, volume 1, pages 448–459, August 2003.

[6] Salvador España-Boquera, Maria José Castro-Bleda, Jorge Gorbe-Moya, and Francisco Zamora-Martinez. Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(4):767–779, 2011.

[7] Volkmar Frinken, Francisco Zamora-Martínez, Salvador España Boquera, Maria José Castro-Bleda, Andreas Fischer, and Horst Bunke. Long-Short Term Memory Neural Networks Language Modeling for Handwriting Recognition. In *21st Int'l Conf. on Pattern Recognition*, pages 701–704, 2012.

[8] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep Sparse Rectifier Neural Networks. In *14th Int'l Conf. on Artificial Intelligence and Statistics*, pages 315–323.

[9] Alex Graves and Navdeep Jaitly. Towards End-to-End Speech Recognition with Recurrent Neural Networks. In *31st Int'l Conf. on Machine Learning*, pages 1764–1772, 2014.

[10] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

[11] Alex Graves, Abdel rahmen Mohamed, and Geoffrey E. Hinton. Speech Processing with Deep Recurrent Neural Networks. In *ICASSP*, 2013.

[12] Alex Graves and Jürgen Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM Networks. In *Int'l Joint Conf. on Neural Networks*, volume 4, pages 2047–2052, 2005.

[13] Haşim Hak, Senior Andrew, and Françoise Beaufays. Long Short-Term Memory Recurrent Neural Network Architecture for Large Scale Acoustic Modeling. In *Interspeech*, pages 338–342, 2014.

[14] Mahdi Hamdani, Patrick Doetsch, Michal Kozielski, and Amr El-Desoky Mousaand Hermann Ney. The RWTH Large Vocabulary Arabic Handwriting Recognition System. In *11th IAPR Int't Workshop on Document Analysis Systems*, pages 111–115, 2014.

[15] Awni Y. Hannun, Carl Case, Jared Casper, Bryan C. Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *Computing Research Repository*, 2014.

[16] Sepp Hochreiter, Yoshua Bengio, and Paolo Frasconi Jürgen Schmidhuber. *A Field Guide to Dynamical Recurrent Neural Networks*, chapter Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies, pages 237–243. Wiley-IEEE Press, 2001.

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[18] Yann LeCun, L'eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 1998.

[19] Urs-Victor Marti and Horst Bunke. Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System. *Int'l Journal of Pattern Recognition and Artificial Intelligence*, 15:65–90, 2001.

[20] Urs-Victor Marti and Horst Bunke. The IAM-Database: An English Sentence Database for Offline Handwriting Recognition. *Int'l Journal on Document Analysis and Recognition*, 5:39–46, 2002.

[21] Réjean Plamondon and Sargur N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[22] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, 323:533–536, 1986.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[24] Jürgen Schmidhuber. Deep Learning in Neural Networks: An Overview. *Neural Networks*, 61:85–117, 2015.

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Large Scale Visual Recognition Challenge 2014 Workshop*, 2014.

[26] Francisco Zamora-Martinez, Volkmar Frinken, Salvador España Boquera, María José Castro-Bleda, Andreas Fischer, and Horst Bunke. Neural Network Language Models for off-line Handwriting Recognition. *Pattern Recognition*, 47:1642–1652, 2014.