Improving gradient-based LSTM training for offline handwriting recognition by careful selection of the optimization method

Martin Schall
Institute for Optical Systems
University of Applied Sciences
Constance, Germany
Email: martin.schall@htwg-konstanz.de

Marc-Peter Schambach
Siemens Postal, Parcel &
Airport Logistics GmbH
Constance, Germany
Email: marc-peter.schambach@siemens.com

Matthias O. Franz
Institute for Optical Systems
University of Applied Sciences
Constance, Germany
Email: mfranz@htwg-konstanz.de

***Abstract*—Recent years have seen the proposal of several different gradient-based optimization methods for training artificial neural networks. Traditional methods include steepest descent with momentum, newer methods are based on per-parameter learning rates and some approximate Newton-step updates. This work contains the result of several experiments comparing different optimization methods. The experiments were targeted at offline handwriting recognition using hierarchical subsampling networks with recurrent LSTM layers. We present an overview of the used optimization methods, the results that were achieved and a discussion of why the methods lead to different results.**

***Index Terms*—offline handwriting recognition; recurrent neural network; long-short-term-memory; connectionist temporal classification; gradient-based learning; adadelta; rmsprop**

I. Introduction

Advances in the field of unconstrained and segmentation-free offline handwriting recognition using artificial neural networks have been considerable in the last years [1] and complete systems for this task have been published [2]. Offline handwriting recognition is in use in applications such as postal automation, banking and historical document analysis.

State of the art solutions for Latin script offline handwriting recognition are based on Multi-Dimensional Long-Short-Term-Memory *MDLSTM* [3] [4] recurrent neural networks organized as hierarchical subsampling networks [5]. Such networks can be trained for sequence classification using Connectionist Temporal Classification *CTC* [6]. CTC allows the training of networks for segmentation-free sequence classification without knowledge about the location of contained labels, based only on knowledge about the correct label sequence.

Newton's method can be used to determine an individual step-size for each parameter during backpropagation training of the artificial neural network [7]. Using Newton's method leads to fast convergence rates but requires the calculation of second-order derivatives of the error function. Since the calculation of second-order information is computationally expensive during backpropagation-based training, methods like AdaDelta [8] try to approximate it using only first-order information.

RProp [9] [10] provides an individual learning rate per parameter using only the changes in the sign of the partial derivative, similar to the Manhattan rule. RMSProp [11] improves on this concept by generalizing to mini-batch training variants of the backpropagation algorithm. RMSProp does so by normalizing the gradient using the rolling mean value of the previous first-order derivatives.

This work provides an overview and comparison of contemporary gradient-based optimization methods for training hierarchical subsampling MDLSTM-networks using CTC. All experiments were done using the IAM offline handwriting database [12]. It is meant as a guide for practitioners in the field of offline handwriting recognition. In addition, this work includes theoretical interpretations of the observed results.

The paper starts by describing the used network topology in section II, the investigated optimization methods in section III and the experiments executed in section IV. Section V presents the results of the experiments and section VI discusses the problems arising with the optimization methods. Section VII concludes the paper.

II. Network

The network topology was identical for all experiments and is based on the hierarchical subsampling network using MDLSTM-cells applied for Arabic handwriting recognition [2] [5]. While the network topology was unchanged, the hyperparameters and sizes of the neuron layers were modified. The exact network topology, beginning at the network input, and hyperparameters are described in table I.

The LSTM variant [13] used in the comparisons includes forget gates, peephole connections, bias values and the full gradient for backpropagation. The fully connected feedforward neurons had no bias, except for the very last neuron layer. The non-linearities are the standard logistic sigmoid $\phi(x) = \frac{1}{1+e^{-x}}$ for the LSTM gates and the hyperbolic tangent $tanh(x)$ for all other activations. The network consists of a total of 148799 parameters, all of which were initialized

TABLE I
NETWORK TOPOLOGY USED FOR THE EXPERIMENTS

Type of layer	Configuration
Input image	Grayscale; 81 pixel in height
Subsampling	2×3 (width $\times$ height)
MDLSTM	2 cells per scan direction
Subsampling	2×3 (width $\times$ height)
Fully connected feedforward	6 neurons; no bias
MDLSTM	10 cells per scan direction
Subsampling	2×3 (width $\times$ height)
Fully connected feedforward	20 neurons; no bias
MDLSTM	50 cells per scan direction
Fully connected feedforward	79 neurons; with bias
Collapse	
Softmax	
CTC	78 glyph labels; 1 blank label

drawing from a random uniform distribution in the interval $[-0.1; +0.1]$.

III. METHODS

The following paragraphs outline the gradient-based optimization methods: Steepest descent with momentum, RMSProp and AdaDelta. In all equations, g_t is the gradient at time t and δx_t the parameter updates at time t. μ is always the learning rate, α the decay rate and β the dampening factor. All variables are initialized to zero if not otherwise defined.

Algorithm 1 Steepest descent with momentum

$$\delta x_t = (\mu \times g_t) + (\alpha \times \delta x_{t-1})$$

Algorithm 1 describes the steepest descent optimizer with a simple momentum term added. It scales the first-order derivative of the error function by a constant learning rate, thus generating parameter updates that are directly proportional to the gradient. The added momentum term prevents the optimizer from following jitters in the error function along the current path. Figuratively speaking, if the optimization process is a ball moving down the error landscape, momentum changes the gradient from being a vector of movement to a vector of force applied to the ball.

Algorithm 2 RMSProp

$$E[g^2]_t = ((1-\alpha) \times g_t^2) + (\alpha \times E[g^2]_{t-1})$$
$$\delta x_t = \mu \times \frac{g_t}{\sqrt{E[g^2]_t}}$$

RMSProp, outlined in Algorithm 2, is a generalization of RProp that allows mini-batch training. Both only take the sign of the gradient into account but determine the step size of parameter updates independently from the absolute value of the gradient. RMSProp does so by using a rolling mean of the gradient for normalization. It effectively allows the user to choose the actual step size of parameter updates as a hyperparameter.

Algorithm 3 describes AdaDelta, which uses an approximation of the diagonal values of the Hessian matrix to do quasi-Newton updates. AdaDelta provides per-dimension step sizes and basically removes the need to manually choose a learning rate. The idea behind AdaDelta is outlined in the according publication [8], calculating the parameter updates based on the inverse Hessian as $\frac{\delta x}{g}$. Since both the total parameter updates δx and the total gradient g for the Newton step are unknown, they are approximated using a rolling mean of the last values. A variant of AdaDelta adds an additional learning rate μ, which should be chosen as a value near 1.0 since the unmodified AdaDelta implies a global learning rate of 1.0.

Algorithm 3 AdaDelta with additional learning rate

$$E[g^2]_t = ((1-\alpha) \times g_t^2) + (\alpha \times E[g^2]_{t-1})$$
$$u_t = g_t \times \frac{\sqrt{E[\delta x^2]_{t-1} + \beta}}{\sqrt{E[g^2]_t + \beta}}$$
$$E[\delta x^2]_t = ((1-\alpha) \times u_t^2) + (\alpha \times E[\delta x^2]_{t-1})$$
$$\delta x_t = \mu \times u_t$$

When gradient clipping was applied, only the error signal that is transported from a LSTM layer to its predecessor was truncated. Recalling the network topology defined in Table I, this concerns only the transition between the last two MDLSTM layers and their previous fully connected feedforward layers. The error signal was hard clipped to be within the interval $[-1; +1]$.

IV. EXPERIMENTS

All experiments were carried out using the IAM offline handwriting database [12] with the images being rescaled to 8-bit grayscale and fixed 81 pixel in height with a variable width. A random subset of 90% (86809) samples were used for training and 5% (4822) each for validation and evaluation. A sample of the IAM database is shown in figure 1.

Fig. 1. Example of the IAM database

The network is specified in section II and the target function of supervised training was CTC with 78 visible character classes of the IAM database. No normalization of labels was applied.

If not otherwise noted, the training was done using mini-batch updates of size 8 and the full non-clipped gradient. The gradients within a mini-batch were summed, but not normalized afterwards. The training samples were processed in a random permutation for each training epoch. The experiments used early stopping until the validation error rate did not improve for 5 epochs.

The following individual experiments were conducted in this work:

1) Steepest descent with momentum and full gradient.
2) Steepest descent with momentum and gradient clipping.

3) RMSProp.
4) AdaDelta without additional learning rate.
5) AdaDelta with additional learning rate.

The hyperparameters were chosen on basis of previous experiments with this network architecture and the IAM database. The hyperparameters have proven to be suitable for training this network for offline handwriting recognition.

Error rate was measured in terms of Character Error Rate *CER* at the end of each training epoch. CER is defined as the percentage $CER(y,z) = \frac{100 \times ED(y,z)}{|y|}$. It measures the part of the edit-distance [14] $ED(y,z)$ between the correct label string y and the decoded network output z in relation to the length $|y|$ of the correct label. The CER of these experiments are averages over all samples within the training set or validation set respectively.

V. Results

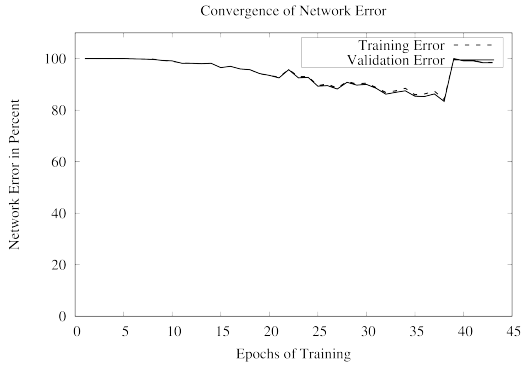

Fig. 2. Steepest descent with $\mu = 1e^{-4}$ and $\alpha = 0.9$

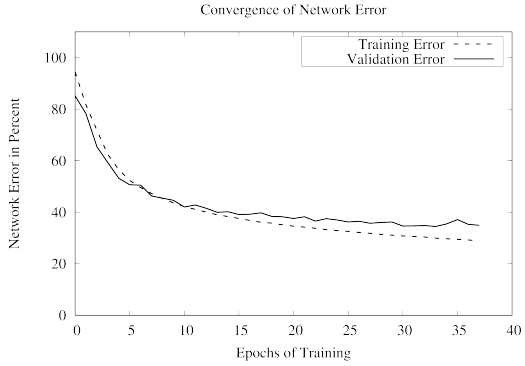

Fig. 3. Steepest descent with $\mu = 1e^{-4}$ and $\alpha = 0.9$ (gradient clipping)

Figures 2 and 3 show the convergence of the CER during training using steepest descent with momentum. Hyperparameters were $\mu = 1e^{-4}$ and $\alpha = 0.9$. The training using the full non-clipped training did not converge to acceptable error rates as can be seen in figure 2. The use of gradient clipping did improve the convergence of CER, see figure 3, during training. The convergence rate is still lower than with RMSProp or AdaDelta, however.

As can be seen in figure 2, the CER initially decreases for some epochs but then started increasing again and stabilizes at 99%.

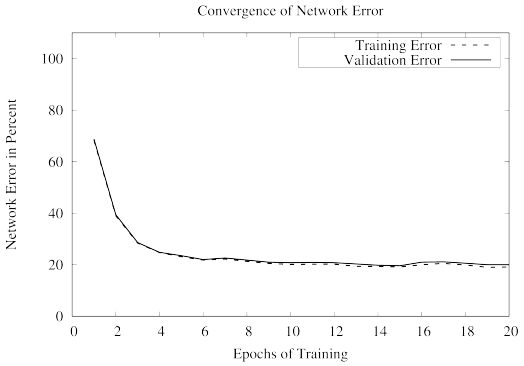

Fig. 4. RMSProp with $\mu = 1e^{-3}$ and $\alpha = 0.9$

Figure 4 shows the convergence of the error rate using RMSProp with $\mu = 1e^{-3}$ and $\alpha = 0.9$. It shows a faster convergence rate than steepest descent with gradient clipping and achieves a lower error rate.

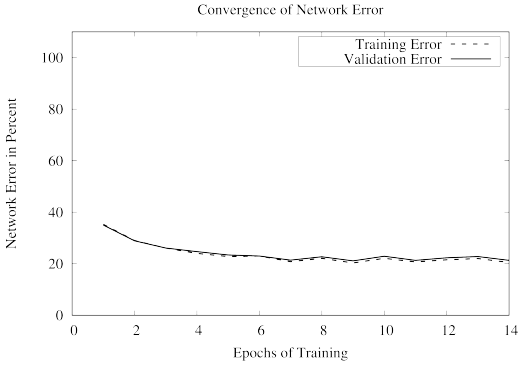

Fig. 5. AdaDelta with $\mu = 1$, $\alpha = 0.95$ and $\beta = 1e^{-6}$

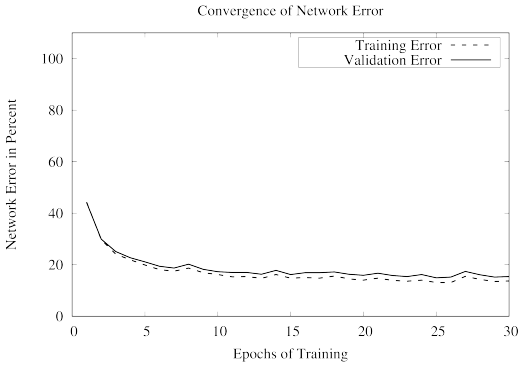

Fig. 6. AdaDelta with $\mu = 0.5$, $\alpha = 0.95$ and $\beta = 1e^{-6}$

Figures 5 and 6 contain the results using AdaDelta. Both use the hyperparameters $\alpha = 0.95$ and $\beta = 1e^{-6}$. The experiment described in figure 5 used a learning rate of $\mu = 1$, thus corresponds to the original work by the authors of AdaDelta [8]. Figure 6 uses an additional learning rate of $\mu = 0.5$, which reduces the convergence rate by the same factor. Using an additional learning rate proved to result in lower final error rates.

The fastest convergence rate in these experiments was achieved using AdaDelta with $\mu = 1$, $\alpha = 0.95$ and $\beta = 1e^{-6}$, the lowest error rate with AdaDelta and $\mu = 0.5$.

VI. Discussion

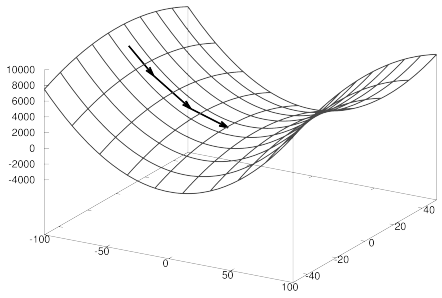

Fig. 7. Exemplary saddle point of an error function in a two-dimensional parameter space

In the following section, we discuss possible reasons for why the compared optimization methods behave differently in terms of convergence of network error. Recent work [15] [16] has shown that saddle points in the error function tend to be a major problem while training artificial neural networks. Other potential problems arise from the interaction between Backpropagation-Through-Time *BPTT* [17] and a momentum term in the optimization method. Figure 7 shows an error function with a saddle point that highlights the different behavior of the three optimization methods in this situation. Saddle points in the error function are interesting because they both consist of steep and shallow parts but the direction of any gradient descent optimization will change on a saddle point. Differences arise as soon as the gradient descent optimization moves from the steep flank of the error function to somewhere near the saddle point.

Consider Algorithm 1 (steepest descent with momentum): while descending down the steep part of the error function, the momentum will increase accordingly. The absolute value of the gradient will be very small in comparison to the gradient on the steep part, which results in only a small impact of the current gradient when updating the parameters. In this exemplary case, gradient descent will overshoot the saddle point instead of following the gradient to the decreasing error values.

RMSProp and AdaDelta, see algorithms 2 and 3, tackle this problem by normalizing the parameter updates with the expectation value of the absolute gradient. The expectation value is again large after traversing the steep flank of the error function. After normalization, the relatively small gradient near the saddle point will be even smaller. The actual per-parameter learning rate is decreased and thus the gradient descent slows down near the saddle point. An increase in the per-parameter learning rate will occur as soon as the expectation value of the gradient has adapted to the small gradient value. This behavior allows for a change of direction near saddle points without overshooting it.

Another potential problem arises in the BPTT algorithm in combination with training samples of variable sizes, e.g. different sizes of the input images. BPTT calculates the gradient by virtually unrolling the recurrent network into a feedforward network. Training samples of longer sequences will result in 'deeper' unrolled networks. Parameters of recurrent layers are shared in the unrolled network and thus their gradients need to be summed again before updating their parameters. Similar to mini-batch training, the gradients summed up to obtain the accumulated gradient for the recurrent layer.

Steepest descent with momentum and full gradient is prone to an effect similar to the 'exploding gradient': The absolute value of the gradient is directly proportional to the sequence length. For a long sequence, the momentum will be accumulated, while short sequences have little impact on gradient descent. This again leads to overshooting of minimum points or saddle points. This 'exploding gradient' explains why gradient clipping is effective for steepest descent, as can be seen in the convergence rates of figures 2 and 3.

VII. Conclusion

This work presents the results of several experiments training hierarchical subsampling networks using LSTM-cells for offline handwriting recognition. Three different gradient-based optimization methods were used: steepest descent, RMSProp and AdaDelta. Steepest descent was tested both with the full, non-clipped, gradient and with gradient clipping.

The results show a better convergence rate for RMSProp and AdaDelta than for normal steepest descent. Both RMSProp and AdaDelta are easy to implement and cause only a linear overhead in memory consumption which makes them reasonable choices for practitioners. AdaDelta with a reduced learning rate of 0.5 achieved the lowest error rate of all experiments.

Section VI rationales why steepest descent shows a worse behavior than RMSProp or AdaDelta in the presence of saddle points or when using BPTT for recurrent neural networks. Saddle points can be expected [16] in high-dimensional non-convex optimization problems such as offline handwriting recognition. BPTT is the establish method for gradient-based training of recurrent neural networks and as such, problems arising out of the interaction between BPTT and the optimization method should be considered.

Based on the observations during the experiments and the following reflections, the authors are suggesting to use

AdaDelta for training LSTM networks for offline handwriting recognition. Newer optimization methods, such as Adam [18], were not taken into consideration but may give even results.

Acknowledgment

The authors would like to thank the Siemens Postal, Parcel & Airport Logistics GmbH for funding this work. The authors would also like to thank Jörg Rottland for proof-reading this work and his valuable suggestions.

References

[1] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[2] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, 2012.

[3] A. Graves, S. Fernandez, and J. Schmidhuber, "Multi-Dimensional Recurrent Neural Networks," IDSIA/USI-SUPSI, Tech. Rep., 2007.

[4] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM." *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[5] A. Graves and J. Schmidhuber, "Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks," in *Advances in Neural Information Processing Systems 21, NIPS'21*, 2008, pp. 545–552.

[6] A. Graves, S. Fernandez, F. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proceedings of the 23rd international conference on Machine Learning*. ACM Press, 2006, pp. 369–376.

[7] T. Schaul, S. Zhang, and Y. LeCun, "No More Pesky Learning Rates," *Journal of Machine Learning Research*, vol. 28, no. 2, pp. 343–351, 2013.

[8] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," p. 6, 2012.

[9] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," *IEEE International Conference on Neural Networks*, 1993.

[10] C. Igel and M. Hüsken, "Improving the Rprop learning algorithm," in *Proceedings of the Second International Symposium on Neural Computation*, 2000, pp. 115–121.

[11] Y. N. Dauphin, J. Chung, and Y. Bengio, "RMSProp and equilibrated adaptive learning rates for non-convex optimization," Tech. Rep., 2014.

[12] U. V. Marti and H. Bunke, "The IAM-database: An English sentence database for offline handwriting recognition," *International Journal on Document Analysis and Recognition*, vol. 5, no. 1, pp. 39–46, 2003.

[13] K. Greff, R. K. Srivastava, J. Koutník, B. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," IDSIA/USI-SUPSI, Tech. Rep., 2015.

[14] R. A. Wagner and M. J. Fischer, "The String-to-String Correction Problem," *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[15] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun, "The Loss Surfaces of Multilayer Networks," *Aistats*, vol. 38, pp. 192—-204, 2015. [Online]. Available: http://arxiv.org/abs/1412.0233

[16] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *arXiv*, pp. 1–14, 2014. [Online]. Available: http://arxiv.org/abs/1406.2572

[17] R. Rojas, "The Backpropagation Algorithm," in *Neural Networks*. Springer, 1996, pp. 151–184.

[18] D. P. Kingma and J. L. Ba, "Adam: a Method for Stochastic Optimization," *International Conference on Learning Representations*, pp. 1–13, 2015.