Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de

Yoshua Bengio
Dept. Informatique et Recherche Opérationnelle
Université de Montréal, CP 6128, Succ. Centre-Ville,
Montréal, Québec, Canada, H3C 3J7
bengioy@iro.umontreal.ca

Paolo Frasconi
Dept. of Systems and Computer Science
University of Florence
via di Santa Marta 3, 50139 Firenze (Italy)
paolo@dsi.unifi.it

Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch

1 Introduction

Recurrent networks (crossreference Chapter 12) can, in principle, use their feedback connections to store representations of recent input events in the form of activations. The most widely used algorithms for learning *what* to put in short-term memory, however, take too much time to be feasible or do not work well at all, especially when minimal time lags between inputs and corresponding teacher signals are long. Although theoretically fascinating, they do not provide clear *practical* advantages over, say, backprop in feedforward networks with limited time windows (see crossreference Chapters 11 and 12). With conventional "algorithms based on the computation of the complete gradient", such as "Back-Propagation Through Time" (BPTT, e.g., [22, 27, 26]) or "Real-Time Recurrent Learning" (RTRL, e.g., [21]) error signals "flowing backwards in time" tend to either (1) blow up or (2) vanish: the temporal evolution of the backpropagated error exponentially depends on the size of the weights [11, 6]. Case (1) may lead to oscillating weights, while in case (2) learning to bridge long time lags takes a prohibitive amount of time, or does not work at all.

In what follows, we give a theoretical analysis of this problem by studying the asymptotic behavior of error gradients as a function of time lags. In Section 2, we consider the case of standard RNNs and derive the main result using the approach first proposed in [11]. In Section 3, we consider the more general case of adaptive dynamical systems, which include, besides standard RNNs, other recurrent architectures based on different connectivities and choices of the activation function (e.g., RBF or second order connections). Using the analysis reported in [6] we show that one of the following two undesirable situations necessarily arise: either the system is unable to robustly store past information about its inputs, or gradients vanish exponentially. Finally, in Section 4 we shortly review alternative optimization methods and architectures that have been suggested to improve learning in the presence of long-term dependencies.

*In S. C. Kremer and J. F. Kolen, eds., A Field Guide to Dynamical Recurrent Neural Networks. IEEE press, 2001

2 Exponential error decay

Gradients of the error function

The results we are going to prove hold regardless of the particular kind of cost function used (as long as its continuous in the output) and regardless of the particular algorithm which is employed to compute the gradient. Here we shortly explain how gradients are computed by the standard BPTT algorithm (e.g., [27], see also crossreference Chapter 14 for more details) because its analytical form is better suited to the forthcoming analyses.

The error at time t is denoted by $E(t)$. Considering only the error at time t, output unit k's error signal is

$$\delta_k(t) = \frac{\partial E(t)}{\partial net_k(t)}$$

and some non-output unit j's backpropagated error signal at time $\tau < t$ is

$$\delta_j(\tau) = f_j'(net_j(\tau)) \left(\sum_i w_{ij} \ \delta_i(\tau + 1) \right),$$

where

$$net_i(\tau) = \sum_j w_{ij} \ a_j(\tau - 1)$$

is unit i's current net input,

$$a_i(\tau) = f_i(net_i(\tau))$$

is the activation of a non-input unit i with differentiable transfer function f_i, and w_{ij} is the weight on the connection from unit j to i. The corresponding contribution to w_{jl}'s total weight update is $\eta \ \delta_j(\tau) \ a_l(\tau - 1)$, where η is the learning rate, and l stands for an arbitrary unit connected to unit j.

Error path integral

Suppose we have a fully connected net whose non-input unit indices range from 1 to n. Let us focus on local error flow from output unit k to arbitrary unit v (later we will see that the analysis immediately extends to global error flow). The error occurring at k at time step t is propagated "back in time" for $t - s$ time steps, to an arbitrary unit v at time $s < t$. This scales the error by the following factor:

$$\frac{\partial \delta_v(s)}{\partial \delta_k(t)} = \begin{cases} f_v'(net_v(t-1)) \;\; w_{kv} & t-s=1 \\ f_v'(net_v(s)) \left(\sum_{l=1}^{n} \frac{\partial \delta_l(s+1)}{\partial \delta_k(t)} \;\; w_{lv} \right) & t-s>1 \end{cases} . \tag{1}$$

In order to solve the above equation, we will expand it by unrolling over time (as done for example in deriving BPTT). In particular, for $s < \tau < t$ let l_τ denote the index of a generic non input unit in the replica of the network at time τ. Moreover, let $l_s = v$ and $l_t = k$. We obtain:

$$\frac{\partial \delta_v(s)}{\partial \delta_k(t)} = \sum_{l_{t-1}=1}^{n} \ldots \sum_{l_{s+1}=1}^{n} \left(w_{l_t l_{t-1}} \left(\prod_{\tau=t-1}^{s+1} f_{l_\tau}'(net_{l_\tau}(\tau)) \;\; w_{l_\tau l_{\tau-1}} \right) f_{l_s}'(net_{l_s}(s)) \right) \tag{2}$$

(proof by induction).

It can be immediately shown that if the local error vanishes, then the *global* error vanishes too. To see this compute

$$\sum_{k \in O} \frac{\partial \delta_v(s)}{\partial \delta_k(t)}$$

where O denotes the set of output units.

Intuitive explanation of equation (2)

If

$$\left| f_{l_\tau}'(net_{l_\tau}(\tau)) \;\; w_{l_\tau l_{\tau-1}} \right| \; > \; 1.0$$

for all τ then the largest product increases exponentially with $t-s-1$. That is, the error blows up, and conflicting error signals arriving at unit v can lead to oscillating weights and unstable learning (for error blow-ups or bifurcations see also [19, 2, 8]). On the other hand, if

$$\left| f_{l_\tau}'(net_{l_\tau}(\tau)) \;\; w_{l_\tau l_{\tau-1}} \right| \; < \; 1.0$$

for all τ, then the largest product *decreases* exponentially with $t-s-1$. That is, the error vanishes, and nothing can be learned in acceptable time.

If f_{l_τ} is the logistic sigmoid function, then the maximal value of f_{l_τ}' is 0.25. If $a_{l_{\tau-1}}$ is constant and not equal to zero, then the size of the gradient $\left| f_{l_\tau}'(net_{l_\tau}) w_{l_\tau l_{\tau-1}} \right|$ takes on maximal values where

$$w_{l_\tau l_{\tau-1}} = \frac{1}{a_{l_{\tau-1}}} \coth\left(\frac{1}{2} net_{l_\tau} \right),$$

the size of the derivative goes to zero for $\left|w_{l_\tau l_{\tau-1}}\right| \to \infty$, and it is less than 1.0 for $\left|w_{l_\tau l_{\tau-1}}\right| < 4.0$ (e.g., if the absolute maximal weight value w_{max} is smaller than 4.0). Hence with conventional logistic sigmoid transfer functions, the error flow tends to vanish as long as the weights have absolute values below 4.0, especially in the beginning of the training phase. In general the use of larger initial weights does not help though — as seen above, for $\left|w_{l_\tau l_{\tau-1}}\right| \to \infty$ the relevant derivative goes to zero "faster" than the absolute weight can grow (also, some weights may have to change their signs by crossing zero). Likewise, increasing the learning rate does not help either — it does not change the ratio of long-range error flow and short-range error flow. BPTT is too sensitive to recent distractions. Note that since the summation terms in equation (2) may have different signs, increasing the number of units n does not necessarily increase error flow.

Weak upper bound for scaling factor

The following, slightly extended vanishing error analysis also takes n, the number of units, into account. For $t - s > 1$, formula (2) can be rewritten as

$$\left(W_k^T\right)^T \; F'(t-1) \; \left(\prod_{\tau=t-2}^{s+1} \; W \; F'(\tau) \right) \; W_v \; f_v'\left(net_v(s)\right),$$

where the weight matrix W is defined by $[W]_{ij} := w_{ij}$, v's outgoing weight vector W_v is defined by $[W_v]_i := [W]_{iv} = w_{iv}$, k's incoming weight vector W_k^T is defined by $[W_k^T]_i := [W]_{ki} = w_{ki}$, and $F'(t)$ is the diagonal matrix of first order derivatives defined as: $[F'(t)]_{ij} := 0$ if $i \neq j$, and $[F'(t)]_{ij} := f_i'(net_i(t))$ otherwise. Here T is the transposition operator, $[A]_{ij}$ is the element in the i-th column and j-th row of matrix A, and $[x]_i$ is the i-th component of vector x.

Using a matrix norm $\|.\|_A$ compatible with vector norm $\|.\|_x$, we define

$$f'_{max} := \max_{\tau=t-1,\ldots,s}\{\|F'(\tau)\|_A\}.$$

For $\max_{i=1,\ldots,n}\{|x_i|\} \leq \|x\|_x$ we get $\left|x^T y\right| \leq n \; \|x\|_x \; \|y\|_x$. Since

$$|f_v'(net_v(s))| \leq \|F'(s)\|_A \leq f'_{max},$$

we obtain the following inequality:

$$\left|\frac{\partial \delta_v(s)}{\partial \delta_k(t)}\right| \leq n \; (f'_{max})^{t-s} \; \|W_v\|_x \; \|W_k^T\|_x \; \|W\|_A^{t-s-2} \leq n \; \left(f'_{max} \; \|W\|_A\right)^{t-s}.$$

This inequality results from

$$\|W_v\|_x \;=\; \|We_v\|_x \;\leq\; \|W\|_A \, \|e_v\|_x \;\leq\; \|W\|_A$$

and

$$\|W_k^T\|_x \;=\; \|e_k W\|_x \;\leq\; \|W\|_A \, \|e_k\|_x \;\leq\; \|W\|_A,$$

where e_k is the unit vector whose components are 0 except for the k-th component, which is 1. Note that this is a weak, extreme case upper bound — it will be reached only if all $\|F'(\tau)\|_A$ take on maximal values, and if the contributions of all paths across which error flows back from unit k to unit v have the same sign. Large $\|W\|_A$, however, typically result in small values of $\|F'(\tau)\|_A$, as confirmed by experiments (see, e.g., [11]).

For example, with norms

$$\|W\|_A \;:= \max_r \sum_s |w_{rs}|$$

and

$$\|x\|_x := \max_r |x_r| \,,$$

we have $f'_{max} = 0.25$ for the logistic sigmoid. We observe that if

$$|w_{ij}| \leq w_{max} < \frac{4.0}{n} \quad \forall i, j,$$

then $\|W\|_A \;\leq n \;\; w_{max} < 4.0$ will result in exponential decay; by setting $\lambda := \left(\frac{n w_{max}}{4.0}\right) < 1.0$, we obtain

$$\left|\frac{\partial \delta_v(s)}{\partial \delta_k(t)}\right| \;\leq\; n \; \lambda^{t-s}.$$

We refer to Hochreiter's thesis [11] for more details.

3 Dilemma: Avoiding gradient decay prevents long-term latching

In Bengio *et al.*'s paper [6], the analysis of the problem of gradient decays is generalized to parameterized dynamical systems (hence including second order and other recurrent architectures). The main theorem shows that a

sufficient condition to obtain gradient decay is also a necessary condition for the system to robustly store discrete state information for the long-term. In other words, when the weights and the state trajectory are such that the network can "latch" on information in its hidden units (i.e., represent long-term dependencies), the problem of gradient decay is obtained. When the long-term gradients decay exponentially, it is very difficult to learn such long-term dependencies because the total gradient is the sum of long-term and short-term influences and the short-term influences then completely dominate the gradient.

This result is based on a decomposition of the state-space of hidden units in two types of regions: one where gradients decay and one where it is not possible to robustly latch information. Let $y(t)$ denote the n-dimensional state vector at time t (for example, the vector $[net_1(t), \ldots, net_n(t)]$ when considering a standard first-order recurrent network) and let $y(t) = M(y(t-1))$ be the map from the state at time $t-1$ to t for the autonomous (without inputs) dynamical system. The above decomposition is expressed in terms of the condition $|M'| > 1$ (no robust latching possible) or $|M'| < 1$ (gradient decay), where $|M'|$ is the norm of the Jacobian (matrix of partial derivatives) of the map M. The analysis focuses on the basins of attraction of attractors of M in the domain of $y(t)$ (or manifolds within that domain). In particular, the analysis is concerned with so-called hyperbolic attractors, which are locally stable (but need not be fixed points) and where the eigenvalues of M' are less than 1 in absolute value. If the state (or a function of it) remains within a certain region of space (versus another region) even in the presence of perturbations (such as noise in the inputs) then it is possible to store at least one bit of information for arbitrary durations.

In regions where $|M'| > 1$ it can be shown that arbitrarily small perturbations (for example due to the inputs) can eventually kick the state out of a basin of attraction [18] (see the sample trajectory on the right of Figure 1). In regions where $|M'| < \lambda < 1$ there is a level of perturbation (depending on λ) below which the state will remain in the basin of attraction (and will gradually get closer to a certain volume around the attractor — see left of Figure 1). For this reason we call this condition "information latching," since it allows to store discrete information for arbitrary duration in the state variable $y(t)$.

Unfortunately, in the regions where $|M'| < 1$ (where one can latch information) one can also show that gradients decay. The argument is similar to the one developed in the previous section. The partial derivative of $y(t)$

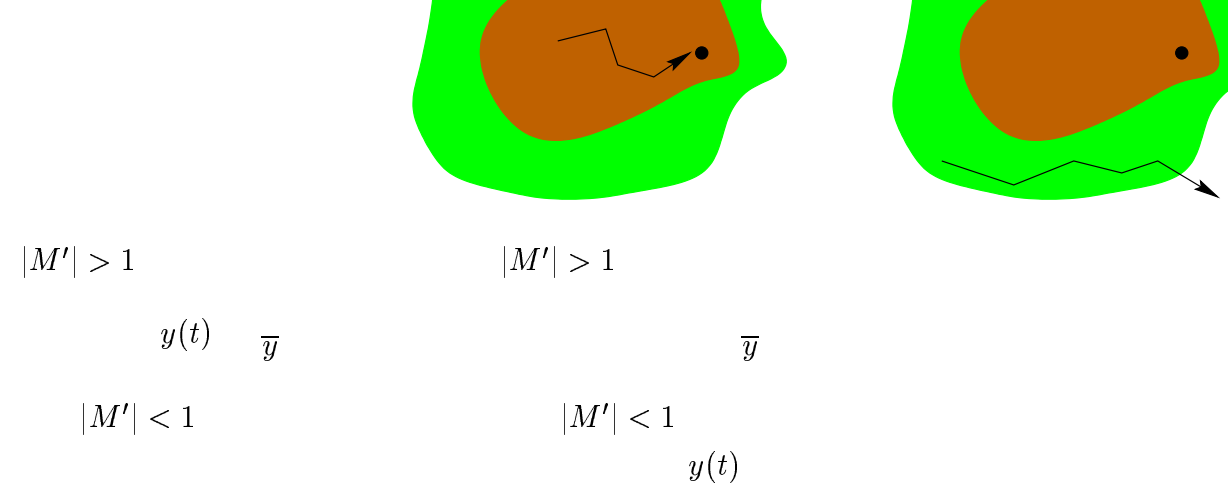

Figure 1: Robust latching. For simplicity a fixed-point attractor $\overline{y}$ is shown. The shadow region is the basin of attraction. The dark shadow region is the subset where $|M'| < 1$ and robust latch occurs. See text for details.

with respect to $y(s)$ with $s < t$ is simply the product of the map derivatives between s and t:

$$\frac{\partial y(t)}{\partial y(s)} = \frac{\partial y(t)}{\partial y(t-1)} \frac{\partial y(t-1)}{\partial y(t-2)} \cdots \frac{\partial y(s+1)}{\partial y(s)}.$$

When the norm of each of the factors on the right hand side is less than 1, the left hand side converges exponentially fast to zero as $t - s$ increases. The effect of this decay of gradients can be made explicit as follows:

$$\frac{\partial E(t)}{\partial W} = \sum_{\tau \leq t} \frac{\partial E(t)}{\partial y(\tau)} \frac{\partial y(\tau)}{\partial W} = \sum_{\tau \leq t} \frac{\partial E(t)}{\partial y(t)} \frac{\partial y(t)}{\partial y(\tau)} \frac{\partial y(\tau)}{\partial W}.$$

Hence for a term of the sum with $\tau \ll t$, we have

$$\left| \frac{\partial E(t)}{\partial y(\tau)} \frac{\partial y(\tau)}{\partial W} \right| \to 0.$$

This term tends to become very small in comparison to terms for which τ is close to t. This means that even though there might exist a change in W that would allow $y(\tau)$ to jump to another (better) basin of attraction, the gradient of the cost with respect to W does not clearly reflect that possibility. The explanation is that the effect of a *small* change in W would be felt mostly on the near past (τ close to t).

4 Remedies

The above theoretical investigations indicate a basic limitation of gradient descent as a search procedure for finding optimal weights in a RNN. Several proposals have been made to cope with the problem of long-term dependencies, some attempting to solve the optimization problem using alternative search algorithms, other trying to devise alternative architectures. In the following we give a brief accounts of these proposals.

Time constants

To deal with long time lags, Mozer [17] uses time constants influencing changes of unit activations (deVries and Principe's related approach [7] may be viewed as a mixture of time-delay neural networks (TDNN) [14] and time constants). For long time lags, however, the time constants need external fine tuning [17]. Sun *et al.*'s alternative approach [25] updates the activation of a recurrent unit by adding the old activation and the (scaled) current net input. The net input, however, tends to perturb the stored information, which makes long-term storage impractical. Lin *et al.* [16] also propose variants of time-delay networks, called NARX networks (crossreference see also Chapter 11). Gradient flow in this architecture can be improved because embedded memories effectively introduce "shortcuts" in the error propagation path through time. The same idea can be applied to other architectures, by inserting multiple delays in the connections among hidden state units rather than output units [15]. However, these architectures cannot solve the general problem since they can only increase by a constant multiplicative factor the duration of the temporal dependencies that can be learned. Finally, El Hihi & Bengio [9] looked at hierarchically organized recurrent neural networks with different levels of time-constants or time-delays.

Ring's approach

Ring [20] also proposes a method for bridging long time lags. Whenever a unit in his network receives conflicting error signals, that is, certain error signals suggest to increase the unit's activity while others suggest otherwise, he adds a higher order unit influencing appropriate connections. Although his approach can sometimes be extremely fast, to bridge a time lag involving 100 steps may require the addition of 100 units. Also, Ring's net does not

generalize to unseen lag durations.

Searching without gradients

The difficulty of learning long-term dependencies is strictly related to the continuous optimization approach that guides the search for a weight solution. One possibility for avoiding the problem is to resort to other kinds of search in weight space, in which the operators for generating another candidate weight solution are not based on continuous gradients. Bengio *et al.* [6] investigate methods such as simulated annealing, multi-grid random search, and discrete error propagation. Angeline *et al.* [1] (see also crossreference Chapter 15) propose a genetic approach that also avoids gradient computation.

The simplest kind of search without gradient, however, simply randomly initializes all network weights until the resulting net happens to classify all training sequences correctly. In fact, as discussed in crossreference Chapter 9 of this book, simple weight guessing solves several popular benchmarks described in previous work faster than the recurrent net algorithms proposed therein (compare [13]). This does not mean that weight guessing is a good algorithm. It just means that the problems are very simple. More realistic tasks require either many free parameters (e.g., input weights) or high weight precision (e.g., for continuous-valued parameters), such that guessing becomes completely infeasible. Currently it is unclear to which extent more complex gradient-less methods can improve upon guessing in case of more realistic tasks.

Probabilistic target propagation

Bengio and Frasconi [4] propose a probabilistic approach for propagating targets. With n so-called "state networks", at a given time, their system can be in one of only n different discrete states. Parameters are adjusted using the expectation-maximization algorithm. But to solve problems that require a significant amount of memory to store contextual information, such systems would require an unacceptable number of states (i.e., state networks).

Adaptive sequence chunkers

Schmidhuber's hierarchical chunker systems [23, 24] can in principle bridge arbitrary time lags, but only if there is local predictability across the sub-

sequences causing the time lags (see also [17]). For instance, in his post-doctoral thesis [24], Schmidhuber uses hierarchical recurrent networks with self-organizing time scales to rapidly solve certain grammar learning tasks involving minimal time lags in excess of 1000 steps. The performance of chunker systems, however, deteriorates as the noise level increases and the input sequences become less compressible.

Long Short-Term Memory

There is a novel, efficient, gradient-based method called "Long Short-Term Memory" (LSTM) [12]. LSTM is designed to get rid of the vanishing error problem. Truncating the gradient where this does not do harm, LSTM can learn to bridge minimal time lags in excess of 1000 discrete time steps by enforcing *constant* error flow through "constant error carrousels" within special units. Multiplicative gate units learn to open and close access to the constant error flow. LSTM is local in space and time; its computational complexity per time step and weight is $O(1)$. So far, experiments with artificial data involved local, distributed, real-valued, and noisy pattern representations. In comparisons with RTRL, BPTT, Recurrent Cascade-Correlation, Elman networks, and Neural Sequence Chunking, LSTM led to many more successful runs, and learned much faster. LSTM also solved complex, artificial long time lag tasks that have never been solved by previous recurrent network algorithms. It will be interesting to examine to which extent LSTM is applicable to real world problems such as speech recognition.

5 Conclusions

In principle, RNNs are the most general and powerful current sequence learning method. For instance, unlike Hidden Markov Models (HMMs, the most successful technique in several sequence processing applications - see [3] for a review) they are not limited to discrete internal states but allow for continuous, distributed sequence representations. Hence they can solve tasks no other current method can solve (e.g., [10]). The problem of vanishing gradients, however, makes conventional RNNs hard to train. We suspect this is why feedforward neural networks outnumber RNNs in terms of successful real-world applications. Some of the remedies outlined in this chapter may lead to more effective learning systems. However, long lime lag research still

seems to be in an early stage — no commercial applications of any of these methods have been reported so far.

Long time lags pose problems to any soft computing method, not just RNNs. For instance, when dealing with long sequences (e.g., speech or biological data), HMMs mostly rely on a localized representation of time by means of highly constrained non ergodic transition diagrams (different states are designed for different portions of a sequence). Belief propagation over long time lags does not effectively occur, a phenomenon called diffusion of credit [5], which closely resembles the vanishing gradients problem in RNNs.

References

[1] P. J. Angeline, G. M. Saunders, and J. P. Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE Transactions on Neural Networks*, 5(1):54–65, 1994.

[2] P. Baldi and F. Pineda. Contrastive learning and neural oscillator. *Neural Computation*, 3:526–545, 1991.

[3] Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2:129–162, 1999.

[4] Y. Bengio and P. Frasconi. Credit assignment through time: Alternatives to backpropagation. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 75–82. San Mateo, CA: Morgan Kaufmann, 1994.

[5] Y. Bengio and P. Frasconi. Diffusion of context and credit information in Markovian models. *Journal of Artificial Intelligence Research*, 3:249–270, 1995.

[6] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[7] B. de Vries and J. C. Principe. A theory for neural networks with time delays. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 162–168. San Mateo, CA: Morgan Kaufmann, 1991.

[8] K. Doya. Bifurcations in the learning of recurrent neural networks. In *Proceedings of 1992 IEEE International Symposium on Circuits and Systems*, pages 2777–2780, 1992.

[9] S. El Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 493–499. MIT Press, Cambridge MA, 1996.

[10] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. In *Proc. ICANN'99, Int. Conf. on Artificial Neural Networks*, pages 850–855, Edinburgh, Scotland, 1999. IEE, London.

[11] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. See www7.informatik.tu-muenchen.de/~hochreit.

[12] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[13] S. Hochreiter and J. Schmidhuber. LSTM can solve hard long time lag problems. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 473–479. MIT Press, 1997.

[14] K. Lang, A. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[15] T. Lin, B. G. Horne, and C. L. Giles. How embedded memory in recurrent neural network architectures helps learning long-term temporal dependencies. *Neural Networks*, 11(5):861–868, 1998.

[16] T. Lin, B. G. Horne, P. Tiño, and C. L. Giles. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, 7(6):1329–1338, November 1996.

[17] M. C. Mozer. Induction of multiscale temporal structure. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 4*, pages 275–282. San Mateo, CA: Morgan Kaufmann, 1992.

[18] J.M. Ortega and W.C. Rheinboldt. *Iterative Solution of Non-linear Equations in Several Variables and Systems.* Academic Press, New York, 1970.

[19] F. J. Pineda. Dynamics and architecture for neural computation. *Journal of Complexity*, 4:216–245, 1988.

[20] M. B. Ring. Learning sequential tasks by incrementally adding higher orders. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 115–122. Morgan Kaufmann, 1993.

[21] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1, pages 318–362. MIT Press, 1986.

[23] J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[24] J. Schmidhuber. Netzwerkarchitekturen, Zielfunktionen und Kettenregel. Habilitationsschrift, Institut für Informatik, Technische Universität München, 1993.

[25] G. Sun, H. Chen, and Y. Lee. Time warping invariant neural networks. In J. D. Cowan S. J. Hanson and C. L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 180–187. San Mateo, CA: Morgan Kaufmann, 1993.

[26] P. J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1, 1988.

[27] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Back-propagation: Theory, Architectures and Applications*. Hillsdale, NJ: Erlbaum, 1992.