Applying LSTM to Time Series Predictable through Time-Window Approaches

Felix A. Gers felix@idsia.ch
Douglas Eck doug@idsia.ch
Jürgen Schmidhuber juergen@idsia.ch

IDSIA, Galleria 2, 6928 Manno, Switzerland, www.idsia.ch

Abstract. Long Short-Term Memory (LSTM) is able to solve many time series tasks unsolvable by feed-forward networks using fixed size time windows. Here we find that LSTM's superiority does *not* carry over to certain simpler time series prediction tasks solvable by time window approaches: the Mackey-Glass series and the Santa Fe FIR laser emission series (Set A). This suggests to use LSTM only when simpler traditional approaches fail.

1 Overview

Long Short-Term Memory (LSTM) is a recurrent neural network (RNN) architecture that had been shown to outperform traditional RNNs on numerous temporal processing tasks [1–4].

Time series benchmark problems found in the literature, however, often are conceptually simpler than many tasks already solved by LSTM. They often do not require RNNs at all, because all relevant information about the next event is conveyed by a few recent events contained within a small time window. Here we apply LSTM to such relatively simple tasks, to establish a limit to the capabilities of the LSTM-algorithm in its current form. We focus on two intensively studied tasks, namely, prediction of the Mackey-Glass series [5] and chaotic laser data (Set A) from a contest at the Santa Fe Institute (1992) [6].

LSTM is run as a "pure" autoregressive (AR) model that can only access input from the current time-step, reading one input at a time, while its competitors — e.g., multi-layer perceptrons (MLPs) trained by back-propagation (BP) — simultaneously see several successive inputs in a suitably chosen time window. Note that Time-Delay Neural Networks (TDNNs) [7] are not purely AR, because they allow for direct access to past events. Neither are NARX networks [8] which allow for several distinct input time windows (possibly of size one) with different temporal offsets.

We also evaluate stepwise versus iterated training (IT) [9, 10].

2 Experimental Setup

The task is to use currently available points in the time series to predict the future point, $t + T$. The target for the network t_k is the difference between

G. Dorffner, H. Bischof, and K. Hornik (Eds.): ICANN 2001, LNCS 2130, pp. 669–676, 2001.

the values $x(t+p)$ of the time series p steps ahead and the current value $x(t)$ multiplied by a scaling factor f_s: $t_k(t) = f_s \cdot (x(t+p) - x(t)) = f_s \cdot \Delta x(t)$. The value for f_s scales $\Delta x(t)$ between –1 and 1 for the training set; the same value for f_s is used during testing. The predicted value is the network output divided by f_s plus $x(t)$. During iterated prediction with $T = n * p$ the output is clamped to the input (self-iteration) and the predicted values are fed back n times. For direct prediction $p = T$ and $n = 1$; for single-step prediction $p = 1$ and $n = T$.

The error measure is the normalized root mean squared error: NRMSE = $\langle (y_k - t_k)^2 \rangle^{\frac{1}{2}} / \langle (t_k - \langle t_k \rangle)^2 \rangle^{\frac{1}{2}}$, where y_k is the network output and t_k the target. The reported performance is the best result of 10 independent trials.

LSTM Network Topology. The input units are fully connected to a hidden layer consisting of memory blocks with 1 cell each. The cell outputs are fully connected to the cell inputs, to all gates, and to the output units. All gates, the cell itself and the output unit are biased. Bias weights to input and output gates are initialized block-wise: -0.5 for the first block, -1.0 for the second, -1.5 for the third, and so forth. Forget gates are initialized with symmetric positive values: $+0.5$ for the first block, $+1$ for the second block, etc. These are standard values that we use for all experiments. All other weights are initialized randomly in the range $[-0.1, 0.1]$. The cell's input squashing function g is a sigmoid function with the range $[-1, 1]$. The squashing function of the output unit is the identity function.

To have statistically independent weight updates, we execute weight changes every 50+rand(50) steps (where rand(max) stands for a random positive integer smaller than max which changes after every update). We use a constant learning rate $\alpha = 10^{-4}$.

MLP. The MLPs we use for comparison have one hidden layer and are trained with BP. As with LSTM, the one output unit is linear and Δx is the target. The input differs for each task but in general uses a time window with a time-space embedding. All units are biased and the learning rate is $\alpha = 10^{-3}$.

3 Mackey-Glass Chaotic Time Series

The Mackey-Glass (MG) chaotic time series can be generated from the MG delay-differential equation [5]: $\dot{x}(t) = \frac{\alpha x(t-\tau)}{1+x^c(t-\tau)} - \beta x(t)$. We generate benchmark sets using a four-point Runge-Kutta method with step size 0.1 and initial condition $x(t) = 0.8$ for $t < 0$. Equation parameters were set at $a = 0.2$, $b = 0.1$, $c = 10$ and $\tau = 17$. The equation is integrated up to $t = 5500$, with the points from $t = 200$ to $t = 3200$ used for training and the points from $t = 5000$ to $t = 5500$ used for testing.

MG Previous Work. In the following sections we list attempts to predict the MG time series. To allow comparison among approaches, we did not consider works where noise was added to the task or where training conditions were very different from ours. When not specifically mentioned, an input time window with time delays t, $t-6$, $t-12$ and $t-18$ or larger was used. Summary of previous approaches:

BPNN [11]: A BP continuous-time feed forward NNs with two hidden layers and with fixed time delays. **ATNN** [11]: A BP continuous-time feed forward NNs with two hidden layers and with adaptable time delays. **DCS-LMM** [12]: Dynamic Cell Structures combined with Local Linear Models. **EBPTTRNN** [13]: RNNs with 10 adaptive delayed connections trained with BPTT combined with a constructive algorithm. **BGALR** [14]: A genetic algorithm with adaptable input time window size (Breeder Genetic Algorithm with Line Recombination). **EPNet** [15]: Evolved neural nets (Evolvable Programming Net). **SOM** [16]: A Self-organizing map. **Neural Gas** [17] : The Neural Gas algorithm for a Vector Quantization approach. **AMB** [18]: An improved memory-based regression (MB) method [19] that uses an adaptive approach to automatically select the number of regressors (AMB). The results from these approaches are found in Table 1.

Table 1. Results for the MG task, showing (from left to right) the number of units, the number of parameters (weights for NNs), the number of training sequence presentations, and the NRMSE for prediction offsets $T \in \{1, 6, 84\}$.

Reference	Units	Para.	Seq.	NMSE		
				$T=1$	$T=6$	$T=84$
Linear Predictor	-	-	-	0.0327	0.7173	1.5035
6th-order Polynom.	-	-	-	-	0.04	0.85
BPNN	-	-	-	-	0.02	0.06
FTNN	20	120	$7 \cdot 10^7$	-	0.012	-
ATNN	20	120	$7 \cdot 10^7$	-	0.005	-
Cascade-Correlation	20	≈ 250	-		0.04	0.17
DCS-LLM	200	200^2	$\approx 1 \cdot 10^5$	-	0.0055	0.03
EBPTTRNN	6	65	-	-	0.0115	-
BGALR	16	≈ 150	-	-	0.2373	0.267
EPNet	≈ 10	≈ 100	$\approx 1 \cdot 10^4$	-	0.02	0.06
SOM	-	10x10	$\approx 1.5 \cdot 10^4$	-	0.013	0.06
	-	35x35	$\approx 1.5 \cdot 10^4$	-	0.0048	0.022
Neural Gas	400	3600	$2 \cdot 10^4$	-	-	0.05
AMB	-		-	-	-	0.054
MLP, $p=T$	16	97	$1 \cdot 10^4$	0.0113	0.0502	0.4612
MLP, $p=1$	16	97	$1 \cdot 10^4$	$p=T=1$	0.0252	0.4734
MLP, $p=1$, IT	16	97	$1 \cdot 10^4$	0.0094	0.0205	0.3929
MLP, $p=6$	16	97	$1 \cdot 10^4$	-	$p=T=6$	0.1466
MLP, $p=6$, IT	16	97	$1 \cdot 10^4$	-	0.0945	0.2820
LSTM, $p=T$	4	113	$5 \cdot 10^4$	0.0214	0.1184	0.4700
LSTM, $p=1$	4	113	$5 \cdot 10^4$	$p=T=1$	0.1981	0.5927
LSTM, $p=1$, IT	4	113	$1 \cdot 10^4$	s. text	0.1970	0.8157
LSTM, $p=6$	4	113	$5 \cdot 10^4$	-	$p=T=6$	0.2910
LSTM, $p=6$, IT	4	113	$1 \cdot 10^4$	-	0.1903	0.3595

MG Results. The LSTM results are listed at the bottom of Table 1. After six single-steps of iterated training ($p = 1$, $T = 6$, $n = 6$) the LSTM NRMSE for single step prediction ($p = T = 1$, $n = 1$) is 0.0452. After 84 single-steps of iterated training ($p = 1$, $T = 84$, $n = 84$) the LSTM NRMSE single step prediction ($p = T = 1$, $n = 1$) is 0.0809. Increasing the number of memory blocks did not significantly improve the results.

The results for AR-LSTM approach are clearly worse than the results for time window approaches, for example with MLPs. Why did LSTM perform worse than the MLP? The AR-LSTM network does not have access to the past as part of its input and therefore has to learn to extract and represent a Markov state. In tasks we considered so far this required remembering one or two events from the past, then using this information before over-writing the same memory cells. The MG equation, contains the input from $t-17$, hence its implementation requires the storage of all inputs from $t-17$ to t (time window approaches consider selected inputs back to at least $t-18$). Assuming that any dynamic model needs the event from time $t-\tau$ with $\tau \approx 17$, we note that the AR-RNN has to store all inputs from $t-\tau$ to t and to overwrite them at the adequate time. This requires the implementation of a circular buffer, a structure quite difficult for an RNN to simulate.

MG Analysis It is interesting that for MLPs ($T = 6$) it was more effective to transform the task into a one-step-ahead prediction task and iterate than it was to predict directly (compare the results for $p = 1$ and $p = T$). It is in general easier to predict fewer steps ahead, the disadvantage being that during iteration input values have to be replaced by predictions. For $T = 6$ with $p = 1$ this affects only the latest value. This advantage is lost for $T = 84$ and the results with $p = 1$ are worse than with $p = 6$, where fewer iterations are necessary. For MLPs, iterated training did not in general produce better results: it improved performance when the step-size p was 1, and worsened performance for $p = 6$. For LSTM iterated training decreased the performance. But surprisingly, the relative performance decrease for one-step prediction was much larger than for iterated prediction. This indicates that the iteration capabilities were improved (taking in consideration the over-proportionally worsened one-step prediction performance).

The single-step predictions for LSTM are not accurate enough to follow the series for as much as 84 steps. Instead the LSTM network starts oscillating, having adapted to the strongest eigen-frequency in the task. During self-iterations, the memory cells tune into this eigen-oscillation, with time constants determined by the interaction of cell state and forget gate.

4 Laser Data

This data is set A from the Santa Fe time series prediction competition [6][1]. It consists of one-dimensional data recorded from a Far-Infrared (FIR) laser in a

[1] The data is available from http://www.stern.nyu.edu/~aweigend/Time-Series/SantaFe.html.

chaotic state [20]. The training set consists of 1,000 points from the laser, with the task being to predict the next 100 points. We run tests for stepwise prediction and fully iterated prediction, where the output is clamped to the input for 100 steps.

For the experiments with MLPs the setup was as described for the MG data but with an input embedding of the last 9 time steps as in Koskela, Varsta and Heikkonen [21].

FIR-laser Previous Work Results are listed in Table 2. Linear prediction is no better than predicting the data-mean. Wan [22] achieved the best results submitted to the original Santa Fe contest. He used a Finite Input Response Network (FIRN) (25 inputs and 12 hidden units), a method similar to a TDNN. Wan improved performance by replacing the last 25 predicted points by smoothed values (sFIRN). Koskela, Varsta and Heikkonen [21] compared recurrent SOMs (RSOMs) and MLPs (trained with the Levenberg-Marquardt algorithm) with an input embedding of dimension 9 (an input window with the last 9 values). Bakker et. al. [10] used a mixture of predictions and true values as input (Error Propagation, EP). Then Principal Component Analysis (PCA) was applied to reduce the dimensionality of the time embedding. Kohlmorgen and Müller [23] pointed out that the prediction problem could be solved by pattern matching, if it can be guaranteed that the best match from the past is always the right one. To resolve ambiguities they propose to up-sample the data using linear extrapolation (as done by Sauer [24]). The best result to date, according to our knowledge, was achieved by Weigend and Nix [25]. They used a non-linear regression approach in a maximum likelihood framework, realized with feed-forward NN (25 inputs and 12 hidden units) using an additional output to estimate the prediction error. McNames [26] proposed a statistical method that used cross-validation error to estimate the model parameters for local models, but the testing conditions were too different to include the results in the comparison. Bontempi et. al. [27] used a similar approach called " Predicted Sum of Squares (PRESS)" (here, the dimension of the time embedding was 16).

FIR-laser Results The results for MLP and LSTM are listed in Table 2. The results for these methods are not as good as the other results listed in Table 2. This is true in part because we did not replace predicted values by hand with a mean value where we suspected the system to be lead astray.

Iterated training yielded improved results for iterated prediction, even when single-step prediction became worse, as in the case of MLP.

FIR-laser Analysis The LSTM network could not predict the collapse of emission in the test set (Figure 1). Instead, the network tracks the oscillation in the original series for only about 40 steps before desynchronizing. This indicates performance similar to that in the MG task: the LSTM network was able to track the strongest eigen-frequency in the task but was unable to account for high-frequency variance. Though the MLP performed better, it generated inaccurate amplitudes and also desynchronized after about 40 steps. The MLP did however manage to predict the collapse of emission.

Table 2. Results for the FIR-laser task, showing (from left to right): The number of units, the number of parameters (weights for NNs), the number of training sequence presentations, and the NRMSE.

Reference	Units	Para.	Seq.	NMSE	
				stepwise	iterated
Linear Predictor	-	-	-	1.25056	-
FIRN [22]	26	≈ 170	-	0.0230	0.0551
sFIRN [22]	26	≈ 170	-	-	0.0273
MLP [21]	70	≈ 30	-	0.01777	-
RSOM [21]	13	-	-	0.0833	-
EP-MLP Bakker et. al. (2000) [10]	73	> 1300	-	-	0.2159
Sauer (1994) [24]	-	32	-	-	0.077
Weigend and Nix (1994) [25]	27	≈ 180	-	0.0198	0.016
Bontempi (1999) [27]	-	-	-	-	0.029
MLP	32	353	$1 \cdot 10^4$	0.0996017	0.856932
MLP IT	32	353	$1 \cdot 10^4$	0.158298	0.621936
LSTM	4	113	$1 \cdot 10^5$	0.395959	1.02102
LSTM IT	4	113	$1 \cdot 10^5$	0.36422	0.96834

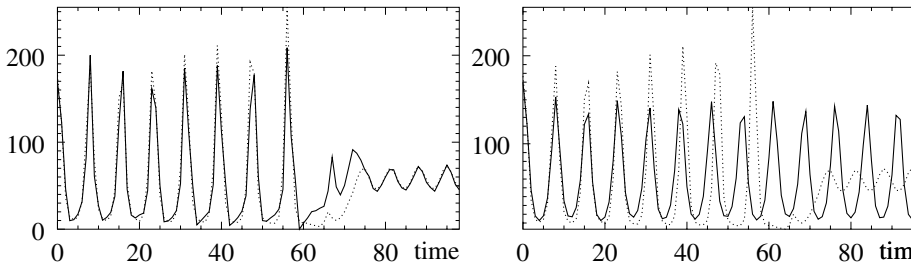

Fig. 1. Test run with LSTM network solution after iterated training for the FIR-laser task. Left: Single-Step prediction. Right: Iteration of 100 steps.

5 Conclusion

A time window based MLP outperformed the LSTM pure-AR approach on certain time series prediction benchmarks solvable by looking at a few recent inputs only. Thus LSTM's special strength, namely, to learn to remember single events for very long, unknown time periods, was not necessary here.

LSTM learned to tune into the fundamental oscillation of each series but was unable to accurately follow the signal. The MLP, on the other hand, was able to capture some aspects of the chaotic behavior. For example the system could predict the collapse of emission in the FIR-laser task.

Iterated training has advantages over single-step training for iterated testing only for MLPs and when the prediction step-size is one. The advantage is evident when the number of necessary iterations is large.

Our results suggest to use LSTM only on tasks where traditional time window based approaches must fail. One reasonable *hybrid* approach to prediction of unknown time series may be this: start by training a time window-based MLP, then freeze its weights and use LSTM only to reduce the residual error if there is any, employing LSTM's ability to cope with long time lags between significant events.

LSTM's ability to track slow oscillations in the chaotic signal may be applicable to cognitive domains such as rhythm detection in speech and music.

Acknowledgment. This work was supported by SNF grant 2100-49'144.96.

References

1. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
2. 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.
3. F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proc. IJCNN'2000, Int. Joint Conf. on Neural Networks*, (Como, Italy), 2000.
4. F. A. Gers and J. Schmidhuber, "LSTM recurrent networks learn simple context free and context sensitive languages," *IEEE Transactions on Neural Networks*, 2001. accepted.
5. M. Mackey and L. Glass, "Oscillation and chaos in a physiological control system," *Science*, vol. 197, no. 287, 1977.
6. A. Weigend and N. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past.* Addison-Wesley, 1993".
7. P. Haffner and A. Waibel, "Multi-state time delay networks for continuous speech recognition," in *Advances in Neural Information Processing Systems* (J. E. Moody, S. J. Hanson, and R. P. Lippmann, eds.), vol. 4, pp. 135–142, Morgan Kaufmann Publishers, Inc., 1992.
8. 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*, vol. 7, pp. 1329–1338, Nov. 1996.
9. J. C. Principe and J.-M. Kuo, "Dynamic modelling of chaotic time series with neural networks," in *Advances in Neural Information Processing Systems* (G. Tesauro, D. Touretzky, and T. Leen, eds.), vol. 7, pp. 311–318, The MIT Press, 1995.
10. R. Bakker, J. C. Schouten, C. L. Giles, F. Takens, and C. M. van den Bleek, "Learning chaotic attractors by neural networks," *Neural Computation*, vol. 12, no. 10, 2000.
11. S. P. Day and M. R. Davenport, "Continuous-time temporal back-progagation with adaptive time delays," *IEEE Transactions on Neural Networks*, vol. 4, pp. 348–354, 1993.
12. L. Chudy and I. Farkas, "Prediction of chaotic time-series using dynamic cell structuresand local linear models," *Neural Network World*, vol. 8, no. 5, pp. 481–489, 1998.
13. R. Bone, M. Crucianu, G. Verley, and J.-P. Asselin de Beauville, "A bounded exploration approach to constructive algorithms for recurrent neural networks," in *Proceedings of IJCNN 2000*, (Como, Italy), 2000.

14. I. de Falco, A. Iazzetta, P. Natale, and E. Tarantino, "Evolutionary neural networks for nonlinear dynamics modeling," in *Parallel Problem Solving from Nature 98*, vol. 1498 of *Lectures Notes in Computer Science*, pp. 593–602, Springer, 1998".
15. X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 8, pp. 694–713, May 1997.
16. J. Vesanto, "Using the SOM and local models in time-series prediction," in *Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4–6*, pp. 209–214, Espoo, Finland: Helsinki University of Technology, Neural Networks Research Centre, 1997.
17. T. M. Martinez, S. G. Berkovich, and K. J. Schulten, "Neural-gas network for vector quantization and its application to time-series prediction," *IEEE Transactions on Neural Networks*, vol. 4, pp. 558–569, July 1993.
18. H. Bersini, M. Birattari, and G. Bontempi, "Adaptive memory-based regression methods," in *In Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, pp. 2102–2106, 1998.
19. J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, pp. 213–225, 1991.
20. U. Huebner, N. B. Abraham, and C. O. Weiss, "Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared nh3 laser," *Phys. Rev. A*, vol. 40, p. 6354, 1989.
21. T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, "Recurrent SOM with local linear models in time series prediction," in *6th European Symposium on Artificial Neural Networks. ESANN'98. Proceedings. D-Facto, Brussels, Belgium*, pp. 167–72, 1998.
22. E. A. Wan, "Time series prediction by using a connectionist network with internal time delays," in *Time Series Prediction: Forecasting the Future and Understanding the Past* (W. A. S. and G. N. A., eds.), pp. 195–217, Addison-Wesley, 1994.
23. J. Kohlmorgen and K.-R. Müller, "Data set a is a pattern matching problem," *Neural Processing Letters*, vol. 7, no. 1, pp. 43–47, 1998.
24. T. Sauer, "Time series prediction using delay coordinate embedding," in *Time Series Prediction: Forecasting the Future and Understanding the Past* (A. S. Weigend and N. A. Gershenfeld, eds.), Addison-Wesley, 1994.
25. A. S. Weigend and D. A. Nix, "Predictions with confidence intervals (local error bars)," in *Proceedings of the International Conference on Neural Information Processing (ICONIP'94)*, (Seoul, Korea), pp. 847–852, 1994.
26. J. McNames, "Local modeling optimization for time series prediction," in *In Proceedings of the 8th European Symposium on Artificial Neural Networks*, pp. 305–310, 2000.
27. B. H. Bontempi G., Birattari M., "Local learning for iterated time-series prediction," in *Machine Learning: Proceedings of the Sixteenth International Conference* (B. I. and D. S., eds.), (San Francisco, USA), pp. 32–38, Morgan Kaufmann, 1999.