A Comparison Between Spiking and Differentiable Recurrent Neural Networks on Spoken Digit Recognition

Alex Graves, Nicole Beringer, Jürgen Schmidhuber
IDSIA *Istituto Dalle Molle di Studi sull'Intelligenza Artificiale*
Galleria 2, 6928 Manno, Switzerland

ABSTRACT
In this paper we demonstrate that Long Short-Term Memory (LSTM) is a differentiable recurrent neural net (RNN) capable of robustly categorizing timewarped speech data. We measure its performance on a spoken digit identification task, where the data was spike-encoded in such a way that classifying the utterances became a difficult challenge in non-linear timewarping. We find that LSTM gives greatly superior results to an SNN found in the literature, and conclude that the architecture has a place in domains that require the learning of large timewarped datasets, such as automatic speech recognition.

KEY WORDS
Speech Recognition, LSTM, RNN, SNN, Timewarping

1 Introduction

Non-linear timewarping is a major difficulty in speech recognition. To identify words under realistic conditions, a recogniser must be able to handle large variations in speaker rate, both over whole words and over sub-word units, such as syllables and phones. So far, this goal has been most successfully achieved by Hidden Markov Models [1], especially when augmented by neural nets [2, 3]. However, we feel that a model able to build a low-complexity representation of sequential input would provide a powerful alternative.

In the following, we will compare two neural net candidates for such a model: spiking recurrent neural networks (SNNs) and long short term memory (LSTM)[4]. We show that LSTM can identify spoken digits from the *mus silicium* competition with far greater accuracy than Liquid State Machines [5], an SNN that has been recently applied to speech recognition. Furthermore LSTM performs quickly and reliably, and requires fewer adjustable parameters than do SNNs. We conclude that LSTM combines desirable dynamical characteristics of SNNs with the RNN advantage of powerful gradient descent learning, and that it has the potential to play an important role in automatic speech recognition systems.

The structure of the paper is as follows: In Section 2.1, we examine the problems historically experienced by neural nets in automatic speech recognition (ASR) and other sequence learning tasks. In Section 2.2, we review SNNs and their application to ASR. In Section 2.3 we introduce and describe LSTM, and in Section 2.4 we discuss its suitability for ASR. In Section 3, we provide experimental comparisons between LSTM and SNN's on a digit recognition task, while concluding remarks are presented in Section 4.

2 Background

2.1 Differentiable Neural Networks

For differentiable neural nets, dealing with time dependent inputs means one of two things: either collecting the inputs into time windows and treating the task as spatial; or using recurrent connections and an internal state to model time directly. We now summarize the application of these methods to speech recognition.

Time windowed networks have been successful in recognising isolated syllables or small words — see [6] or Section 3.0.2. However, on more challenging tasks, like continuous speech, they run into two fundamental problems. Firstly, the window size is fixed by the topology of the network (and usually limited by speed and memory considerations). This means that either the net has a huge number of inputs (and thus a huge number of parameters and a very long training time), or else that important long time dependencies, such as the position of the window in a word, are simply ignored. Secondly, such nets are inflexible with regard to temporal displacements or changes in the rate of input (non-linear timewarping). This leaves them easily confused by ubiquitous features of speech, such as delays or variations in talking speed.

With RNNs, on the other hand, it is not necessary to transform temporal patterns into spatial ones. Instead, a time series can be presented one frame at a time, with the flow of activations around the connections creating a memory of previous inputs. Recurrent training algorithms such as RTRL [7] and BPTT [8] can perform weight updates based on the entire history of the network's states. Therefore it seems feasible that they could process any length of time series. In practice though, these algorithms share a common weakness: their backpropagated errors either explode or decay in time, preventing them from learning dependencies of more than a few timesteps in length [9].

2.2 Spiking Neural Networks

Biological neurons communicate using trains of pulse-like voltage spikes, with active neurons producing more spikes on average than inactive ones. This behaviour is captured in traditional differentiable networks through the use of an "activation" term that encodes mean firing rate.

Some suggest that mean firing rate is not enough. Rather, it is believed that a computational model that simulates the spike-like nature of biological neural systems will benefit from a richer set of dynamical behaviours than traditional neural networks. In particular, firing synchrony is only available to models that simulate (rather than average) spike onsets. For an overview of SNNs see [10].

One major advantage of these models is an ability to find temporal structure in a signal using synchronization. Synchrony-based binding mechanisms as are found in SNNs tend to be robust to significant time-warping and temporal noise. (See [11] for one author's application of SNNs to the discovery of metrical structure in music, a problem that shares some similarities with speech recognition.)

SNNs have shown great promise as pattern recognisers [12]. Unfortunately, SNNs do not benefit from powerful learning mechanisms, making it difficult to apply them to complex sequence learning tasks such as continuous word or phone prediction. (Some progress has been made in this area. See for example [13].) What is desired is a model that benefits from the powerful gradient descent learning found in traditional neural networks but that captures some of the rich dynamical properties of SNNs. Such a model would be able to learn to categorize timewarped datasets as well as SNNs but would also have potential for harder sequence learning tasks.

2.3 LSTM

It is out of the scope of this paper to describe LSTM in depth, and the interested reader should consult [14, 4] for details. However, we can provide a brief outline of the network's structure, as illustrated in figure 1.

LSTM was designed to obtain constant error flow through time and to protect this error flow from undesired perturbations. It uses linear memory cells called *Constant Error Carousels* (CECs) to overcome error decay problems plaguing previous RNNs. Each CEC has a fixed self-connection and is surrounded by a cloud of nonlinear units responsible for controlling the flow of information in and out of the CEC. Typically, a multiplicative *input gate unit* learns to protect the flow from perturbation by irrelevant inputs. Likewise, a multiplicative *output gate unit* learns to protect other units from perturbation by currently irrelevant memory contents. A *forget gate* learns to reset a memory cell when its content is obsolete. Learning is carried out by a gradient descent method based on a slightly modified, truncated form of BPTT and a customized version of RTRL. (Unlike RTRL however, this requires only O(1) computations per time step and weight.) Output units use backpropagation; output gates use the truncated version of BPTT; while weights to cells, input gates and forget gates use truncated RTRL. LSTM performance is improved in online learning situations by using a Kalman filter to control weight updates [15].

2.4 Why Use LSTM for Speech Recognition?

As mentioned in Section 2.1, it is essential for an RNN used in speech recognition to be able to bridge long time lags, and adapt to time warped data. These are two areas in which LSTM has outperformed other RNN's. That LSTM can deal with long time lags has been demonstrated in [4, 16], while its utility with time-warping is clear from its success in learning context free languages [16], and in generating music [17]. In both cases, its advantage comes from the fact that because its central timing mechanism is not (as for most RNNs) a decaying flow of recurrent activation. Instead, its memory cells act as a set of independent counters. These cells (along with the gates used to open, close and reset them) allow LSTM to extract and store information at a very wide range of timescales.

However, HMMs, rather than RNNs, are the standard tool for speech recognition, so the question becomes: why use LSTM instead of HMMs? One answer is that LSTM is more biologically plausible. Another, perhaps more important, is that statistical models like HMMs tend to be less robust than RNNs. For example, HMMs are very sensitive to channel errors, and coding algorithms are needed to clean up the data before they see it. Also, the parameters and language models used in HMMs are tuned towards particular datasets, and the choice of acoustic model they use is dependent on the size of the corpus. LSTM on the other hand, is a general purpose algorithm for extracting statistical regularities from noisy time series. Unlike HMMs, it doesn't rely on the manual introduction of linguistic and acoustic models, but can learn its own internal models directly from the data. And although (like all neural nets) it does have free parameters, such as learning rate and layer size, we demonstrate below that these do not need to be adjusted for particular corpora.

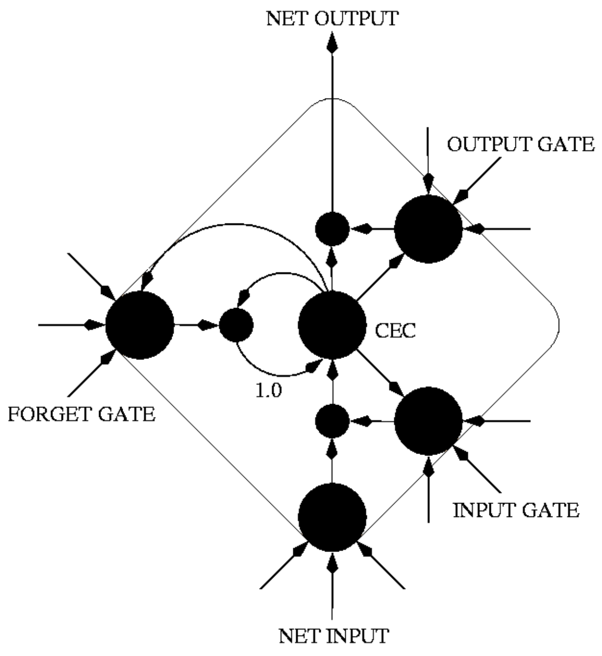

Figure 1. LSTM memory block with one cell. The internal state of the CEC (Constant Error Carousel) is maintained with a recurrent connection of weight 1.0. The three gates collect activations from both inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates effectively scale the input and output of the CEC while the forget gate scales the internal state — for example by resetting it to 0 (making it forget).

3 Experiments

The data used in this paper were downloaded from the website [18] of the *mus silicium* neural computation competition. As described in *Nature* (see [19]), they were a set of pre-processed speech files, each obtained from a single recording of a spoken digit. There were 10 speakers, each uttering the words "zero" to "nine" five times, making a total of 500 files. The data consisted of 40 spike trains with either one or zero spikes per train (representing onsets, peaks or offsets at 40 different frequencies, ranging from 100 Hz to 5 kHz).

Following Maass [5], we subdivided the 500 data files into a training set of size 300 and a test set of size 200. We trained the network and then used it to identify which of the ten digits, 0 to 9, each test pattern represented. We measured the identification score for the digit "one," according to the following formula: $\frac{N_{fo}}{N_{co}} + \frac{N_{fn}}{N_{cn}}$, where N_{fo} (N_{co}) = number of falsely (correctly) identified"one" files, N_{fn} (N_{fo}) = number of falsely (correctly) identified "not one" files. We also measured the fraction of correctly identified digits across the whole test set. The data was sampled at 100 timesteps per second. By way of comparison, we repeated the experiment with a mel-frequency cepstrum coefficient (MFCC) front end instead of the spike trains (using the HTK toolkit [20]).

An important note is that most of the difficulty of this data was due to the discrete, spike encoded, pre-processing, and not to the inherent difficulty of identifying spoken digits (LSTM can do this almost perfectly with a more conventional front end — see table 1.) Having no input other than the spikes, the net was forced to bridge the time lags between them, and to cope with non-linear distortions in these time lags, *without* the mass of spectral information that would normally be available.

3.0.1 Network Topology and Parameters

For the LSTM experiments, we used a neural net with a mix of extended LSTM (see section 2.3 for details) and sigmoidal layers. A cross-entropy objective function was used, and the output layer had a gain of 3. The network had two hidden layers. The LSTM layer contained 30 memory blocks, each with two cells: its squashing function was logistic with range $[-2, 2]$, and the activation functions of the gates were logistic in range $[0, 1]$. The second hidden layer consisted of 10 sigmoidal units. During training, errors were fed back on every timestep, encouraging the net to make correct decisions as early as possible (a useful property for real time applications). Gaussian noise (about the spike times) was injected into the training data to prevent overfitting.

Online learning was used with a learning rate of

Table 1. Digit recognition scores

Network	Avg	Best	% Correct
LSTM (MFCC input)	0.015	0	99.4
LSTM (spike encoded input)	0.041	0	88.4
Liquid State Machine	0.140	0.013	-

5×10^{-7} and momentum of 0.9. The network activations were reset to zero after each pattern presentation. Errors were fed back on every timestep and the input was sampled at a rate of 100 timesteps per second.

3.0.2 Results

In Table 1, averages for LSTM and feedforward were taken over 10 runs. Liquid State Machine results were from [5]. MFCC input means input features were mel-frequency cepstrum coefficients. "Avg" and "Best" were the average and best scores, as calculated with the formula from Section 3 (note that a lower score is better). "% Correct" is the average percentage of correctly identified digits.

3.0.3 Analysis

LSTMs high performance on the spike encoded version of this task illustrates its robustness to time warped data, and its superiority as a generic sequence processing tool to Liquid State Machines. Moreover, its near perfect performance on the MFCC encoded data shows that it is capable of rivalling state-of-the-art speech recognition systems. Also, it is worth noting that LSTM was able to make correct predictions early, typically after less than thirty percent of the input pattern. This would be a useful ability for an online recognition task, where immediate identifications are required.

4 Conclusions

We applied LSTM to the speech recognition task of identifying isolated spoken digits. We found that it greatly outperformed a well-known SNN found in the literature, and was able to identify any digit with more than 88% probability (99.4% probability when combined with more conventional preprocessing). Additionally, it was usually able to make identifications after having seen only a small part of each pattern.

We believe that the success of LSTM in these experiments was due to two factors: firstly it has, like SNNs, the ability to bridge long time delays and to accurately categorise timewarped data; secondly, like RNNs, it uses gradient descent to move rapidly towards error minima.

In the future, we would like to use LSTM as the basis for a continuous speech recognition system. We feel that its power in sequence processing, and its ability to handle multiple timescales simultaneously, make it an ideal candidate for the task. In particular, we hope that it will provide a real alternative to Hidden Markov Models that will not suffer from their limited and ad hoc approach to time dependencies.

References

[1] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. IEEE*, volume 77, pages 257–286. IEEE, 1989.

[2] H.A. Bourlard and N. Morgan. *Connnectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, 1994.

[3] Anthony J. Robinson. An application of recurrent nets to phone probability estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, March 1994.

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

[5] W. Maass, T. Natschläger, and H. Markram. A model for real-time computation in generic neural microcircuits. *Proc. of NIPS 2002, Advances in Neural Information Processing Systems*, 15, 2002. In Press.

[6] D. J. Burr. Speech recognition experiments with perceptrons. In *NIPS*, volume 0, pages 144–153. American Institute of Physics, 1987.

[7] 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.

[8] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent connectionist networks. In Y. Chauvin and D. E. Rumelhart, editors, *Backpropagation: Theory, Architectures, and Applications.* Erlbaum, Hillsdale, NJ, 1990.

[9] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks.* IEEE Press, 2001.

[10] Wolfgang Maass and Christopher M. Bishop. *Pulsed Neural Networks.* The MIT Press, 1998.

[11] Douglas Eck. Finding downbeats with a relaxation oscillator. *Psychological Research*, 66(1):18–25, 2002.

[12] J. J Hopfield. Pattern recognition computation using action potential timing for stimulus representation. *Nature*, 376:3–36, 1995.

[13] S.M. Bohte, H. La Poutré, and J.N. Kok. Spike-prop: error-backprogation for networks of spiking neurons. *Neurocomputing*, 48(1-4):17–37, 2002.

[14] Felix Gers, Nicol Schraudolph, and Juergen Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.

[15] Juan Antonio Pérez-Ortiz, Felix A. Gers, Douglas Eck, and Juergen Schmidhuber. Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, 2002. In press.

[16] F. A. Gers and J. Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[17] Douglas Eck and Juergen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In H. Bourlard, editor, *Neural Networks for Signal Processing XII, Proceedings of the 2002 IEEE Workshop*, pages 747–756, New York, 2002. IEEE.

[18] J. Hopfield and C. Brody. The mus silicium (sonoran desert sand mouse) web page. http://neuron.princeton.edu/ moment/organism/index.html.

[19] Steve Nadis. All together now. *Nature*, 421(6925):780–782, February 2003.

[20] S. Young. *The HTK Book*. Cambridge University, 1995/1996.