Forecasting Electric Energy Demand using a Predictor Model based on Liquid State Machine

Neusa Grando *neusagrando@yahoo.com.br*
CPGEI
Federal University of Technology – Paraná (*UTFPR*)
Curitiba-PR, 80230-901, Brazil

Tania Mezzadri Centeno *mezzadri@utfpr.edu.br*
CPGEI/DAINF
Federal University of Technology – Paraná (*UTFPR*)
Curitiba-PR, 80230-901, Brazil

Silvia Silva da Costa Botelho *silviacb@furg.br*
University of Rio Grande (*FURG*)
Rio Grande-RS, 96201-900, Brazil

Felipe Michels Fontoura *felipe.mfontoura@gmail.com*
CPGEI/DAINF
Federal University of Technology – Paraná (*UTFPR*)
Curitiba-PR, 80230-901, Brazil

Abstract

Electricity demand forecasts are required by companies who need to predict their customers' demand, and by those wishing to trade electricity as a commodity on financial markets. It is hard to find the right prediction method for a given application if not a prediction expert. Recent works show that Liquid State Machines (LSMs) can be applied to the prediction of time series. The main advantage of the LSM is that it projects the input data in a high-dimensional dynamical space and therefore simple learning methods can be used to train the readout. In this paper we present an experimental investigation of an approach for the computation of time series prediction by employing LSMs in the modeling of a predictor in a case study for short-term and long-term electricity demand forecasting. Results of this investigation are promising, considering the error to stop training the readout, the number of iterations of training of the readout and that no strategy of seasonal adjustment or preprocessing of data was achieved to extract non-correlated data out of the time series.

Keywords: Liquid State Machine, Pulsed Neural Networks, Prediction, Electric Energy Demand.

1. INTRODUCTION

Prediction of time series is a very important task in different scientific disciplines [1,2]. Temporal patterns is central to any domain in which time-series data are collected and analyzed with the purpose of making enhanced predictions about the likely outcomes of observed trends or

identifying recurrent patterns. Electricity demand forecasts are required by companies who need to predict their customers' demand, and by those wishing to trade electricity as a commodity on financial markets.

Weather forecasts involves matching a temporal pattern to a classification, whereas predicting the temperature, humidity and barometric pressure requires a more complex function, or at least a function with a greater range of outputs. In general, prediction infers the future states of multiple variables based on the historical states of those variables. This problem has been widely studied by researchers in different areas attempting to build models which will provide useful indications of the complex behaviors of systems like markets, populations, atmospheres and ecologies [3-5].

Artificial Neural Networks (ANNs) are a method used to simulate various traditional approaches to time-series prediction, such as curve-fitting, linear regression, and even AutoRegressive Moving Average (ARMA) stationary stochastic models [6]. The traditional application of these techniques, the building of time-series functions for specific phenomena, is difficult and time-consuming. Neural networks have shown reasonable success in approximating these non-linear functions and their parameters, and while they themselves can require the tuning of many parameters, they promise to greatly enhance the speed with which such analysis may be conducted. A popular approach using ANNs is to learn the prediction from previously collected data. The advantages are that knowledge of the internal structure is not necessarily needed, arbitrary nonlinear prediction could be learned and additionally some past observations could be integrated in the prediction.

Although classical ANNs have been shown to be quite powerful in many domains, their structure is not very well suited to represent temporal patterns [7]. In order to deal with temporal data, one of the possible solutions is to use Recurrent Neural Networks (RNNs). In such recurrent networks connections are incorporated enable the flow of back information for future time steps. Using this recurrence, dynamic temporal patterns can be registered by the network over time. This produces some kind of internal memory that allows obtaining complex functions. This is a central feature for networks that will be used for prediction of time series, because a current output is not solely a function of the current sensory input, but a function of the current and previous sensory inputs and also of the current and previous internal network states. This makes possible a system to incorporate a much richer range of dynamic behaviors.

However, exactly these internal temporal dynamics make it much harder to train the network. The training problem consists of adjusting the parameters (weights) of the network so that it can provide a specific answer for a series of inputs [8]. In order to solve the task of training in RNNs, [9,10] proposed an approach under the name of Reservoir Computing (RC) [11].

Reservoir Computing is a recent architecture for RNNs composed by two main modules. The first one is the 'reservoir' which is a RNN where the recurrent connections are not trained at all. The reservoir is a randomly generated dynamic system with unchanged weights. The training is performed only at the second stage, called 'readout' [10]. RC produces rich dynamics of temporal nature and has been used as a powerful tool for the computation on time series [1,12-14].

In recent years, data from neurobiological experiments have made it increasingly clear that biological neural networks, which communicate through pulses, use the timing of these pulses to transmit information and to perform computation. Based on the RC, Maass and colleagues [9] proposed the Liquid State Machine (LSM) using a reservoir of pulsed Neural Networks (NNs) with integrate-and-fire neurons in combination with simple learning algorithms at readout stage. However the temporal dynamic associated with the treatment of continuous time series is still a challenge for pulsed NN.

In this paper we propose a predictor model using LSM for continuous temporal series prediction applied in electricity demand forecasting. We analyze the dynamic of the network, focusing in the treatment of the temporal memory reset and the conversion of the continuous temporal signal in a

set of pulses for pulsed NN. Our approach is validated in an electricity demand forecast, which is an important challenge for the economic and secure operation of power systems. We describes a prediction method based on LSM applied to a time-series of CEEE (Companhia Estadual de Energia Elétrica do Rio Grande do Sul, Brazil).

The remainder of this paper is organized as follows. The next section provides an overview of the LSM. Section 3 presents the architecture of LSM used for this work and the description of the simulation carried out. The accomplished experiments and associated results are presented in Section 4. Finally, the last section summarizes and concludes this work.

2. THE LIQUID STATE MACHINE

2.1 The framework of a Liquid State Machine

The Liquid State Machine (LSM) has been proposed by Maass and colleagues [9] as a new framework for neural computation based on perturbations [15]. A LSM uses an excitable medium to transform low-dimensional inputs into a high-dimensional 'liquid', so that simple readout unit can extract more detailed temporal features from the input data. Its function resembles a tank of liquid: as the inputs disturb the surface they create unique ripples that propagate, interact and eventually fade away. After learning how to read the water's surface we can extract a lot of information about recent events, without having to do the complex input integration ourselves [16].

To understand the basic idea behind LSMs imagine a pool of water into which various objects are dropped. As the objects enter the liquid, they perturb its surface. The resulting splash and ripples that are created can be transformed in real-time into 'liquid states' (a spatio-temporal pattern of liquid displacement). These ripples propagate over the water's surface for a while and will interact with the ripples caused by other recent events. The water can thus be said to retain and integrate information about recent events, so if we're somehow able to 'read' the water's surface we can extract information about what has been recently going on in this pool. We refer to this trained spectator as a readout unit that we can ask at any one time what's going on in the pool, provided that we can show him a snapshot of the liquid's surface [16].

Thus, LSMs are composed of two parts: a Dynamical Liquid Unit - a model of dynamic liquid flow that is used as a 'reservoir' of complex dynamics to transform the input time series $u(\cdot)$ into 'liquid states' – and a Readout Unit – a simple function which maps the liquid state at time t onto the output.

The idea of the Maass' LSM [9] is shown in figure 1. A continuous input stream $u(\cdot)$ of disturbances is injected into excitable medium L^M that acts as a liquid filter. This liquid can be virtually anything from which we can read it's current liquid state $x^M(t)$ at each time step t. The liquid state is mapped to target output function $y(t)$ by means of a memory-less readout function f^M. This readout map f^M is in general chosen in a task-specific manner (and there may be many different readout maps, that extract different task-specific information in parallel from the current output of L^M) [9,16].

2.2 The Dynamical Liquid Unit

A model of dynamic liquid flow can be implemented using recurrent Spiking Neural Networks (SNNs) which are very powerful tools for solving complex temporal machine learning tasks [17].

The liquid is a non-linear dynamical system composed of a set of spiking neurons pool that receives time-varying input and transforms these different temporal inputs into significantly different liquid states. The task-dependent part is executed by the readout unit that can be trained to extract information from the liquid state transforming this information into a useful form, e.g. into a prediction. The pool of neurons (figure 2) is composed by $N = n_x \times n_y \times n_z$ neurons placed on

a regular grid in 3D space. The number of neurons along the x, y and z axis, n_x, n_y and n_z respectively, can be chosen freely [1].

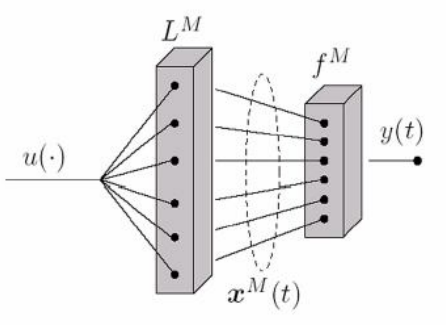

FIGURE 1: Architecture of a LSM [9].

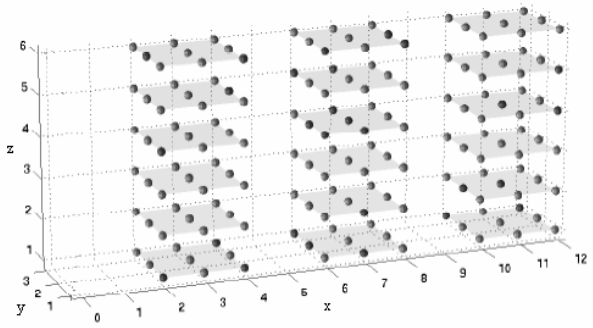

FIGURE 2: Example of the structure of liquid middle of a LSM, formed by a pool of 3×3×6 neurons [18].

2.3 The Readout Layer

The readout units must be capable of detecting stable features between a set of patterns. This unit receives the high-dimensional inputs from the liquid and takes as input the instantaneous reservoir state, i.e. the collection of all of the states of the individual elements, and produces an output decision [16].

A miscellaneous of implementations to the readout unit in LSMs can be achieved [16]. However, a readout unit consisting of just single neurons can obtain nearly the same results as more sophisticated units like pools of perceptrons [19]. Dual liquid-readout modules can also be implemented. In cases where the target output consists of slowly varying analog values, a single readout neuron can be trained to represent these values through its time-varying firing rate [9]. In addition, a low-pass filter can be applied to transform the spike trains into continuous output that can be weighted and fed to an analog readout [16]. In any case the readout neurons can be trained to perform a specific task by adjusting the strengths of synapses projected onto them from the liquid neurons [9].

2.4 Using a LSM: Features and Issues

The LSM works as follows. The input signal $u(\cdot)$ feed the reservoir (pool of neurons). This signal stimulate the neurons in the pool which acts as a filter and the input signal is then transformed into another signal that encapsulates the dynamics of the liquid. Samples of the state of the liquid are taken and form a sequence of vectors, called state vector, which can then be used to train a readout function. Finally, the readout function can be trained using these state vectors to represent the inputs [20].

As a SNN projects the input into a high-dimensional space, the learned readout function can be simple. Also, any snapshot of the state of the network contains information about both current

and past inputs; the waves of spikes produced by input in the past continue to propagate for some time, intermingling with the waves from the current input. This process is referred to as integration of inputs over time. When a network properly integrates inputs over time, a readout function can be memory-less, relying on the network to remember and represent past and current inputs simultaneously [20].

2.4.1 Training

The original LSM concept stated that the dynamic reservoir states can be processed by any statistical classification or regression technique [21]. The key idea underlying LSM (RC) is to train, by supervised learning, only the readout unit while the liquid has fixed weights [17]. The training thus consists of a linear regression problem, which can easily be solved in a way to find the global optimum (for a given reservoir). Therefore, only the readout needs to be trained according to the task. It is not necessary to take any temporal aspects into account for the supervised learning task since all temporal processing is done implicitly in the liquid [22]. Pruning connections from the liquid the readout unit can be used to selecting subsets of variables (i.e. neurons in the reservoir) which are the most relevant for a given target output [23].

2.4.2 Separation and Approximation Properties

An essential requirement of the LSM architecture is that different inputs sequences into the liquid must result in separable outputs based on the liquid's response. The amount of distance created between those is called the Separation Property (SP) of the liquid. The SP reflects the ability of the liquid to create different trajectories of internal states for each of the input classes. The ability of the readout unit to distinguish these trajectories, generalize and relate them to the target outputs is called the universal Approximation Property (AP) of the readout. This property depends on the adaptability of the chosen readout unit, whereas the SP is based directly on the liquid's complexity [16].

Natschläger and colleagues [22] described these two conditions as necessary for computations in time series using LSMs. For this reason, it would be imperative is to create a liquid that effectively separates classes of input into different patterns of state vectors. Besides, the readout must have the capability to distinguish and transform different internal states of the liquid into given target outputs. Schrauwen and Verstraeten [21] mention that the approximation property is satisfied by a simple linear regression function. Since the AP was already close to optimal, the primary limitation in performance lay in the SP [9].

There are so far no design principles to create an 'ideal' liquid state for a special type of input [24]. SP can be engineered in many ways such as incorporating neuron diversity, implementing specific synaptic architectures, altering liquid connectivity, or simply recruiting more columns for neural implementations [9]. Although, Buonomano and Merzenich [25] had already shown that generic recurrent circuits of integrate-and-fire neurons are able to transform temporal input patterns into spatial activity patterns of the circuit. Consequently, it suffices to verify that such recurrent circuits have the SP [26].

The effectiveness of the liquid architecture is affected by a large number of parameters such as size of the reservoir, node types, input connectivity and recurrent connections. These parameters determine the short term memory and separation capability of a reservoir [27]. An inadequate set of parameters limits the potential of the liquid. Thus, the parameter selection has been the topic of much research [11,28,29]. In general, optimization of LSM parameters for applications is based on experience and heuristics and partly on a brute-force search of the parameter space [21].

2.4.3 Interference and Initalization

Closely related to the effectiveness of the liquid is a characteristic of the LSM which generates interference between successive input signals, so that they are merged and transformed into a combined representation. Consequently, the current input and the input history determine the liquid response, due to the recurrent connections. Knüsel and colleagues [30] and Vink [31] showed that a reset mechanism is an essential component to improve the network performance,

since the temporal mixing of information from past and present stimuli can compromise the results. They verified a critical dependence between the initial state of the network at stimulus onset and its internal state after stimulus presentation. Thus, in order to improve the performance, a system initialization at stimulus onset could be required.

2.4.4 Features and Issues associated with the use of LSM

In the next section we propose an approach to predict continuous time series. We introduce the general setup that was used during our experiments to solve the prediction with real-world data from the time series provided by the CEEE (Companhia Estadual de Energia Elétrica do Rio Grande do Sul, Brazil).

3. LSM PREDICTION

3.1 A LSM Architecture to Deal with Continuous Time Series Forecast

This paper describes a model for predicting continuous time series using LSM. The proposal is applied to a case study associated with forecasting future electricity demand.

The architecture of the LSM used in our approach consists of four different modules as seen in figure 3: an input layer that is used to feed data into the liquid, a pool of neurons forming the liquid, an exponential filter that decode the liquid response, and the readout unit which computes an output using the membrane potentials obtained from the liquid neurons.

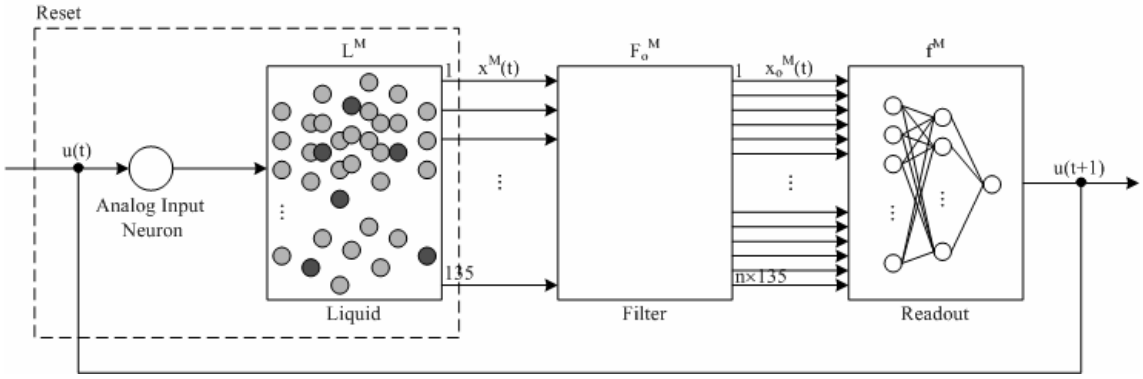

FIGURE 3: Architecture of LSM used in experiments.

The Input Layer. The input layer is an excitatory analog input neuron connected by a static analog synapse to all neurons in the liquid middle scaled to a spectral radius of $|\lambda_{max}| = +\infty$, and $C_{Scale} = +\infty$ with the strength of the synaptic connections scaled to $\lambda = 0.05$. The parameter C_{Scale} specifies how to scale the overall connection probability [9,32], $C_{Scale} = +\infty$ ensures that there will be a synaptic connection between each pair of neurons in the source region and the destination region [18].

The Dynamical Liquid Unit. The liquid L^M (reservoir) consists of a recurrent network composed by a pool of dimension $3\times3\times15$ (135) integrate-and-fire neurons randomly connected via dynamic spiking synapses scaled to a spectral radius of $|\lambda_{max}| = 3$ and $C_{Scale} = 1$ with the strength of the synaptic connections scaled to $\Omega = 1$. Randomly, 20% of the neurons are chosen to be inhibitory.

The Decoder Filter. The liquid response $x^M(t)$ (i.e. the set of all spike times of the neurons in the liquid) is decoded into the liquid state $x_o^M(t)$ (analog values) using an exponential filter f_o^M before being fed into the readout unit f^M.

The Readout Unit. The readout unit consists of a Multi-Layer Perceptron (MLP) network. The input layer of the readout consists of $n\times135$ sigmoidal neurons, where n is the number of sample time points. We sample the state every 100ms. The output layer is a single linear neuron. The hidden layer is composed by 50 sigmoidal neurons. This number of neurons was defined empirically. The readout unit was trained by the Resilient Backpropagation (RPROP) algorithm. It is possible to use a feedback to store the context of the sequence dynamically. In this context a feedback loop is incorporated in order to flow back information for future time steps.

The experiments were done using CSIM (a neural Circuit SIMulator) which is a tool that can simulate heterogeneous networks constituted by different neurons and the synapses, it is written in C++ language with an interface to Matlab®[32]. The multi-'column' neural microcircuits construction is allowed by the Circuit-Tool while the Learning-Tool can analyses the real-time computing in the neural microcircuit models [18,33].

3.2 Simulation

For training and testing the analog time series is split in two parts: 80% for training, and 20% for validation and testing. The time series needs to be normalized to the interval [0, 1], otherwise all neurons would be saturated and thus loosing information. The input vector $u(t)$ is composed of the current input at time step t and the last 14 inputs in the sequence resulting in a vector of fixed length (15 inputs) to represent the past context. This approach builds a predictive model that uses information from the entire sequence, although the vector has a finite length. The size of the vector $u(t)$ was defined by inspecting the seasonality of the time series.

To feed activation sequences into the liquid the input data are provided as vectors of continuous values between 0 and 1, from a single input neuron. The liquid's activity generates a set of 135 state vectors $x^M(t)$ as output. These state vectors are decoded by the filter f_o^M and projected onto the readout unit f^M which predict the value at the time step t+1. The LSM is initialized and reset prior to the presentation of each sequence. Comparing the prediction performance with and without resetting the network reveals that the latter outperforms the former.

The Mean Square Error (MSE) was introduced to evaluate the performance of the method. The evaluation of time series y and its prediction $\hat{y}$ considering N predictions is done by:

$$MSE = \sum_{t=1}^{N} \frac{(y_t - \hat{y}_t)^2}{N}$$

4. EXPERIMENTAL RESULTS

The approach described in the previous section has been applied in seven different experiments to obtain the forecast of electric energy demand in state of Rio Grande do Sul (Brazil). Figure 4 presents the electric energy demand in Rio Grande do Sul (Brazil) for ten years (1998 – 2008). The plot shows that data are marked by a temporal dependence characterizing seasonality. The period November 2006 – September 2008 (20% of the analog time series) has been chosen as a test set to check the forecasting errors and to validate the proposed method.

The experiments 1 and 2 consisted in to predict one time step ahead based on an input vector $u(t)$ composed of the currrent input at time step t and the last 14 inputs in the sequence resulting in a vector of fixed length (15 inputs) representing the past context. The experiments 1 and 2 were achieved without feedback and with feedback respectively. The next 23 samples were predicted based on training set of 92 samples. We used the sigmoidal normalization. The readout was trained until the convergence to the goal after 60 epochs considering an error of 0.001. Figure 5 shows a plot of the network output compared to the target output with feedback and without feedback.

The experiments 3 and 4 were carried out using the same data as described for the experiments 1 e 2, but the readout was trained until the convergence to the goal after 68 epochs considering an error of 0.0005. The experiments 3 and 4 were achieved without feedback and with feedback respectively. Figure 6 shows a plot of the network output compared to the target output.

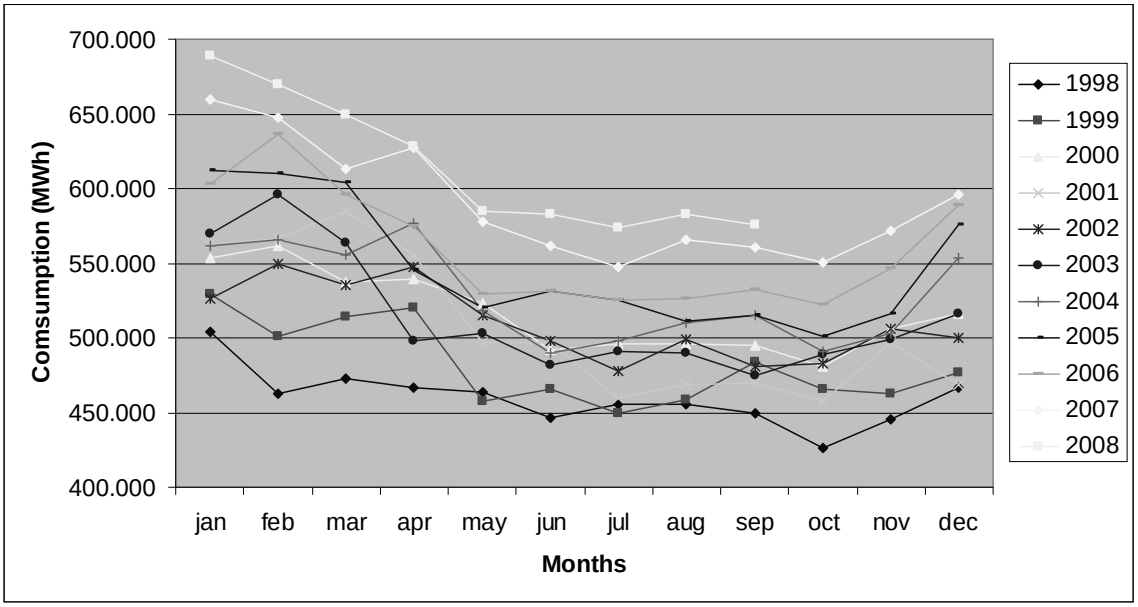

FIGURE 4: Rio Grande do Sul (Brazil) electric energy consumption for ten years (1998 – 2008).

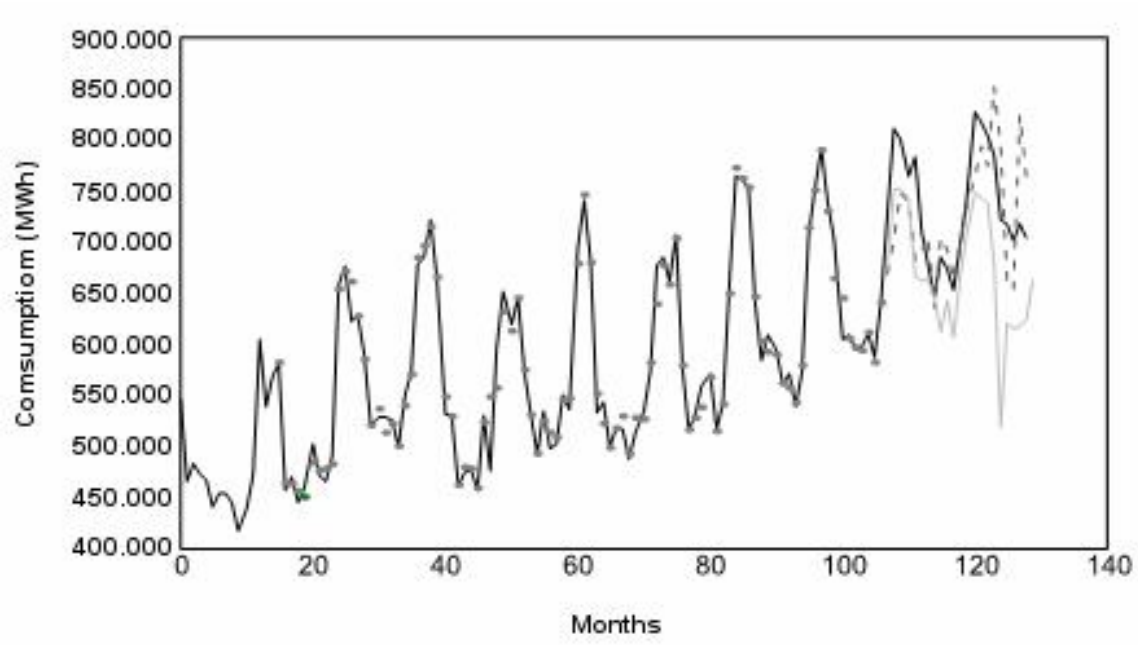

FIGURE 5: Network output versus the desired output for the experiments 1 and 2 training the readout to an error of 0.001. Solid dark gray line: original series, gray points: output of the network for the training set, dashed line: network output without feedback (experiment 1), light gray line: network output with feedback (experiment 2).

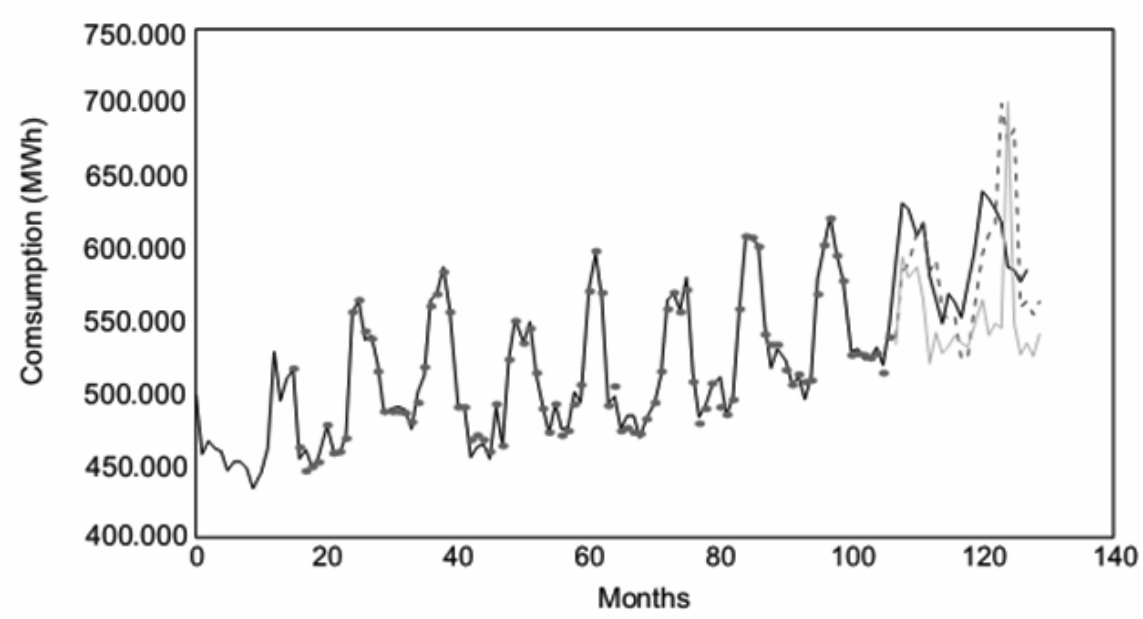

FIGURE 6: Network output versus the desired output for the experiments 3 and 4 training the readout to an error of 0.0005. Solid dark gray line: original series, gray points: output of the network for the training set, dashed line: network output without feedback (experiment 3), light gray line: network output with feedback (experiment 4).

The experiments 5 and 6 were performed to determine the accuracy of the long-term prediction (experiment 5 and 6). These experiments were done to predict the demand for the next 18 years using the same data as described for the previous experiments without feedback (experiment 5) and with feedback (experiment 6). The readout was trained until the convergence to the goal after 134 epochs considering an error of 0.0001. Figure 7 shows the results for these experiments.

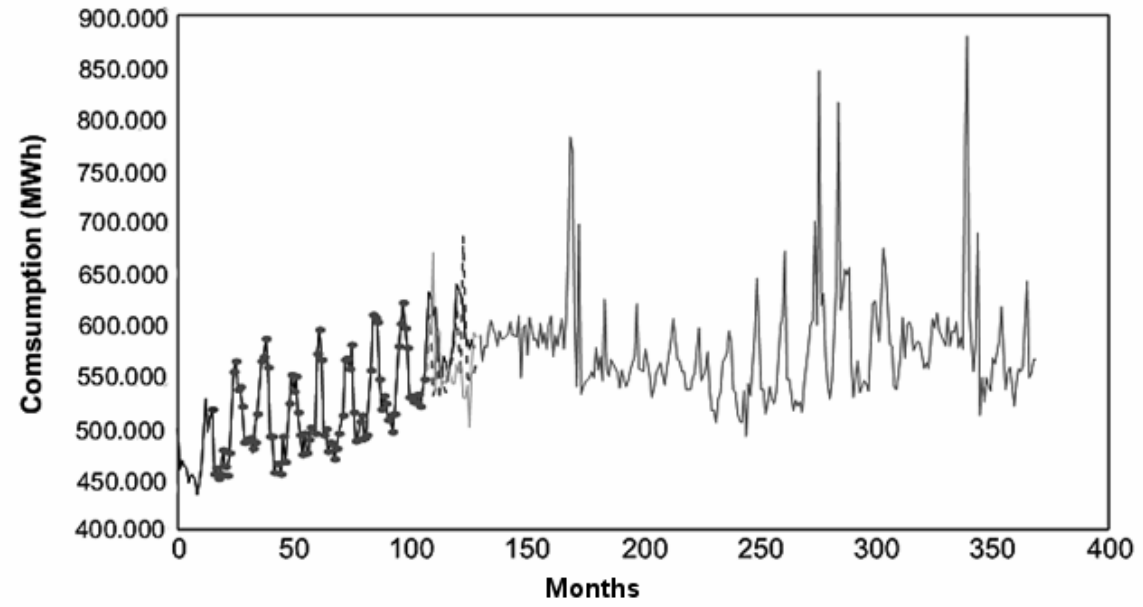

FIGURE 7: Results for the next 18 years with feedback, training the readout to an error of 0.0001. Solid dark gray line: original series, gray points: output of the network for the training set, dashed line: network output without feedback (experiment 5), light gray line: network output with feedback (experiment 6).

An additional experiment (experiment 7) was carried out to predict the demand for the next 20 years. For this experiment we used the same data sets as described for the experiments 1 e 2, but employing all the data contained in the time series for training. The readout was trained with feedback until the convergence to the goal after 280 epochs considering an error of 0.00001. Figure 8 shows a plot of the network output.

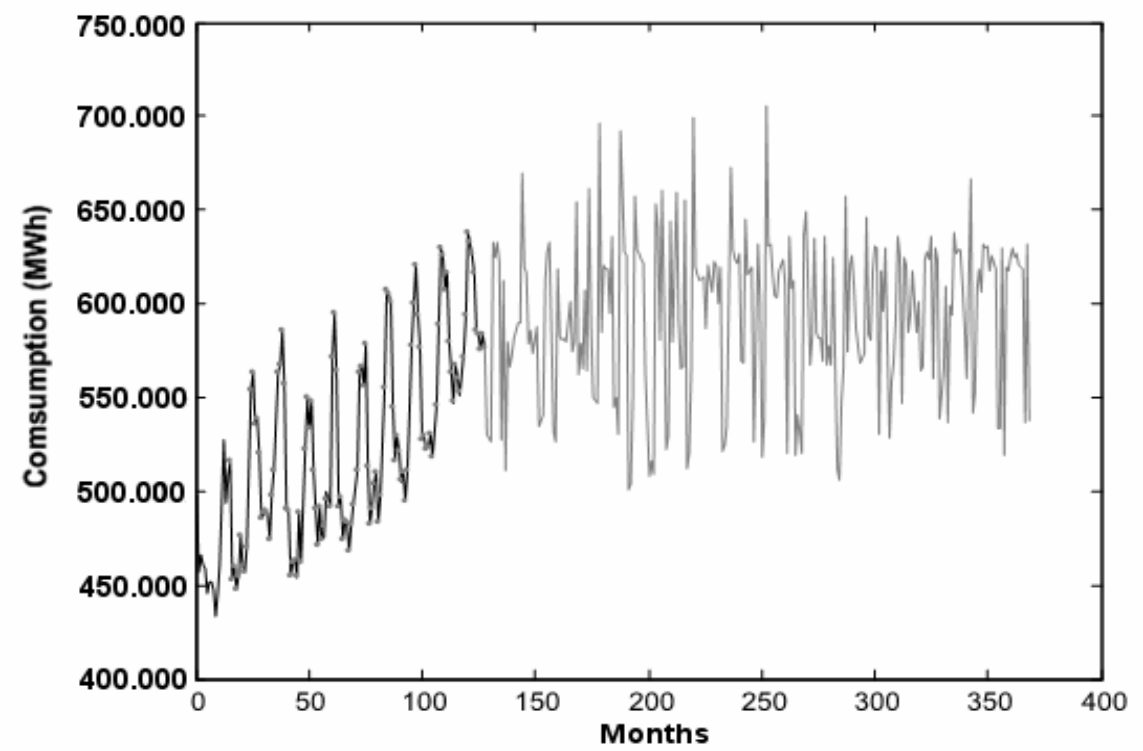

FIGURE 8: Results for the next 20 years with feedback, training the readout to an error of 0.00001. Black line: original series, gray line: output of the network.

The table 1 summarizes the results for the experiments with feedback and without feedback showing the MSE averaged for each experiment and the number of iterations reached. The MSE is not presented for the experiment 7, since the desired output is not available.

Experiment	MSE without feedback	MSE with feedback	MSE readout	N° iterations
1, 2	0.0120	0.0249	0.001	60
3, 4	0.0289	0.0502	0.0005	68
5, 6	0.0274	0.0413	0.0001	134
7	-	-	0.00001	280

Table 1: Comparison of results for the experiments 1-7.

Results shown in Table 1 show us that the model prediction reaches the error minimum in a small number of iterations. What is worth mentioning here is that no strategy of seasonal adjustment or preprocessing of data was achieved to extract non-correlated data out of the time series. Moreover, the data set comprises a small number of samples. Considering the predetermined error (MSE readout) to stop training the readout and the low number of iterations of training achieved by the readout, results of our prediction model are reasonable good.

The model with feedback results in a moderate performance, even for short periods of time. Feedback connections can cause oscillations. Since previous states are used in predictions of future states, these oscillations propagate and cause a degradation of performance.

Not surprisingly, the performance of the model for long periods of time is lower. RC methods tend to be sensitive to a small temporal range [34] which could be one of the reasons of this loss in performance. In addition, the lack of a seasonal adjustment can compromise the performance for the long term prediction since the monthly time series reveal strong seasonal components [2]. This probably produces the saturation of neurons that lose processing power.

5. CONCLUSION & FUTURE WORKS

Demand prediction is of great importance in the electricity supply industry. It is hard to find the right prediction method for a given application if not a prediction expert. This problem has been widely studied by researchers in different areas attempting to devise strategies to obtain accurate time series predictions. Recent works show that LSMs can be applied to the prediction of time series. The main advantage of the LSM is that it projects the input data in a high-dimensional space incorporating valuable information about recent inputs into input signal. This meaningful information allows that simple learning methods can be used to train the readout. As a result we can effectively apply the approach in temporal prediction.

In this paper we have presented an experimental investigation of an approach for the computation of time series prediction by employing LSMs in the modeling of a predictor in a case study for short-term and long-term electricity demand forecasting.

Four different modules composed the architecture of the LSM used in our approach: an input layer that is an excitatory analog input neuron used to feed data into the liquid, a pool of neurons of dimension $3\times3\times15$ (135) integrate-and-fire neurons formed the liquid, an exponential filter that decoded the liquid response, and a readout unit which computed an output obtained from the liquid neurons.

The approach was applied in seven different experiments to obtain the forecast of electric energy demand in state of Rio Grande do Sul (Brazil). The experiments evaluated short-term prediction with and without feedback and long-term prediction with feedback. The input vector of LSM was composed of a sequence of values whose size was defined by inspecting the seasonality of the time series. The information was presented to the LSM in short sequences $u(t)$ of 15 inputs, the current input at time step t and the last 14 inputs. The comparison of prediction performance with and without resetting the network reveals that the latter outperforms the former.

Each sequence was fed into the input layer; this sequence generates a signal of high dimensionality that encapsulates the dynamics of the liquid. Samples of the state of the liquid were taken in different instants of time forming a sequence of vectors, called state vector. The state vector representing the state of LSM for that sequence was decoded by the filter and projected onto the readout unit which predicted the value at the time step t+1 based on the input sequence $u(t)$. After that, the internal states of LSM were reset; a new sequence was presented to the LSM and so successively until all the information has been processed.

Results of this investigation are promising, considering the predetermined error (MSE readout) to stop training the readout, the low number of iterations of training reached by the readout and that no strategy of seasonal adjustment or preprocessing of data was achieved to extract non-correlated data out of the time series.

The results with feedback show a moderate performance even for short periods of time. Probably the oscillations generated by the feedback connections are propagated by the network resulting in a degradation of performance. The same fact occurs for long-term prediction for which the performance was still lower increasing the oscillations on the long timescale. In addition, RC methods tend to be sensitive to a small temporal range which could be another reason of this loss in performance.

The experiments show a reasonable performance of the LSM in predicting the "continuously" streams and that although our predictor model isn't perfect it may be used for prediction of temporal time series problems and can serve as a powerful model for short-term and long-term prediction and pattern classification [35].

Many questions concerning the structure of the LSM are still open, due predominantly to the newness of the conception, which new research will have to address. Further research is encouraged to improve the performance of the LSM by investigating the size, topology and optimal parameters settings of the reservoir for prediction tasks. Also a larger data set is needed for a better comparison.

6. ACKNOWLEDGMENTS

This work was partially supported by the Brazilian National Research Council (CNPq, under research grant n° 304867/2008-0) to T. M. Centeno. N. Grando also acknowledges the financial support from CAPES as a scholarship.

7. REFERENCES

1. H. Burgsteiner, M. Kröll, A. Leopold, G. Steinbauer. *"Movement prediction from real-world images using a liquid state machine"*. Applied Intelligence, 36(2):99-109, 2007
2. F. Wyffels, B. Schrauwen. *"A comparative study of reservoir computing strategies for monthly time series prediction"*. Neurocomputing, 73(10-13):1958-1964, 2010
3. E. Joyce, T. Berg, A. Rietz. *"Prediction markets as decision support systems"*. Information Systems Frontiers, 5(1):79-93, 2003
4. T. Coulson, G. M. Mace, E. Hudson, H. Possingham. *"The use and abuse of population viability analysis"*. Trends in Ecology & Evolution, 16(5):219-2211, 2001
5. S. Nickovic, G. Kallos, A. Papadopoulos, O. Kakaliagou. *"A model for prediction of desert dust cycle in the atmosphere"*. Journal of Geophysical Research, 106(D16):18113-18129, 2001
6. R. Drossu, Z. Obradovic. *"Rapid design of neural networks for time series prediction"*. IEEE Computational Science & Engineering, 3(2):78-89, 1996
7. S. Haykin. *"Neural Networks: a new comprehensive foundation"*, Prentice-Hall, 2nd ed., New Jersey, (1999)
8. A. F. Atiya, A. G. Parlos. *"New results on recurrent network training: unifying the algorithms and accelerating convergence"*. IEEE Transactions in Neural Networks, 11(3):697-709, 2000
9. W. Maass, T. Natschläger, H. Markram. *"Real-time computing without stable states: a new framework for neural computation based on perturbations".* Neural Computation, 14(11):2531–2560, 2002
10. H. Jaeger. *"The "echo state" approach to analysing and training recurrent neural networks".* Technical Report. Fraunhofer Institute for Autonomous Intelligent Systems: German National Research Center for Information Technology (GMD Report 148), 2001
11. D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt. *"An experimental unification of reservoir computing methods"*. Neural Networks, 20(Special Issue):391-403, 2007a
12. A. Lazar, G. Pipa, J. Triesch. *"Fading memory and times series prediction in recurrent networks with different forms of plasticity"*. Neural Networks, 20(3):312-322, 2007
13. L. Pape, J. Gruijl, M. Wiering. *"Democratic liquid state machines for music recognition*". Studies in Computational Intelligence (SCI), 83:191-211, 2008

14. C. Gros, G. Kaczor. *"Semantic learning in autonomously active recurrent neural networks"*. Logic Journal of IGPL Online, jzp045v1-jzp045, 2009

15. H. Paugam-Moisy. "*Spiking Neuron Networks a Survey"*. Technical Report IDIAP RR 06-11. IDIAP Research Institute, 2006

16. J. Vreeken. *"On real-world temporal pattern recognition using Liquid State Machines"*. Master's Thesis, University Utrecht (NL), 2004

17. E. A. Antonelo, B.Schrauwen, X. Dutoit, D. Stroobandt, M. Nuttin. *"Event detection and localization in mobile robot navigation using reservoir computing"*. In Proccedings of 17th International Conference on Artificial Neural Networks, LNCS 4669: 660-669, Porto, Portugal, 2007

18. IGI LSM Group. *"Circuit-Tool: a tool for generating neural microcircuits"*. User Manual. Institute for Theoretical Computer Science, Graz University of Technology. Available in http://www.lsm.turgraz.at, 2006

19. S. Häusler, H. Markram, W. Maass. *"Perspectives of the high dimensional dynamics of neural microcircuits from the point of view of low dimensional readouts".* Complexity, 8(4)39-50, 2003

20. E. Goodman, D. Ventura. "*Spatiotemporal pattern recognition via liquid state machines*". In Proceedings of the International Joint Conference on Neural Networks. Vancouver, BC, Canada, 3848-3853, 2006

21. B. Schrauwen, D. Verstraeten, J. Van Campenhout. *"An overview of reservoir computing: theory, applications and implementations"*. In Proceedings of the 15th European Symposium on Artificial Neural Networks. 471-482, Bruges, Belgium, 2007

22. T. Natschläger, W. Maass, H. Markram. *"The "liquid computer": a novel strategy for real-time computing on time series"*. Special Issue on Foundations of Information Processing of Telematik, 8(1):39-43, 2002

23. X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, M. Nuttin. *"Pruning and regularization in reservoir computing"*. Neurocomputing, 72(7-9):1534-1546, 2009

24. K. P. Dockendorf, I. Park, P. Hea, J. C. Príncipe, T. B. DeMarse. *"Liquid state machines and cultured cortical networks: The separation property"*. BioSystems, 95(2):90-97, 2009

25. D. V. Buonomano, M. M. Merzenich. *"Temporal information transformed into a spatial code by a neural network with realistic properties"*. Science, 267:1028-1030, 1995

26. W. Maass, H. Markram. *"On the computational power of circuits of spiking neurons".* Journal of Computer and System Sciences, 69(4):593-616, 2004

27. A. Ghani, M. McGinnity, L. Maguire, J. Harkin. *"Hardware/software co-desing for spike based recognition"*. Computer Research Repository CoRR abs/0807.2282, 2008

28. E. Goodman, D. Ventura. "*Effectively using recurrently connected spiking neural networks"*. In Proceedings of the International Joint Conference on Neural Networks. (3):1542–1547, Montreal, Canada, 2005

29. D. Verstraeten, B. Schrauwen, D. Stroobandt. *"Adapting reservoirs to get gaussian distributions"*. In Proceedings of the 15th European Symposium on Artificial Neural Networks. 495–500, Bruges, Belgium, 2007b

30. P. Knüsel, R. Wyss, P. König, P. F. M. J. Verschure. *"Decoding a temporal population code"*. Neural Computation, 16(10):2079-2100, 2004

31. R. Vink. *"Temporal pattern analysis using reservoir computing".* Master's Thesis. Leiden Institute of Advanced Computer Science, Universiteit Leiden, 2009

32. T. Natschläger. *"CSIM: a neural Circuit SIMulator".* User Manual. Institute for Theoretical Computer Science, Graz University of Technology. Available in http://www.lsm.tugraz.at/csim/index.html, 2006

33. IGI LSM Group. *"Learning-Tool: analysing the computational power of neural microcircuits"*. User Manual. Institute for Theoretical Computer Science, Graz University of Technology. Available in http://www.lsm.turgraz.at, 2006

34. F. Wyffels, B. Schrauwen, D. Stroobandt. *"Using reservoir computing in a decomposition approach for time series prediction"*. In Proceedings of the European Symposium on Time Series Prediction, 149-158, Provo, Finland, 2008

35. S. Biswas, S. P. Mishra, S. Acharya, S. Mohanty. *"A hybrid oriya named entity recognition system: harnessing the power of rule*". International Journal of Artificial Intelligence and Expert Systems (IJAE), 1(1):1-6, 2010