Dynamical Recurrent Neural Networks – Towards Environmental Time Series Prediction

Alex Aussem †‡, Fionn Murtagh ∓, Marc Sarazin †

†Very Large Telescope Division,
European Southern Observatory, Karl-Schwarzschild-Str. 2,
D-85748 Garching bei München - Germany

‡Université René Descartes, UFR de Mathématiques et Informatique,
45, rue des Saints-Pères, 75006 Paris - France

∓ Space Telescope - European Coordinating Facility,
(Affiliated to Astrophysics Div., Space Science Dep., European Space Agency)
European Southern Observatory, Karl-Schwarzschild-Str. 2,
D-85748 Garching bei München - Germany

Abstract

Dynamical Recurrent Neural Networks (DRNN) (Aussem 1995a) are a class of fully recurrent networks obtained by modeling synapses as autoregressive filters. By virtue of their internal dynamic, these networks approximate the underlying law governing the time series by a system of nonlinear difference equations of internal variables. They therefore provide history-sensitive forecasts without having to be explicitly fed with external memory. The model is trained by a local and recursive error propagation algorithm called temporal-recurrent-backpropagation. The efficiency of the procedure benefits from the exponential decay of the gradient terms backpropagated through the adjoint network. We assess the predictive ability of the DRNN model with meteorological and astronomical time series recorded around the candidate observation sites for the future VLT telescope. The hope is that reliable environmental forecasts provided with the model will allow the modern telescopes to be preset, a few hours in advance, in the most suited instrumental mode. In this perspective, the model is first appraised on precipitation measurements with traditional nonlinear AR and ARMA techniques using feedforward networks. Then we tackle a complex problem, namely the prediction of astronomical seeing, known to be a very erratic time series. A fuzzy coding approach is used to reduce the complexity of the underlying laws governing the seeing. Then, a fuzzy correspondence analysis is carried out to explore the internal relationships in the data. Based on a carefully selected set of meteorological variables at the same time-point, a nonlinear multiple regression, termed *nowcasting* (Murtagh et al. 1993, 1995), is carried out on the fuzzily coded seeing records. The DRNN is shown to outperform the fuzzy k-nearest neighbors method.

Keywords - recurrent neural networks, gradient descent, fuzzy k-nearest neighbor method, time series prediction.

1 Introduction

There has been considerable interest in the past few years in exploring the applications of artificial neural networks for time series identification and prediction (Aussem 1995a, Gershenfeld et al. 1993, Lang et al. 1988, Lapedes et al. 1987, Mozer 1993, Wan 1993, Weigend et al. 1990, Williams et al. 1989). In this paper we present a class of recurrent neural networks called Dynamical Recurrent Neural Networks as very general models for modeling and anticipating the behavior of nonlinear dynamical systems. These networks are obtained by modeling each synapse by an autoregressive (AR) filter. The main attraction of these networks is that they have trainable internal memory

and hence can directly model dynamical systems. In addition, the model includes as particular cases most sigmoidal connectionist networks of the type described by Rumelhart (Rumelhart et al. 1986) that have been introduced to date for temporal processing (Aussem 1995a, Lang et al. 1988, Leighton et al. 1991, Sastry et al. 1994, Tsoi et al. 1994, Wan 1993, Williams et al. 1989).
Predicting the continuation of a time series amounts to picking one of a class of functions so as to approximate the input-output behavior in the most appropriate manner. For deterministic dynamical behaviors, the observation at a current time point can be modeled as a function of a certain number of preceding observations. In such cases, the model being used should have some internal memory to store and update context information. In spite of this, most attempts in the literature at using connectionist models for sequence prediction rely on feedforward networks (Lang et al. 1988, Lapedes et al. 1987, Wan 1993, Weigend et al. 1993). This is achieved by feeding the network with a delayed version of the past observations, commonly referred to as a delay vector or tapped delay line. Even if they include delays such as the time delay neural network (Lang et al. 1988), the autoregressive networks (Leighton et al. 1991), or more recently the FIR networks (Wan 1993), feedforward networks have a finite impulse response and cannot store information for an indefinite time. They learn the memoryless transformation that captures the dependence of the current value of the series on the specified past observations. But the method may fail when the temporal contingencies span unknown intervals. In addition, we illustrate through a generic example that feedforward networks are inadequate for handling a certain class of time series.
Not long ago, several refinements were proposed to circumvent these limitations. In the Jordan-Elman architecture (Jordan 1993), a limited set of carefully chosen recurrent connections are used. These networks do not try to achieve credit assignment back through time but instead use the previous state as part of the current input. Such a simple approach may be seen as a natural extension to feedforward networks in much a same way that ARMA models generalize autoregressive models (Connor et al. 1994). Besides, a generic class of architectures called local-recurrent-global-feedforward has recently been introduced by a number of research groups to benefit from the simplicity of the feedforward training algorithms while using feedbacks either from the activations or from the outputs (Tsoi et al. 1994, Sastry et al. 1994). In this case, feedback is allowed from only the output of the neuron itself, and not from other neurons. Unfortunately, all these models have limited storage capabilities and may be inappropriate to deal with confusing time series.
More attractive is the Williams-Zipser architecture (Williams et al. 1989). It deals with fully interconnected networks with connections of unit delay. The ability of the model to achieve complicated sequence recognition and reproduction tasks has clearly been illustrated through difficult sequence recognition tasks. However, the attractive features displayed by the model are at the detriment of the computational and storage requirements of their forward-like propagation algorithm. Moreover, it has been reported that the architecture typically suffers a lack of stability and appears to be difficult to train optimally (Bengio et al. 1994, Tsoi et al. 1994).
In light of this, a fully *time*- and *space*-recurrent architecture seems to be the ultimate improvement a neural network can be endowed with. Rather than performing a static input-output mapping, the model is described by a system of nonlinear difference equations, the internal variables of which are the hidden unit activations. Now, an algorithm is needed that can properly train fully recurrent DRNN models to implement dynamical systems. In this paper, we describe a powerful learning procedure for networks of this kind, called the temporal-recurrent-backpropagation (TRBP) algorithm. TRBP is local in time like the forward propagation algorithms and requires computation only proportional to the number of weights. The derivation of this algorithm has been given elsewhere (Aussem 1995a, 1995b) but is repeated in this paper for convenience.
The DRNN is used in our studies as a generic dynamical model trained to emulate the behavior of environmental time series recorded on candidate observation sites for the European Southern

Observatory (ESO). The framework pursued in this paper attempts to propose reliable observation quality estimates for the ground-based telescope to be set, some time in advance, in the most suited mode to allow an efficient use of the observing time. We experiment with two real-word time series, namely the quarterly precipitation recorded in Namibia, and the fuzzily coded astronomical seeing. These series are typical cases where the attempt to derive physical equations governing their behavior failed. The practical experience and theoretical knowledge of the meteorologists points to the inherent difficulty of deriving rules for local forecasting. Therefore prediction is sought without reference to any underlying physical model. Our main goals in the experiments described here were 1) to assess the predictability of these time series, especially the seeing, and 2) to appraise the DRNN models with standard neural-based techniques and state space reconstruction approaches. The precipitation records were used as a toy series to appraise our model whereas seeing, measured in La Silla and in Paranal, is a variable for which the astronomical community shows great concern because it strongly influences the observation quality (Murtagh et al. 1992, 1993, 1995). In order to lessen the complexity of the task devoted to the neural network, an intuitively plausible and interpretation-friendly form of coding was adopted, consisting of "fuzzifying" the seeing measurements. Such coding is particularly appropriate since seeing categories ("very good", "very poor", etc.) are especially helpful for interpretation. A fuzzy correspondence analysis was used to explore the initial relationships in the data and select the most relevant variables to be used.

The paper is organized as follows. In section 2, we briefly review the neural-based techniques that have been employed so far as *nonlinear* extensions of traditional linear models with a view to highlighting their differences and ascertaining the problems for which they are best suited. Section 3 presents the architecture of the DRNN model. The complete derivation of the TRBP procedure is carried out in section 4. A thorough discussion of its asymptotic properties is also provided. Various forecasting techniques are appraised and simulation results are presented in section 5 to illustrate the power of the DRNN and assess the predictability of series of interest.

2 Connectionist Models for Time Series Prediction

The standard approach in time series prediction involves constructing an underlying model which gives rise to the observed sequence denoted by $\{x_t\}$. Because the explicit equations of the underlying model are usually unknown, the rules that govern the system evolution must be inferred from the regularities of the past observations. In this perspective, connectionist models are legitimate choices for those involved in forecasting techniques without any a priori knowledge of the characteristics of the time series under consideration.

Neural networks have recently emerged as a successful tool in the fields of sequence recognition and prediction (Aussem et al. 1994a, 1994b, Connor et al. 1994, Gershenfeld et al. 1993, Lapedes et al. 1987, Mozer 1993, Wan 1993, Weigend et al. 1990). This is due to the versatility of a three-layer feedforward neural network in approximating an arbitrary static nonlinearity (Cybenko 1989), and the computational efficiency of the backpropagation algorithm (Rumelhart et al. 1986, Werbos 1972). In addition, the promise of neural networks comes from the hope that they can emulate unanticipated features of the time series without requiring insight into the underlying mechanism. These arguments provide the basic motivation for the use of neural networks in time series prediction.

Though connectionist models have the somewhat appealing ability to infer rules to perform generalization, this is nothing more than real-valued function interpolation of the input-output mapping. Also, for modeling general nonlinear dynamical systems or complex time series, we discuss some cases for which the representation capability of feedforward networks is inadequate and hence we propose dynamic recurrent neural network models to circumvent their weakness.

A. *Nonlinear AR and ARMA models*

The theory of prediction focuses on optimum prediction of dynamic systems in the minimum mean square sense. Let $\hat{x}_t$ denote the single-step estimate for the desired value x_t recorded at time point t. In a stationary stochastic environment, minimizing the expectation of the squared error corresponds to predicting the conditional mean of x_t given the infinite past observations $x_{t-1}, x_{t-2}, \ldots$.

$$\hat{x}_t = \mathrm{E}[x_t \mid x_{t-1}, x_{t-2}, \ldots] \tag{1}$$

A general class of models which appear to offer plausible description of a wide range of different types of time series is the Nonlinear Autoregressive Moving Average (NARMA) model (Box et al. 1976). Therefore it seems reasonable to approximate the conditional mean predictor (1) by a more general function $f()$ such that

$$\hat{x}_t = f(x_{t-1}, \ldots, x_{t-p}, e_{t-1}, \ldots, e_{t-q}) \tag{2}$$

where

$$e_t = x_t - \hat{x}_t \tag{3}$$

The static feedforward network is now driven by a limited version of past known error residuals in addition to the past observations. These errors in turn depend on the past values of its own output $\hat{x}_t$, rendering the standard backpropagation algorithm inadequate for adjusting the network parameters properly.

The hope is that the predictive ability of feedforward networks is improved, insofar as more information is made available to the network. Such a general NARMA scheme is a natural generalization of optimal linear predictors (Box et al. 1976, Connor et al. 1994) and provides a valuable framework for modeling a wide range of different types of nonlinear time series structures. NARMA models have an advantage over NAR models in much the same way that linear ARMA models have advantages over AR models for some types of series (Connor et al. 1994).

Despite this observation, prediction based on NARMA models has not amassed a large literature, owing to the computational burden involved in performing properly credit assignment back through time. Instead, the neural network community has mostly been concerned with NAR models by setting the terms $e(t)$ to zero in (2) so that standard backpropagation could be used at low cost. In addition, the use of NAR models fed with *delay coordinate vectors* has been recently underpinned by theoretical results in dynamical system theory (Ott 1993). NAR models based on feedforward networks have shown great promise not only for the prediction of chaotic processes, but also for certain real-world time series (Wan 1993, Weigend et al. 1990). Nonetheless, the length of delayed vector in (2) must be chosen in advance to accommodate the temporal contingencies present in the input sequence often spanning unknown time intervals. Also this technique provides rich but not total information about the past.

Finally, we mention that both the AR and ARMA models are extrapolated from rather old methods of linear difference equations. In order to increase the potential of neural-based approaches to forecasting, we present in this paper a modern approach drawing from control theory.

C. *The Need for Fully Interconnected Networks*

Connor et al. (1994) have clearly demonstrated that a NARMA network, in which the output is fed back to the input, is a special case of the somewhat more general class of fully interconnected networks. For such networks, there is no need to present the $\hat{e}_{t-j}$ to the input since specific non-trainable recurrent links from a hidden layer towards itself result in the same functional input-output mapping.

On the other hand, it is reasonable to assume the time series to be governed by an underlying system of difference equations with known but also unknown internal variables, i.e. not available for measurements. This case is very usual in meteorology (e.g. Lorenz equation) but is also observed in many fields of physics (e.g. particle motion, laser fluctuations, etc.). In view of the connection between time series and dynamical system analysis, we may regard the time series as a dynamical system driven by its past outputs instead of exogenous signals. When all the states of the system are measurable in addition to its output, it is well known that state-output models (Narendra et al. 1991, Srinivasan et al. 1994) can accurately represent a large class of nonlinear dynamics. Nonetheless, the states underlying a time series, if they exist, are not measurable. Only the past observations can be used for regression.

We illustrate through a simple generic example that NARMA models of the type discussed above, are inadequate to forecast the continuations of such types of time series endowed with unknown internal variables. Consider a series $\{x_t\}$ governed by

$$\left\{ \begin{array}{lcl} x_t & = & f_1(z_t) \\ z_t & = & f_2(z_{t-1}) + \epsilon_t \end{array} \right. \tag{4}$$

where ϵ_t is a sequence of zero mean independent random variables satisfying $\mathrm{E}(\epsilon_t | z_{t-1}, z_{t-2}, \ldots) = 0$ and z_t is the unique internal variable underlying the dynamic of x_t. Now it is readily seen that the optimal predictor is given by

$$\hat{x}_{t+1} = f_1(f_2(f_1^{-1}(x_t))) \tag{5}$$

The invertibility of the mapping $f_1()$ is a condition *sine qua non* for NARMA models to perform the task. Otherwise, a NARMA model based on a feedforward network will fail to anticipate the continuation of the series. As pointed out by Srinivasan et al. (1994), NARMA models are not capable of representing the system if the underlying state reconstruction is not possible. This points to the need for more involved models capable of handling these situations. Because there is some potential for dynamic fully-interconnected networks to provide a better approximation compared to feedforward networks with or without external feedbacks, these models offer the promise to tackle complex problems which have not proven amenable with traditional approaches. This is the reason for the appeal of recurrent networks in recent years in the context of temporal processing.

3 The DRNN model

3.1 Autoregressive Recurrent Connections

This section describes the connectionist model we used to assess the predictability of environmental time series. To remedy for the weakness of NARMA models, we show that the problem of recurrence has to be explicitly tackled when the dynamic of a general time series has to be learned. Temporal association tasks require a neural network that will store and update context information spanning *a priori* unknown temporal intervals. And for the network to remember some aspects of the past, the network must hold memory of its internal state. Thus, one has to build into the network as much knowledge of the past as possible.

Therefore, a key modification of the basic neuron model can be accomplished by replacing each synaptic weight by an AR filter. This basic modification of the neuron interactions allows a continuum of memory forms ranging from high *resolution* and low *depth* to low *depth* and high *resolution* (Mozer 1993). Using the terminology adopted in signal theory, neuron interactions are treated as Finite Impulse Response (FIR) filters. By this, we mean that for an input excitation of finite duration, the output will also be of finite duration. The model is referred to as a dynamic

recurrent neural network (DRNN) since its underlying law governing its evolution is a system of difference equations. This will be discussed later on.
The filter parameters are adjusted continuously so as to select those aspects of the input sequence which are relevant for making temporal associations. This motivates the belief that such an internal dynamic enables the network to find out an appropriate internal state representation which allows the time series behavior to be captured. In addition, a time-lagged vector is not required anymore, It suffices to present only a single input unit and have the network select the most relevant delayed observations to reconstruct the underlying state of the system. As a result, the DRNN are a class of networks that can handle complex tasks that require the preservation of state over potentially unbounded periods of time (Williams et al. 1989).
Note that this idea is not new and has been used by Wan (1993) with feedforward networks. These models, referred to as FIR networks, have shown considerable promise in the field of prediction (Gershenfeld et al. 1993). We emphasize however that there are no feedbacks in this model. The difference from a static feedforward network is a matter of representation convenience and also a more desirable adaptation scheme for on-line training. The architecture is also known as a time-delay neural network (Lang et al. 1988) or autoregressive network (Leighton et al. 1991).
Finally, the DRNN is nothing more than a fully-interconnected FIR network. But it is crucial to note that learning with standard backpropagation (Rumelhart et al. 1986) is ruled out due to recursion. In the rest of the paper, issues encountered in updating the DRNN "parameters amidst recurrence" are addressed.

3.2 A Relaxationwise Dynamic

In this section, the ingredients needed to specify the DRNN model are presented. Consider a fully connected network of N continuous-valued units. Let the matrix element w_{ij}^d denote the connection strength, or coupling, from the d-th delay link connecting node i to node j and D the maximum connection delay, i.e. the filter order. The response of a unit is called its activation value (also called unit output). $\mathbf{v}_k$ ($\mathbf{v}_k \in R^{[N\times 1]}$) is the vector activity of the units at time k. Some of these units feeding input patterns into the rest of the network have their activation clamped to $\mathbf{i}_k$ ($\mathbf{i}_k \in R^{[N\times 1]}$). We do not allow for connections towards input units, i.e. $w_{ij}^d = 0$ if j is an input unit, and set $i_k(j) = 0$ if unit j is not an input unit. Similarly, some are termed output units with target values $\mathbf{d}_k$ ($\mathbf{d}_k \in R^{[N\times 1]}$). Units having neither input nor output connections to the outside world are called *hidden* units. In addition, a *bias* input to each unit is achieved by fixing a particular unit activation to 1.0. For notational convenience, we won't distinguish between input, hidden, output and bias units in the following formulae.
Continuing with notation, the weights for the d-th delay connections heading to unit j at iteration k are specified by the vector $\mathbf{w}_k^d(j) = [w_k^d(1,j),\ldots,w_k^d(N,j)]^T$. Similarly, the $N\times N$ matrix $\mathbf{W}_k^d = [\mathbf{w}_k^d(1),\mathbf{w}_k^d(2),\ldots,\mathbf{w}_k^d(N)]$, contains all adaptive weights of delay $d \leq D$ at iteration k. The initial vector $\mathbf{W}_0^d$ is usually produced by a random number generator. Yet, we will see later on that preliminary conditions on $\mathbf{W}_0^d$ have to be fulfilled in order to ensure the stability of the updating laws.
We determine the relevant system parameters at each step k by the relaxation of the following system of coupled differential equations:

$$\begin{aligned}
\mathbf{T}\frac{d\mathbf{v}_k(t)}{dt} &= -\mathbf{v}_k(t) + g(\mathbf{h}_k(t)) + \mathbf{i}_k \\
\mathbf{h}_k(t) &= \sum_{d=1}^{D} \mathbf{W}_k^{d^T}\mathbf{v}_{k-d} + \mathbf{W}_k^{0^T}\mathbf{v}_k(t)
\end{aligned} \tag{6}$$

The differential equations which express the network update scheme are of particular interest for an analog very-large scale implementation. $\mathbf{T} = [\tau_1, \ldots, \tau_N]$ is a conveniently chosen positive time constant vector, $\mathbf{h}_k$ is the unit internal input vector at iteration k and

$$\begin{aligned} g(\mathbf{h}_k) &= [g(h_k^1), \ldots, g(h_k^n)]^T \\ g(h) &= \frac{1}{1+e^{-\beta h}} \end{aligned} \tag{7}$$

The composition of an affine mapping followed by a sigmoidal function is a common choice for the nonlinear activation function. The parameter β controls the steepness of the logistic function and is arbitrarily set to $1/2$.

It is not guaranteed that Eq. 6 always settles down to a stable equilibrium solution, and an oscillatory or chaotic behavior might be possible. We will not discuss this further and assume the existence of at least one stable fixed point attractor. Therefore, if such a solution exists for system (6), it is readily seen that it verifies

$$\mathbf{v}_k = g(\sum_{d=0}^{D} \mathbf{W}_k^{d^T} \mathbf{v}_{k-d}) + \mathbf{i}_k \tag{8}$$

Note that all N units must keep track of their D past outputs. This involves an overall bookkeeping of $(D+1)N$ values.

As mentioned earlier, rather than using a time-lag vector and performing a global nonlinear approximation of the map underlying the time series, the mutual nonlinear dynamical interactions between neighboring units enable the DRNN to extract, encode and hold information pertaining to the input sequence. The hidden units act as internal states of a complex nonlinear system of recursive equations. Essentially, the DRNN not only makes use of prior observations but it also holds a memory of its previous states to provide context-sensitive forecasts.

At this point, it is important to understand that the problem inherent to the training of a dynamical neural network differs in a significant way from that faced with static networks. The DRNN is not only trained to adjust each individual weight matrix element but, moreover, the algorithm has also to find the most adapted encoded representation of the input sequence within the constraints imposed by the network size and topology.

Let x_t be again the time series to be predicted, and $M < N$, the number of non-input units. From the discussion above, the DRNN offers the hope to approximate the behavior of x_t to the degree that the underlying law of the series is reliably approximated by a deterministic system of recursive difference equations of the form

$$\begin{aligned} v_t^j &= f_j(x_t, \ldots, x_{t-D}, v_t^1, \ldots, v_{t-D}^1, \ldots, v_t^N, \ldots, v_{t-D}^N) \text{ for } j = 1, \ldots, M \\ x_{t+1} &= h(x_t, \ldots, x_{t-D}, v_t^1, \ldots, v_{t-D}^1, \ldots, v_t^N, \ldots, v_{t-D}^N) \end{aligned} \tag{9}$$

insofar as $f_j()$ for $j = 1, \ldots, M$ and $h()$ are "well behaved" functions, accurately approximated by the combination of logistic and linear functions as defined in Eq. 6. Notice that v_t^j, the output of the j-th variable, is defined recursively. Thus x_{t+1} is obtained after the system has settled to a fixed equilibrium state on each cycle.

This construction forms a more powerful representation than that possible in a NARMA model and exhibits appealing characteristics. Besides, we note that (9) extends the Kalman estimators to the nonlinear case. The DRNN can be regarded as a *generic* dynamic system at the outset of training that adapts itself to emulate the dynamic of the system to be learned.

Let us return to our initial series (4). The versatile representation capability of DRNN models includes as a particular case the dynamic of this series provided $f_1()$ and $f_2()$ are smooth enough mappings. A sufficiently large number of hidden units will theoretically suffice to emulate the series. The problem we are now faced with is whether the training algorithm will solve the problem in a reasonable amount of time. And this points to the important issue of whether updating the weights along the negative gradient is sufficient for convergence to the optimal solution.

4 Gradient Descent in the DRNN

In purely feedforward networks, the gradient is straightforwardly obtained by standard backpropagation (Rumelhart et al 1986). But as recurrence is present in the DRNN model, the past parameters affect the present output. More precisely, due to the present output depending upon the past output, a small change in a parameter value will influence the future evolution of the whole dynamical system. So the gradients with respect to past and present parameters have to be determined for proper parameter adaptation.

For the calculation of the "gradient amidst dynamic", we resort to a peculiar chain rule expansion of the partial derivatives. The mathematical concepts underlying this chain rule expansion are developed in Werbos (1972), and provide a formal technique for calculating gradients in general discrete-time dynamical systems. Readers interested in further insights into these concepts may refer to Piché (1994).

For the calculation of the derivatives to be carried out properly, we express the DRNN in a general framework based upon a standard representation commonly used for the control of dynamical systems (Narendra et al. 1991, Srinivasan et al. 1994). The input of the network is composed of two components. The first component is made up of the external input $\mathbf{i}_k$ and D delayed versions of this vector $\mathbf{i}_k, \ldots, \mathbf{i}_{k-D}$. The second components are the delayed state vectors $\mathbf{v}_{k-1}, \ldots, \mathbf{v}_{k-D}$. By including *all* feedback states of the DRNN as an additional input vector, we would like to express the DRNN in the standard representation

$$\mathbf{v}_k = f(\mathbf{i}_k, \ldots, \mathbf{i}_{k-D}, \mathbf{v}_{k-1}, \mathbf{v}_{k-2}, \ldots \mathbf{v}_{k-D}, \mathbf{W}_k^0, \ldots, \mathbf{W}_k^D) \tag{10}$$

where the function $f()$ contains no delays (Narendra et al. 1991, Piché 1994). But as the function $f()$ is not analytically defined for the DRNN model, we are forced to resort to Eq. 8 which is recursive in terms of $\mathbf{v}_k$. This takes us a bit further from the framework of standard derivative rule expansions. Due to the recursivity of (10), we shall introduce now another formalism. The DRNN representation is depicted in Fig. 1.

For this purpose, consider a set of n variables $z_1, \ldots, z_n$ whose values are determined by a set of n *ordered* equations

$$z_i = f_i(z_1, \ldots, z_{i-1}, z_i) \tag{11}$$

where each variable z_i is not only a function of $z_1, \ldots, z_{i-1}$ but also of itself. Let I denotes the identity matrix; assuming $I - \partial f_j / \partial z_j$ to be invertible, we have

$$\left[\frac{\partial z_j}{\partial z_i}\right]_{\{z_{i+1}, \ldots, z_{j-1}\}} = (I - \frac{\partial f_j(z)}{\partial z_j})^{-1} \frac{\partial f_j(z)}{\partial z_i} \tag{12}$$

Note that z_i may refer to either a scalar or a vector. Without loss of generality, it is assumed that z_i is a vector so that $\partial z_j / \partial z_i$ of dimension $\dim(z_j) \times \dim(z_i)$ denotes the jacobian of z_j with respect to z_i.

Given (12), the standard chain rule we will use may now may be written as

$$\frac{\partial z_j}{\partial z_i} = \left[\frac{\partial z_j}{\partial z_i}\right]_{\{z_{i+1},\ldots,z_{j-1}\}} + \sum_{k=i+1}^{j-1} \frac{\partial z_j}{\partial z_k}\left[\frac{\partial z_k}{\partial z_i}\right]_{\{z_{i+1},\ldots,z_{k-1}\}}$$
$$= \sum_{k=i+1}^{j} \frac{\partial z_j}{\partial z_k}\left(I - \frac{\partial f_k(z)}{\partial z_k}\right)^{-1} \frac{\partial f_k(z)}{\partial z_i} \tag{13}$$

Given (13), we are equipped with the mathematical tools to proceed to the derivation of the algorithm.

4.1 Backward Recurrent Propagation

Classically, our goal is to find a procedure to adjust the weight components w_{ij}^d so that a given fixed initial conditions $\mathbf{v}_0$ and a set of input vectors $\{\mathbf{i}_k\}$ taken from the training set result in a set of fixed points, whose components along the output units have a desired set of values $\{\mathbf{d}_k\}$. This will be accomplished by our minimizing a quadratic function

$$E = \frac{1}{2}\sum_{k=0}^{k_f} \mathbf{e}_k^T \mathbf{e}_k \tag{14}$$

which measures the errors $\mathbf{e}_k = \mathbf{d}_k - \mathbf{v}_k$ between the desired and the actual fixed point values along a trajectory (or epoch) starting at iteration 0 up to k_f. Assuming that the adaptation rule takes place at the end of the trajectory, the DRNN is thus governed by an ordered set of equations

$$\begin{cases} \mathcal{W}_k &= \mathcal{W} \\ \mathbf{v}_k &= f(\mathbf{i}_k, \ldots, \mathbf{i}_{k_f - D}, \mathbf{v}_k, \ldots, \mathbf{v}_{k-D}, \mathcal{W}_k) \end{cases} \tag{15}$$

for $k = 0, \ldots, k_f$, where $\mathcal{W}_k$ is a length $(D+1)N^2$ column vector made up of the concatenation $\mathbf{w}_k^d(j)$ for $j = 1, \ldots, N$ and $d = 0, \ldots, D$. What is important is that $\mathcal{W}_k$ is a vector containing all adaptive parameters of the system at some iteration k, not its particular structure.
For the purposes of derivation, we define the vector $\mathcal{W}$ as the first ordered variable, followed by $\mathcal{W}_0, \mathbf{v}_0, \ldots, \mathcal{W}_{k_f}, \mathbf{v}_{k_f}$. Because the error function is computed at the end of the trajectory, E is taken as the last variable.
Classically, a new weight vector is generated at iteration k_f according to

$$\Delta\mathcal{W} = -\eta \frac{\partial E}{\partial \mathcal{W}} \tag{16}$$

where η, the learning rate, is a suitably chosen positive constant. Our primary goal in deriving the algorithm is to find an expression for the term $\partial E/\partial\mathcal{W}$. The repeated application of the chain rule (13) for the computation of $\partial E/\partial\mathcal{W}$ results in a *backsweep* algorithm also termed *backward* recurrent propagation algorithm. We stick to the more intuitive terminology "backward propagation" as it refers to the error signal flows through the links (Aussem 1995a, Srinivasan et al. 1994).
An understanding of the asymptotic behavior of the gradient terms propagated in a reverse fashion will allow us to transform the latter into a very efficient iterative procedure. We begin by noting that $\mathcal{W}_k$ are the only variables which are direct functions of $\mathcal{W}$. Therefore, applying the first chain rule expansion (13) to the partial derivative $\partial E/\partial\mathcal{W}$ collapses to

$$\frac{\partial E}{\partial \mathcal{W}} = \sum_{k=0}^{k_f} \frac{\partial E}{\partial \mathcal{W}_k}\frac{\partial \mathcal{W}_k}{\partial \mathcal{W}} = \sum_{k=0}^{k_f} \frac{\partial E}{\partial \mathcal{W}_k} \tag{17}$$

Once again, we use the chain rule to expand $\partial E/\partial \mathcal{W}_k$. Since E is not a direct function of $\mathcal{W}_k$, we obtain

$$\frac{\partial E}{\partial \mathcal{W}_k} = \sum_{j=k}^{k_f} \frac{\partial E}{\partial \mathbf{v}_j}(I - \frac{\partial f_j}{\partial \mathbf{v}_j})^{-1} \frac{\partial f_j}{\partial \mathcal{W}_k} + \sum_{j=k+1}^{k_f} \frac{\partial E}{\partial \mathcal{W}_j} \frac{\partial \mathcal{W}_j}{\partial \mathcal{W}_k} \tag{18}$$

The components of the vector $\partial \mathcal{W}_j/\partial \mathcal{W}_k$ are equal to zero since $j \neq k$. The term $\partial f_j/\partial \mathcal{W}_k$, which represents the partial derivative of f at time j with respect to $\mathcal{W}_k$, is nonzero only when $k = j$. $\partial f_j/\partial \mathbf{v}_j$ can be obtained from (6), this gives

$$\frac{\partial f_k}{\partial \mathbf{v}_{k-j}} = \mathbf{G'}_k {\mathbf{W}_k^j}^T \tag{19}$$

where $\mathbf{G'}_k$ is given by

$$\mathbf{G'}_k = 2\beta \mathbf{G}_k(1 - \mathbf{G}_k) \tag{20}$$

and $\mathbf{G}_k$ is the $N \times N$ diagonal matrix constructed from $g(h_k(j))$ for $j = 1, \ldots, N$. (Recall that $g'(h) = 2\beta g(h)(1 - g(h))$). Using these results, we find $\partial E/\partial \mathcal{W}_k$ to be

$$\frac{\partial E}{\partial \mathcal{W}_k} = \frac{\partial E}{\partial \mathbf{v}_k}(I - \frac{\partial f_k}{\partial \mathbf{v}_k})^{-1} \frac{\partial f_k}{\partial \mathcal{W}_k} = \frac{\partial E}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k {\mathbf{W}_k^0}^T]^{-1} \frac{\partial f_k}{\partial \mathcal{W}_k} \tag{21}$$

Substituting (21) into (17) gives

$$\frac{\partial E}{\partial \mathcal{W}} = \sum_{k=0}^{k_f} \frac{\partial E}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k {\mathbf{W}_k^0}^T]^{-1} \frac{\partial f_k}{\partial \mathcal{W}_k} \tag{22}$$

All components of the matrix $\partial \mathbf{W}_j/\partial \mathbf{v}_k$ are equal to zero. Thus, using the chain rule once again on the term $\partial E/\partial \mathbf{v}_k$, we find

$$\frac{\partial E}{\partial \mathbf{v}_k} = \mathbf{e}_k^T + \sum_{j=k+1}^{k_f} \frac{\partial E}{\partial \mathbf{v}_j}(I - \frac{\partial f_j}{\partial \mathbf{v}_j})^{-1} \frac{\partial f_j}{\partial \mathbf{v}_k} \tag{23}$$

The terms $\partial f_j/\partial \mathbf{v}_k$ are nonzero for $j \leq k + D$. Using these results and substituting (19) in (23), we finally obtain

$$\frac{\partial E}{\partial \mathbf{v}_k} = \mathbf{e}_k^T + \sum_{j=1}^{D} \frac{\partial E}{\partial \mathbf{v}_{k+j}}[\mathbf{I} - \mathbf{G'}_{k+j} {\mathbf{W}_{k+j}^0}^T]^{-1} \mathbf{G'}_{k+j} {\mathbf{W}_{k+j}}^T \tag{24}$$

Equation (24) is a backward difference equation that can be solved using the following boundary conditions

$$\frac{\partial E}{\partial \mathbf{v}_{k_f}} = \mathbf{e}_{k_f}^T \tag{25}$$

$$\frac{\partial E}{\partial \mathbf{v}_{k_f+1}}, \ldots, \frac{\partial E}{\partial \mathbf{v}_{k_f+D}} = \mathbf{0} \tag{26}$$

Now define

$$\mathbf{y}_k{}^T = \frac{\partial E}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k {\mathbf{W}_k^0}^T]^{-1} \tag{27}$$

Substituting (27) in (24), we obtain $\mathbf{y}_k$ by backward iterating the equation

$$\mathbf{y}_k{}^T = [\mathbf{e}_k^T + \sum_{j=1}^{D} \mathbf{y}_{k+j}{}^T \mathbf{G'}_{k+j} \mathbf{W}_{k+j}^{j}{}^T][\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^0{}^T]^{-1} \tag{28}$$

Transposing (28) gives

$$\mathbf{y}_k = [\mathbf{I} - \mathbf{W}_k^0 \mathbf{G'}_k]^{-1}[\mathrm{e}_k + \sum_{j=1}^{D} \mathbf{W}_{k+j}^{j} \mathbf{G'}_{k+j} \mathbf{y}_{k+j}] \tag{29}$$

Now recall that $\mathbf{W}_k^d = [\mathbf{w}_k^d(1), \mathbf{w}_k^d(2), \ldots, \mathbf{w}_k^d(N)]$ and $\mathbf{w}_k^d(j) = [w_k^d(1,j), \ldots, w_k^d(N,j)]^T$. Define the $N \times N$ matrix $\mathbf{V}_{k-d}(j) = [\mathbf{0}_N, \ldots, \mathbf{0}_N, \mathbf{v}_{k-d}, \mathbf{0}_N, \ldots, \mathbf{0}_N]^T$ where $\mathbf{v}_{k-d}$ is placed at the j-th column of the matrix and $\mathbf{0}_N$ denotes an N vector column made of 0 terms. Deriving (8), we obtain

$$\frac{\partial \mathbf{v}_k}{\partial \mathbf{w}_k^d(j)} = [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^0{}^T]^{-1} \mathbf{G'}_k \mathbf{V}_{k-d}(j) \tag{30}$$

Given (17), (22) and (30), we may write

$$\begin{aligned}
\Delta \mathbf{w}^d(j) &= \eta \sum_{k=d}^{k_f} \frac{\partial E}{\partial \mathbf{v}_k} [\mathbf{I} - \mathbf{G'}_k \mathbf{W}_k^0{}^T]^{-1} \mathbf{G'}_k \mathbf{V}_{k-d}(j) \\
&= \eta \sum_{k=d}^{k_f} g'_k(j) \mathbf{y}_k{}^T \mathbf{V}_{k-d}(j) \\
&= \eta \sum_{k=d}^{k_f} g'_k(j) y_k(j) \mathbf{v}_{k-d}^T && (31) \\
& && (32)
\end{aligned}$$

Finally, we obtain the vectorial delta rule for the DRNN

$$\Delta \mathbf{W}^d = \eta \sum_{k=d}^{k_f} \boldsymbol{\delta}_k \mathbf{v}_{k-d}^T \tag{33}$$

where the vector $\boldsymbol{\delta}_k$ is given by

$$\boldsymbol{\delta}_k = \mathbf{G'}_k \mathbf{y}_k \tag{34}$$

The procedure we arrive at includes as particular cases temporal-backpropagation for a feedforward FIR network (Wan 1993), recurrent-backpropagation for static recurrent networks (Almeida 1987, Pineda 1987), and backpropagation-through-time for recurrent networks (Rumelhart et al. 1986). It can be viewed as recurrent backpropagation applied to an *adjoint* (Srinivasan et al. 1994) or *transposed* network (Hertz et al. 1991). The adjoint is constructed using the following rules: 1) reverse all signal flows so that inputs are converted by outputs, 2) redefine nodes as summing junctions achieved by using the derivative of the nonlinear fixed point equation (8), and 3) reverse time such that iteration k is replaced by $k_f - k$ in all time-varying network variables, where k_f is the current iteration. A simple DRNN model with its adjoint is depicted in Fig. 2.

As discussed above, the backward algorithm makes necessary the artifact of unfolding the adjoint network in a reverse way in time. This is achieved by replicating the units and the connections backward through time until all delays have been removed. The method produces a *time*-feedforward

and *space*-recurrent constrained static structure where the time dependencies have been made external to the network itself. Consequently, the generalized ∂ rule (33) requires the storage of the unit internal inputs $\mathbf{h}_k$ and activations $\mathbf{v}_k$ while going forward in time. The backward phase is also backward in time and recursively uses these activations to compute the required gradients.
Nonetheless, for the unfolded version to be strictly equivalent to the original network, the manyfold duplication of the adjoint model should be repeated *ad infinitum*, that is to say, up to the first iteration. This is inhibitory from an implementation standpoint. We will see however that the cost due to the manyfold duplication of the hardware is maintained within acceptable limits, thanks to the nice properties of the gradient descent in this equivalent network.
According to Eq. 29, $\mathbf{y}_k$ can be found by iterating a linear backward difference equation. This requires however the inversion of the matrix $\mathbf{I} - \mathbf{W}_k^0\mathbf{G'}_k$ at each iteration k. To get around the matrix inversion, Eq. 29 can be solved by the relaxation of a dynamical adjoint network analogous to (6)

$$\mathbf{T}\frac{d\mathbf{y}_k(t)}{dt} = -\mathbf{y}_k(t) + \mathbf{e}_k(t) + \sum_{n=1}^{D} \mathbf{W}_{k+n}^n \mathbf{G'}_{k+n}\mathbf{y}_{k+n} \tag{35}$$

Non-local operations are not required anymore (such as matrix inversion) that go outside the original connectivity of the network. The fixed point is readily given by (29). In light of (35), training the DRNN boils down to training each individual space-recurrent network, which composes the layers of the unfolded adjoint network, by using the standard recurrent-backpropagation (Almeida 1987, Pineda 1987). Note that the stability of (35) follows from the stability of (8) as we assumed (Almeida 1987, Aussem 1995a).
The stable solution of Eq. 29 can be written in a more convenient vector form:

$$\left\{ \begin{array}{lcl} \mathbf{y}_k & = & [\mathbf{I} - \mathbf{W}_k^0\mathbf{G'}_k]^{-1}\tilde{\mathbf{e}}_k \\ \tilde{\mathbf{e}}_k & = & \mathbf{e}_k + \sum_{n=1}^{D} \mathbf{W}_{k+n}^d \mathbf{G'}_{k+n}\mathbf{y}_{k+n} \end{array} \right. \tag{36}$$

Eq. 36 is interesting in the sense that it expresses the actual term $\mathbf{y}_k$ as the sum of two terms: the first accounts for the present network error $\mathbf{e}_k$, while the second corresponds to an additional error term, which is conceptually the equivalent error transposed at the current time step k and due to the influence of the current state vector on the network future errors.
This is interesting in that it shows that $\mathbf{y}_k$ at time k is a linear combination of the current and the subsequent errors $\mathbf{e}_j$ for $j = k, \ldots, k_f$. So let $\mathbf{y}_k^{k+n}$ be the backpropagated error vector at time k when all $\mathbf{e}_{k+j} = 0$ except for $j = n$. Define $E_n = (\mathbf{d}_n - \mathbf{v}_n)^T(\mathbf{d}_n - \mathbf{v}_n)$ as the quadratic error at time n. We may write

$$\mathbf{y}_k^{k+n\,T} = \frac{\partial E_{k+n}}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k\mathbf{W}_k^{0\,T}]^{-1} \tag{37}$$

Given the linearity of E with respect to E_n, we may write

$$\mathbf{y}_k = \sum_{n=0}^{k_f - k} \mathbf{y}_k^{k+n} \tag{38}$$

Now define

$$\boldsymbol{\delta}_k^{k+n} = \mathbf{G'}_k\mathbf{y}_k^{k+n} \tag{39}$$

We may expand the delta-rule (33) by incorporating the terms $\mathbf{y}_k^{k+n}$ and $\boldsymbol{\delta}_k^{k+n}$

$$\begin{aligned}\Delta \mathbf{W}^d &= \eta \sum_{k=d}^{k_f} \boldsymbol{\delta}_k \mathbf{v}_{k-d}^T \\ &= \eta \sum_{k=d}^{k_f} \sum_{n=0}^{k_f-k} \mathbf{G'}_k \mathbf{y}_k^{k+n} \mathbf{v}_{k-d}^T \\ &= \eta \sum_{k=d}^{k_f} \sum_{n=0}^{k_f-k} \boldsymbol{\delta}_k^{k+n} \mathbf{v}_{k-d}^T \\ &= \eta \sum_{n=d}^{k_f} \sum_{k=d}^{n} \boldsymbol{\delta}_k^{n} \mathbf{v}_{k-d}^T \qquad (40)\end{aligned}$$

Therefore, at a given time n, the terms $\mathbf{y}_k^n$, $k < n$ are backpropagated through the adjoint network up to time 0. Each gradient term is recombined afterwards to store the overall weight update due to the error $\mathbf{e}_n$.
The network unfolding takes place at the end of each time step n in order to calculate the terms $\boldsymbol{\delta}_k^n$ for $k = 0, \ldots, n$. As discussed before, the storage and memory requirements grow proportionally with the pattern index. To prevent the systematic expansion of the error-propagation network, we investigate in the next section the dynamic of $\boldsymbol{\delta}_k^n$ with respect to k. This study is also of interest for the stability analysis of the weight updates.

4.1.1 Stability of the Weight Updates

Before we describe the complete algorithm, the *asymptotic stability* of the weight update rules when k_f becomes large must be addressed. Due to the dynamical behavior of the DRNN, the initial conditions of the system at time 0, namely $\mathbf{v}_0$ and $\mathcal{W}_0$, along with the sequence of input patterns $\mathbf{i}_k$ completely determine the trajectory of the system. Therefore any change in a weight value at some time will influence the subsequent behavior of the system. Typically, all the individual weight adjustment resulting from the duplication of the connections in the unfolded adjoint network must be recombined to compute the overall change. This is achieved by summing the individual gradient components over the overall unfolding interval, that is

$$\Delta \mathbf{W}^d = \sum_{k=d}^{k_f} \Delta \mathbf{W}_k^d \qquad (41)$$

The question we are now faced with is whether the sum is diverging or remains bounded as $k_f \to \infty$. Answering this question in the general framework is a formidable task since the weight evolution is strongly dependent on the dataset itself. In practice the weights may blow up, only to be limited by their saturation limits. In fact, it seems that no general answers can be formulated. But as we are focusing on a samplewise algorithm, what we are concerned with is not the *global* stability of the weight updates over the overall trajectory, but instead the *convergence* at *each* time step of the recursive backward propagation through the adjoint network. And the convergence is *a priori* not ensured insofar as the number of layers in the unfolded adjoint is increasing with k_f. So we limit ourselves to the issue of finding conditions assuring the *existence* of the weight update at each time step. For this purpose, let us introduce the following definition of stability

Definition 1 - *The weight update procedure in the DRNN is said to be stable when*

$$\lim_{n\to\infty} \|\sum_{k=d}^{n} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T\| < \infty \tag{42}$$

Not only shall $\boldsymbol{\delta}_{k-j}^n$ tend to zero with respect to j but the decay rate must be exhibited to conclude on the convergence of the above sum.
For this purpose, we introduce the following theorem

Theorem 1 - *Consider a discrete-time relaxation-wise dynamical recurrent neural network represented at each time step* k *by the fixed point recurrent equations* (8). *Let* N *and* D *be the number of units and the AR filter order,* $\mathbf{i}_k$ *the exogenous input vector,* $\mathbf{v}_k$ *the state vector, and* $\mathbf{d}_k$ *the target output vector* $(\mathbf{i}_k, \mathbf{v}_k, \mathbf{d}_k \in R^{[N\times 1]})$.
Let the instantaneous scalar cost be $E_n = \frac{1}{2}(\mathbf{d}_n - \mathbf{v}_n)^T(\mathbf{d}_n - \mathbf{v}_n)$ *and consider the error vector* ${\mathbf{y}_k^{k+n}}^T = \frac{\partial E_{k+n}}{\partial \mathbf{v}_k}[\mathbf{I} - \mathbf{G'}_k {\mathbf{W}_k^0}^T]^{-1}$, $\boldsymbol{W}_k^d$ *representing the weight matrix associated with the* d*-th delay connections, and* $\boldsymbol{G'}_k$ *being defined in* (20).
Let $k_f > 0$ *be any trajectory length of the system, if the following conditions*

$$\|\mathbf{W}_k^0\| < 1 \quad \text{and} \quad \frac{\|\mathbf{W}_k^d\|}{1 - \|\mathbf{W}_k^0\|} < \frac{1}{D} \tag{43}$$

hold $\forall d = 1, \ldots, D$, $\forall k = 0, \ldots, k_f$, ***then*** $\exists 0 < \rho < 1$ *and* $\exists M > 0$ *such that the inequality*

$$\|\mathbf{y}_{k_f-n}^{k_f}\| < M\rho^n, \tag{44}$$

holds for $n = 0, 1, \ldots, k_f$.

Proof: See Aussem (1995a, 1995b).

The theorem says, in essence, that the terms $\mathbf{y}_{k_f-n}^{k_f}$ are bounded by a series decaying at exponential rate with respect to n whenever conditions (43) are satisfied. While we omit the discussion of the details of this analysis, different possible trajectories can be expected for $\mathbf{y}_{k_f-n}^{k_f}$ (Aussem 1995a). From Theorem 1, we derive the following propositions.

Proposition 1 (43) *are sufficient conditions for the stability of the weight update procedure for the DRNN.*

Proof: Let n be the current iteration. Given (40) along with (20), we may write

$$\begin{aligned}
\sum_{k=d}^{n} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T &\leq \sum_{k=d}^{n} \|\mathbf{G'}_k\| \|\mathbf{y}_k^n\| \|\mathbf{v}_{k-d}\| \\
&\leq 2\beta \sum_{k=d}^{n} \|\mathbf{G}_k\| \|\mathbf{I} - \mathbf{G}_k\| \|\mathbf{y}_k^n\| \|\mathbf{v}_{k-d}\| \\
&\leq \sum_{k=d}^{n} \|\mathbf{y}_k^n\| \|\mathbf{v}_{k-d}\| \\
&\leq N \sum_{k=d}^{n} \|\mathbf{y}_k^n\|
\end{aligned} \tag{45}$$

where the last inequalities stem from $\beta = 1/2$ and $||\mathbf{v}_{k-d}||_p < N$ for $p = 1, 2, \infty$. In accordance with Theorem 1, we can find $\rho < 1$ and $M > 0$ such that

$$\sum_{k=d}^{n} ||\mathbf{y}_k^n|| \leq M \sum_{k=0}^{n} \rho^k \leq \frac{M}{1-\rho} \tag{46}$$

completing the proof.

In regard to the epochwise algorithm, note that conditions (43) reduce to

$$||\mathbf{W}^0|| < 1 \quad \text{and} \quad \frac{||\mathbf{W}^d||}{1 - ||\mathbf{W}^0||} < \frac{1}{D} \quad \forall d = 1, \ldots, D \tag{47}$$

as the weights remain fixed along the trajectory.

Typically, when the network is stable, the residual backpropagated errors drop off exponentially through the past time slices, rapidly becoming negligible. Numerical simulations conducted by the authors strongly suggest that in practice the gradient descent is stable for most configurations even if conditions (43) are not satisfied (Aussem 1995a, 1995b).

4.1.2 Unfolding Stopping Criteria

Though the overall exponential decay of $||\mathbf{y}_{k-n}^k||$ is established, Theorem 1 doesn't say anything about their exact trajectories. The convergence towards the origin is not necessarily monotonic, and may instead oscillate (Aussem 1995a, 1995b). Therefore, we propose below two useful propositions to define a stopping criterion in the adjoint network.

Proposition 2 *If conditions* (43) *are satisfied, then for all* ϵ*, if* $||\mathbf{y}_{k+j}^{k_f}|| \leq \epsilon$ *holds for* $j = 1, \ldots, D$*, then* $||\mathbf{y}_{k-n}^{k_f}|| \leq \epsilon$ *holds for* $n = 0, 1, \ldots, k$*.*

Proof: See Aussem (1995a, 1995b).

Proposition 3 *Consider the samplewise backward recurrent propagation algorithm. If the conditions* (47) *are fulfilled, then for the weight changes computed at each iteration to follow as closely as desired the true gradient, it is sufficient to backpropagate a finite number of time steps through the adjoint.*

Proof: Let the current iteration be n and assume we unfold the adjoint s retrograde time steps. The estimated weight matrix update is given by $\sum_{k=n-s}^{n} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T$ so that the resulting error is $\sum_{k=d}^{n-s-1} \boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T$. Now using Theorem 1 and (45), we may write

$$\sum_{k=d}^{n-s-1} ||\boldsymbol{\delta}_k^n \mathbf{v}_{k-d}^T|| \leq \frac{NM\rho^{s+1}}{1-\rho} \tag{48}$$

The term on the the left clearly tends to zero with respect to s, completing the proof.

Given the propositions above, we propose the following criterion to decide when to stop the unfolding process:

$$\frac{||\mathbf{y}_{k-s+j}^k||}{||\mathbf{y}_k^k||} \leq \epsilon \quad \text{for} \quad j = 1, \ldots, D \tag{49}$$

where ϵ is an arbitrary predefined value, equal to 10^{-3} in our implementation. Instead of inferring M and ρ from the decay of $||\boldsymbol{\delta}_k^n||$, (49) is an implementation-friendly criterion which avoids any computational burden. Note that s varies with k.

4.1.3 Temporal Recurrent Backpropagation

In light of the previous discussion, we describe an algorithm which takes advantage of the exponential decay of the $\boldsymbol{\delta}_k^n$ as $n \to \infty$. We call this algorithm working in incremental fashion the temporal recurrent backpropagation (TRBP) training algorithm.
The epochwise backsweep algorithm was derived on the assumption that the weights remained fixed along the trajectory. Since the calculation of the gradient at each iteration depends only upon the current and past states of the network, we may deviate from this assumption and adjust the weights while the network is running in order to avoid any array storage requirements depending on the current iteration value.
The snag is that the calculation of the true error derivative $\partial E/\partial w$ is not possible because of the weight adjustments at each iteration, with the result that the proper gradient can only be approximated. From proposition 3, we know however that a limited number of past states suffice to calculate what is required to follow as precisely as desired the true gradient in the epoch-wise scheme. Not to depart too much from the true gradient, we may choose η small enough so as to bound $||\Delta \mathbf{W}^d||$ over a trajectory of length Max(s) into decent limits. This is guaranteed to keep the gradient alteration, with respect to its true value, under control.
The on-line version of the epochwise backsweep algorithm is based on the concept of introducing artificial epochs into the system and then using the epochwise algorithm to update the weights. An artificial epoch is created by assuming that the current iteration is the final iteration $k = k_f$. The first iteration is not 0 anymore but instead an iteration $k - s$ determined from (49).
Therefore, the complete algorithm is implemented by performing the following at each time increment k:

- Relax the DRNN once until a fixed point solution is obtained, using the current input vector $\mathbf{i}_k$ and all past variables up to $k - D$ required by

$$\mathbf{v}_k = f(\mathbf{i}_k, \ldots, \mathbf{i}_{k-D}, \mathbf{v}_k, \mathbf{v}_{k-1}, \ldots \mathbf{v}_{k-D}, \boldsymbol{\mathcal{W}}_k) \tag{50}$$

- Calculate an approximation for $\mathbf{y}_{k-n}^k$, denoted by $\hat{\mathbf{y}}_{k-n}^k$, for $n > 0$ by unfolding the adjoint network backward in time and relaxing

$$\mathbf{T}\frac{d\hat{\mathbf{y}}_{k-n}^k(t)}{dt} = [\mathbf{W}_k^0 \mathbf{G'}_{k-n} - \mathbf{I}]\hat{\mathbf{y}}_{k-n}^k(t) + \sum_{n=1}^{D} \mathbf{W}_k^n \mathbf{G'}_{k-n+j}\hat{\mathbf{y}}_{k-n+j}^k \tag{51}$$

The boundary conditions of (51) take the form

$$\hat{\mathbf{y}}_k^k = \mathbf{y}_k^k = [\mathbf{I} - \mathbf{W}_k^0 \mathbf{G'}_k]^{-1}\mathbf{e}_k \tag{52}$$

and

$$\hat{\mathbf{y}}_{k+1}^k, \ldots, \hat{\mathbf{y}}_{k+D}^k = \mathbf{0} \tag{53}$$

Notice that $\mathbf{y}_{k-n}^k$ cannot be calculated precisely because of the weight changes at each iteration and the weight matrix $\mathbf{W}_{k-n+j}^j$ is approximated by $\mathbf{W}_k^j$, which is the current weight matrix. Obviously, the smaller the difference between the weights at iteration k and those at iteration $k - n + j$ the more accurate this approximation will be and thus the more accurate the calculation of the terms $\hat{\mathbf{y}}_{k-n}$.

Locally in the adjoint, the connections are reversed, that is, the coupling w_{ij}^d from i to j is replaced by $g'(h_j)w_{ji}^d$ from j to i. The network errors are now taken as inputs and the cells now perform a linear transformation of their input.

- Stop the backward recursion of (51) at iteration $k - s$ satisfying

$$\frac{||\mathbf{y}_{k-s+j}^k||}{||\mathbf{y}_k^k||} \leq \epsilon \quad \text{for} \quad j = 1, \ldots, D \tag{54}$$

  The terms $\mathbf{y}_k^{k-j}$ are stored at each backward iteration until the iterative process terminates.

- The overall weight variation - defined as the sum of individual updates - is then obtained by applying the generalized delta-rule

$$\Delta\mathbf{W}^d = \eta \sum_{n=d}^{k} \boldsymbol{\delta}_n^k \mathbf{v}_{n-d}{}^T \tag{55}$$

- Store for the next gradient propagation the terms $g(\mathbf{h}_{k-j})$ and $\mathbf{v}_{k-j}$ for $j = 0, 1, \ldots, L$ where L is taken large enough so satisfy $s < L$ at any time. In practice, L is first set to a multiple of D and incremented each time $s > L$ until a correct value is found.

The algorithm preserves the symmetry between the forward propagation of unit outputs and the backward propagation of error terms. This leads to an efficient parallel implementation requiring s relaxations of the transposed error network. Since s is of order D, storage of order DN is needed to keep track of the network previous states. The time required at each backward pass to compute the elementary weight update scales as $O(DN^2)$ so that $O(D^2N^2)$ operations are needed at each time step to adjust the network parameters. In contrast, a forward propagation algorithm entails storage capacity of $O(DN^3)$ and the computational requirements scale as $O(D^2N^4)$ (Aussem 1995a, 1995b). Moreover, a matrix inversion is required to compute the gradient resulting in that locality is lost. Under these circumstances, the computational and storage requirements clearly support use of the TRBP algorithm. We mention that BPTT (Rumelhart et al. 1986) and RTRL (Williams et al. 1989) can be merged together by decomposing the calculation of the gradient into blocks of size $O(N)$. For each block, several BPTT-like passes are performed plus one RTRL-like calculation for integrating the results of these BPTT-like passes into the results obtained from previous blocks (Schmidhuber 1992). Applying this to the DRNN, the method reduces the average time complexity per time step to $O(D^2N^3)$ instead of $O(D^2N^4)$ while still calculating the same gradient.
It is also worth remarking that the relaxation phase boils down to a single iteration of the network forward in time when no *space*-recurrent links are used. It is the case for any synchronous dynamics (i.e. $w_{ij}^d \neq 0$ for $d > 0$ only) including those proposed by Williams et al. (1989) and Wan (1993). Thus, the DRNN model belongs to a more general class/type of neural networks.
Given the low resource requirements along with the observation that the DRNN structure encompasses most of the recurrent networks proposed to date, one can conclude that TRBP is preferable for training any space- and time-recurrent networks *with* gradient descent. In view of its straightforward implementation, it is our hope that the algorithm will prove useful to those involved in neural-based temporal processing approaches.

5 Experimental Results

In this section, we report on some preliminary investigation on the performance of the DRNN model. We experiment with two real-word time series, namely the quarterly precipitation recorded in Namibia, and the astronomical seeing recorded at La Silla and at Paranal in Chile. These series, of interest at ESO, are typical cases where the initial attempt to understand the underlying features of the series failed either because of too small an amount of data, or because the data exhibited very erratic and confusing variations. Besides, the practical experience and theoretical knowledge of the meteorologists points to the inherent difficulty of deriving rules for local forecasting. Therefore prediction is sought without reference to any underlying physical model. Our main goals in the experiments described here were 1) to assess the predictability of these time series, especially the seeing, and 2) to appraise the DRNN models with standard neural techniques and a state space reconstruction approach.

5.1 Experimental Set-up and Performance Measure

Before we turn to the experiments, we detour to review well-known techniques we found to be useful for networks to run optimally.

In an attempt to optimize the learning rate η automatically, we allow the η to vary in the course of training since the best values at the outset of training may not be so good later on. Our approach is to check whether the weight updates at each iteration actually decrease the cost function. If not then the process is overshooting and η should be reduced. On the other hand, if several steps in a row have actually decreased the cost function, then η is increased. It appears best in the literature (Hertz et al. 1991) to increase η by a constant and to decrease it geometrically to allow rapid decay when necessary. This gives the overall scheme

$$\eta = \begin{cases} +\alpha & \text{if } \Delta E \leq 0 \\ -\beta\eta & \text{if } \Delta E > 0 \end{cases} \tag{56}$$

where α and β are appropriate time constants. In this paper, we took $\alpha = 10^{-3}$ and $\beta = 0.9$.

In order to preempt data overfitting, a topological reduction approach was adopted in these studies. The approach consists in pruning appropriately non-useful connections during training. This is achieved by giving each connection a tendency to decay to zero, such that

$$w_{ij}^{new} = (1-\epsilon)w_{ij}^{old} \tag{57}$$

so that the connections disappear unless reinforced (Hertz 1991, Weigend et al. 1990). For small w_{ij}'s to decay more rapidly than larger ones, we made ϵ dependent on w_{ij} by

$$\epsilon_{ij} = \frac{\gamma\eta}{(1+w_{ij}^2)^2} \tag{58}$$

where γ was varied in the course of training. We found out that to it is useful to begin with $\gamma = 10^{-3}$ until performance reaches a minimum and then to decrease the value up to 10^{-4} so the error continues to decrease at a slow rate.

Finally, a standard measure of fit in the literature is given by the *normalized mean squared error*:

$$\text{NMSE} = \frac{1}{\sigma^2 N}\sum_{k=1}^{N}(x(k)-\hat{x}(k))^2 \tag{59}$$

where $x(k)$ is the true value of the sequence, $\hat{x}(k)$ is the prediction, and σ^2 is the variance of the true sequence over the prediction duration N. A value of the NMSE $= 1$ thus corresponds to predicting the unconditional mean.

5.2 Precipitation Forecasts

In this section we report some preliminary simulation results using the NAR, NARMA, and DRNN predictive approaches described earlier. They are intended to show: 1) that dynamic and recurrent architectures behave differently than nonlinear AR and ARMA based on feedforward models, 2) to evaluate the predictability of a time series of interest at ESO with various connectionist techniques. We stress that the presented results are only indicative.
We have chosen to experiment on the particular series of precipitation measurement records around the site of Gamsberg in Namibia because the precipitation is an indicator of the astronomical site quality. Gamsberg mountain is at present a candidate site for the European telescopes. In the absence of meteorological stations around the site, a reliable anticipative model for the precipitation would permit to assess the predictability of the overall environmental conditions which are crucial for the determination of the observing site quality. Besides, this series is interesting in that it is periodic with amplitudes displaying *apparently* very irregular variations. Given this observation, along with the limited amount of data, these aspects afford an interesting opportunity to assess different neural-based methods.
Because of the differences in the architectures, it is quite difficult however to find a basis for comparison. Both the complexity of the architectures and the cost function have to be prescribed to allow proper assessments. In light of the small amount of data and the difficulty of the task, we are *forced* to compare the one-step ahead forecast of the predictive models on the training dataset. In fact, the prediction set was taken as the training set and a large weight decay rate was imposed on each network to avoid overfitting.
Not to favor any approach against the other, we chose to take approximatively the same number of adjustable parameters in each network at the outset of training. The length of the time-lagged vector for both the NAR and NARMA models was chosen to be 5 after some preliminary experiments on the data. 5 previous outputs of the NARMA network were fed back in addition to the external time-lagged vector. The number of adjustable coefficients was moderate for all networks, that is around 40 (a little less for the DRNN).
The three models used for assessment were:

- a $5 \times 5 \times 1$ NAR network.
- a $10 \times 5 \times 1$ NARMA network.
- a $1 \times 2 \times 1$ DRNN with $D = 3$. The model is depicted in Fig. 5.

For training purposes, the data was scaled to zero mean. Initial weight values of the networks were chosen randomly until the variance of the activation of the unit output was found to be non-negligible. For the DRNN, the weights afterwards scaled to satisfy conditions (43).
Table 1 summarizes various NMSE values for single-step predictions on the data set. The neural outputs are displayed in Fig. 3 for all these schemes. In summary, although we took extreme care not to gain any unfair advantage over feedforward-based approaches, the DRNN was found to perform significantly better within the constraints we imposed on the number of units. Due to the poor results achieved on multi-step prediction, the iterated performance could not be used to evaluate the candidate networks. In addition, the DRNN model collapsed to a very small network having 19 parameters while the others eliminated only one connection.
It was clear from the nature of the data that predicting the magnitudes in the oscillations would be the most important and difficult aspect of the series to be learned. Also instructive in this regard is the following. Though the closed-loop iterations of the NAR model produced damped oscillations rapidly decaying towards the origin, stable sine-like oscillations of constant amplitude were observed with the DRNN. The multi-step prediction error shown in Fig. 4 remains fluctuating around 0.45.

Very curiously, the average network output was about twice the data mean. We speculate that the model is more sensitive towards high peaks present in the data than low values.
A more detailed comparison of the performance of these architectures is being conducted at present, and will be reported elsewhere.

5.3 Fuzzy Nowcasts of Seeing

This experiment is in the context of an envisaged Astronomical Weather Station, a future interface between the observer and the terrestrial environment of the VLT observatory. Its function is to improve the quality of observations and to guarantee efficient use of the telecope time. If a reliable astronomical seeing estimate is proposed, some time in advance, then the ground-based telescope can be set in the most suited mode and allow an efficient use of the observing time.
Several experiments have been carried out so far (Aussem et al. 1994b, Murtagh et al. 1993), seeking to forecast quantitative-valued seeing measurements 24 hours in advance. However, non-linear multiple regression, including the k-nearest neighbor method and connectionist networks, were found to perform poorly on the seeing time series. In fact, seeing is not easy to predict; Jorgenson et al. (1991) discuss its *somewhat* chaotic behavior.
Therefore, a somewhat different tack is taken in this study with a view to improving our seeing prediction abilities. We split the problem into two parts: i) the prediction of meteorological conditions one day in advance, and ii) the *nowcasting* of seeing according to these predicted parameters. This paper focuses on the latter, namely seeing nowcasting. From the statistics viewpoint, nowcasting is multiple regression, i.e. we attempt to regress a set of meteorological variables on a seeing variable. We assume that good estimates can be provided by an *ad hoc* meteorological model.

5.3.1 Meteorological Variables Used

The meteorological variables used were: wind direction and velocity, at 10m and 2.5m above ground; seeing at 5m; temperature, humidity and pressure at 2m; and ground temperature at -0.1m. Very high correlation among some of these variables led us retaining the following: wind velocity in ms^{-1} at 10m above ground; standard deviation of the wind speed over 20 min. using 2-second samples; relative humidity, in percent; air temperature, in degrees at 2m above ground; pressure in mB; and seeing, measured at 5m above ground. Seeing is defined as the full width at half-maximum of a stellar image with a perfect large telescope at 0.5 μm wavelength and at zenith. Nightly median values are considered for these variables instead of hourly averages. This was done with the aim of giving an indication of the quality of the observing night. The data between spring 1989 and autumn 1991 were from Cerro Paranal. Construction then started at this site and the data for the four final seasons, summer 1992 to spring 1992 were collected from the nearby Cerro NTT. These data were collected and gathered to form a 605-tuples dataset.

5.3.2 Seasonally-Referenced Fuzzy Coding

A major lesson from time series analysis is that the *representation problem* plays an important role. A relevant representation can make information explicit and strip away obscuring clutter. Murtagh et al. (1993) adopted an intuitively plausible and interpretation-friendly form of coding. Each variable was categorized into three categories: low-valued ($= 0$), a linearly varying moderate-value, and high-valued ($= 1$). The boundaries between these categories was set at the 33rd and 67th percentiles. In a practical scenario, we believe it would be relatively easy to obtain high quality estimates of such variables all the more so as we require only minimal information, i.e. high, low, or moderate. The coding of seeing was referenced to the season. A season is a set of about 90 nights which are batched together in time. In fact, around 50 nights remained when

missing values were removed. The seasonal referencing which required a relevant set of 33rd and 67th percentiles was provided by the immediately preceding "season" of 49 cases. This is a robust approach: extreme values are not allowed to play a disproportionate role.
When we recode seeing in this way, we can refer to low values as good seeing, and high values as bad seeing, again referenced to the season or other subset of the data on which we base our definitions of the 33rd and 67th percentiles. The Euclidean distance may be more easily used between vectors which have been recoded in this way since normalization of the very differently-valued variables has been taken care of. We can go further, however, making use of the notions of "high" and "low" which are used in this recoding approach. In order to be consistent with these notions, we have taken them as fuzzy values. The usual definitions of relatedness and dependence generalized for use in such a fuzzy framework are described in what follows.

5.3.3 Definitions of Possibility

"High" and "low" and the linearly interpolated intermediate values are fuzzy quantities. When dealing with such data, we therefore use fuzzy generalizations of traditional notions such as conditional probability. The fuzzy version of this is termed conditional possibility.
Consider three variables, A_i, B_i, and C_i, where index i corresponds to a case or record. Therefore A_i could be the temperature measured at the time-point i; B_i could be the seeing measured at the same time-point; etc. As discussed in Murtagh et al. (1995) we have:

$$\text{Possibility of A and B} = \bigcap_i (A_i, B_i) = \sum_i \min(A_i, B_i)$$

$$\text{Conditional possibility of A, given B and C} = \bigcap_i (A_i, B_i, C_i) / \bigcap_i (B_i, C_i)$$

$$= \sum_i (\min_i(A_i, B_i, C_i)) / \sum_i (\min_i(B_i, C_i))$$

See Murtagh et al. (1995) for further discussion and references.

5.3.4 Fuzzy Correspondence Analysis

Correspondence analysis is a dimensionality-reduction method for the analysis and display of data. It is approximate insofar as we generally consider only two-dimensional projections of a multivariate space. In presenting projections in such a planar display, we may look for proximities (indicating high correlatedness, or perhaps even causal relationships), oppositions (for example, between good and bad seeing), and various other interrelationships.
Input to correspondence analysis was, in all cases, based on the fuzzily coded data, as described in the previous section. In addition, the input data was "doubled" in the following way: each variable was not only fuzzily coded, but its complement ($1-$ the fuzzy value) was also used. Thus, values for both low humidity and high humidity were entered for each case. Given the complementary between each such pair of values, there is considerably redundancy in the input data.
The reasons for using this form of input coding were two-fold. Firstly, we make reference to a technical aspect of correspondence analysis, discussed in Murtagh et al. (1993). Correspondence analysis uses the chi-squared metric, as opposed to the use of the Euclidean metric in principal components analysis. The Euclidean metric is often more convenient for visualization and interpretation. A side-effect of the "doubling" used in coding is that the total of values of all variables associated with each case is constant. When this happens, the chi-squared distance is reduced to a weighted Euclidean distance. Furthermore, each case is now identically weighted in the correspondence analysis, which would be naturally the case in a principal components analysis.

The second reason for using the "doubled" form of coding is that we are led to a particular form of interpretation. Having projections on the planar maps produced by correspondence analysis which are associated with good seeing and bad seeing is advantageous for our purposes. Having, additionally, projections associated with low and high humidity (for example) means that we will more easily draw conclusions with respect to such fuzzy variables. This "doubled" form of fuzzy coding is therefore helpful for interpretation purposes.

Complementary couples of variables (e.g. low humidity and high humidity) are symmetrically positioned vis-à-vis the origin in the space of full dimensionality. The origin itself represents an average profile, so that the nearer a complementary couple of variables are positioned to the origin, the less discriminating they are.

"High" and "low" were specified with respect to the 33rd and 67th percentile values within a season. Each case referred to a median of night-time recorded values, and was therefore associated with an observing night. The maximum of about 90 nights in each season was lessened because of missing data. The definition of conditional possibility of good seeing (let us call this A_i for case i) given low wind speed and low wind speed standard deviation (let us call these, respectively, B_i and C_i) has been discussed in Section 5.3.3. The "possibility of variables happening" is

$$\bigcap_i (A_i, B_i, C_i) \quad , \; 1 \leq i \leq n$$

where n is the number of cases.

Though we omit the details of the analysis, we can describe the overall conclusions. By seasonally-referencing our data, an appealing list of useful findings could be exhibited (Aussem et al. 1994a), indicating that a few variables often co-occur with good seeing. But there are nonetheless differences between the same season in different years, which limits the usefulness of these findings. Our problem is to make these relationships useful in a practical and consistent framework.

Within a season, we found thus that there is information available which would allow good forecasts. The most relevant meteorological variables are difficult to specify, though, because we find that they differ in different seasons.

5.3.5 Nearest Neighbor Nowcasts

The nearest neighbor approach is extended to deal with the fuzzy coding. Given a set of meteorological variables, we seek among all those stored in our database, the fuzzy seeing value associated with the nearest neighbor vector as the estimate of the desired value. In order to be consistent with the coding used, a fuzzy Jaccard measure (Miyamoto 1990, Murtagh et al. 1993, 1995) is used to find the best match. The Jaccard measure is well-known in information retrieval, and in some tests we carried out, it outperformed a fuzzy Euclidean distance. The definition, for vectors $\{A_i\}$ and $\{B_i\}$ is:

$$\bigcap_i (A_i, B_i) / \bigcup_i (A_i, B_i) = \sum_i \min(A_i, B_i) / \sum_i \max(A_i, B_i) \tag{60}$$

When equivalued similarities were found, we averaged the seeing values in an *ad-hoc* way, using the ordinary definition of mean. The experimental "leave one out" strategy consists in searching for the most similar set of meteorological variables in the entire database which also contains data collected at time beyond the case currently investigated. The seeing associated with the most similar set of meteorological variables provided the estimate. The bigger our database, the more chance we have of capturing every meteorological combination of interest.

The results obtained are presented in Table 2 and may be summarized as follows:

- Conditional probability of good and bad seeing, given that good seeing was estimated, is 48.2 % and 22.0 % respectively.
- Actual and estimated seeing in the same category: 40.2 % of cases.
- Actual and estimated seeing in the opposite category (good-bad or bad-good): 9.6 % of cases.

Here, one may remark that an uninformed policy based on the definition of good seeing would lead to 33% of cases.

5.3.6 A Cost-Adaptive Connectionist Approach

In this section, we report some preliminary simulation results obtained with a connectionist network. We note that an interest in this paper, beyond the quest for reliable forecasts, is in optimizing the scheduling observation time assignment procedure at ESO. A preliminary cost analysis has been carried out given that the queue policy amounts to selecting the observation that best fits the estimated seeing value, see Table 6. While the fuzzy nearest neighbor method provides unequivocal estimates of seeing, the neural net output interpretation is dependent on this particular performance objective.

Since the network output $o(t)$, at any time t, fell mostly inside the interval [0,1] , two thresholds (s_1, s_2) were required to interpret the output value. The prediction was assumed to be low whenever $o(t) < s_1$, a moderate value when $s_1 \leq o(t) \leq s_2$ and a high value when $o(t) > s_2$. These thresholds were optimally estimated from the performance achieved on the training set and then applied to the prediction set to appraise the predictive performance of the model. The thresholds could be adjusted on-line in a practical implementation as time progresses, among the chronologically most recent tuples. Consequently, the results presented are a lower bound of the expected performance. We emphasize that the choice of the cost function can be tailored in a convenient way to meet the changing requirements of the observatory engineers and astronomers.

With a view to presenting the performance of the connectionist model with the above objective in mind, three pairs of thresholds were used, related to the following performance measures (or policies) of interest:

- P1 - maximum hit rate, when actual and estimated seeing are in the same category.
- P2 - maximum conditional probability to have good seeing when good seeing was predicted.
- P3 - minimum cost function given by Table 6.

The results obtained on the prediction set are cross-tabulated in Tables 3 to 5. Various measures of interest can be derived from these contingency tables. They are described in the next section in a convenient way if one wishes to diminish the number of "false alarms", i.e. bad seeing when good seeing is expected, minimize a given cost function, etc.

5.3.7 Neural-Based Nowcast Results

In view of the autocorrelation function plotted in Fig. 6, short-term correlations are present in the coded seeing sequence despite a few sequence disruptions. We use a DRNN to capture the dynamical dependencies between local meteorological conditions and the fuzzily coded seeing. Given the rapid decay of the autocorrelation function, very moderate connection orders were used. The model depicted in Fig. 7 had 126 links including bias. Overfitting was prevented by the weight elimination method discussed in Section 5.1. In performing these simulations, we used a rigorous technique discussed in Aussem (1995a) to find an acceptable solution while substantially reducing the computational effort. In addition, similar investigations have been conducted previously with various non-optimized network configurations and are reported in Aussem et al. (1994a, 1994b). Prediction cumulative histograms are plotted in Fig. 8 and neural network outputs in Fig. 9.

- For policy P1, the conditional probability of good and bad seeing, given that good seeing was predicted, is 71.0 % and 7.9 % respectively. The possibility that good seeing is nowcast, that is, the number of good nowcasts / the total number of nowcasts, is 36.2 %.
- For policy P2, the conditional probability of good and bad seeing, given that good seeing was predicted, is 81.5 % and 3.7 % respectively. The possibility that good seeing is nowcast is 25.7 %.
- For policy P3, the conditional probability of good and bad seeing, given that good seeing was predicted, is 100 % and 0 % respectively. The possibility that good seeing is nowcast is 2.9 %.

These results clearly outperform all the previous simulations we have been carried out sor far with larger standard networks. Ultimately, Fig. 10 clearly illustrates the exponential decay of the terms y^n_{n-j} when backpropagated through the corresponding adjoint. This confirms the theoretical analysis carried out in this paper.
Because not all the details of our experiments are covered in this paper, a reader interested in further insights should refer to the references.

6 Conclusion

In this paper, we have shown that for modeling confusing time series with unknown temporal contingencies, the use of dynamical recurrent neural networks is useful and for some cases inevitable. This model encompasses most of the architectures proposed in the current literature. To calculate the proper "gradient amidst dynamic", it was illustrated that it is sufficient to backpropagate the errors a limited number of time steps through an adjoint model easily constructed from the original network. Though the resulting backward recurrent propagation network performs as many settling operations as the number of layers in the unfolded adjoint, the computational and storage complexity is shown to be very attractive. This stems from the exponential decay of the error terms when propagated backward in time.
The predictive ability of the DRNN model was illustrated with two time series. The model was first appraised with standard feedforward-based techniques on precipitation measurements recorded in Namibia. Then a flexible neuronal methodology is discussed for fuzzily coded astronomical seeing nowcasts. The framework pursued in the long term is to enable the astronomer and the observatory engineer to select appropriately the performance objective function most suited to their requirements. Remarkably reliable estimates were obtained by the model in regard to fuzzy astronomical seeing. Such an approach was found to outperform a k-nearest neighbors method.
The adequacy of the nowcasting approach for seeing prediction is confirmed and further studies conducted at present aim to obtain confident forecasts of these fuzzy seeing values one hour in advance. The assessment figures for nowcasting of seeing which are quoted in this article, and the connectionist model which has been prototyped, are a contribution to a future telescope decision support system.

References

L.B. Almeida 1987, "Backpropagation in perceptrons with feedback," in *Neural Computers*, eds. R. Eckmiller and Ch. von der Malsburg (Springer-Verlag, Neuss), pp. 199-208.

A. Aussem, F. Murtagh and M. Sarazin 1994a, "Fuzzy astronomical seeing nowcasts with a dynamical and recurrent connectionist network," *Neurocomputing*, accepted for publication.

A. Aussem, F. Murtagh and M. Sarazin 1994b, "Dynamical recurrent neural networks and pattern recognition methods for time series prediction: application to seeing and temperature forecasting in the context of ESO's VLT Astronomical Weather Station," *Vistas in Astronomy: Special Issue on Neural Networks* **38**, pp. 357-374.

A. Aussem 1995a, "Training dynamical recurrent neural networks with the temporal recurrent backpropagation algorithm: application to time series prediction and characterization," IEEE *Trans. on Neural Networks*, submitted.

A. Aussem 1995b, "Theory et applications of dynamical recurrent neural networks towards prediction, modeling and control of dynamical processes," Ph.D. Thesis (in french), Université René Descartes.

Y. Bengio, P. Simard and P. Frasconi 1994, "Learning long-term dependencies with gradient descent is difficult," IEEE *Trans. on Neural Networks* **5**, pp. 157-166.

G.E. Box and G.M. Jenkins 1976, *Time Series Analysis: Forecasting and Control* (Holden-Day, San Francisco).

J.T. Connor, R.D. Martin, an L.E. Atlas 1994, "Recurrent neural networks and robust time series prediction," IEEE *Trans. on Neural Networks* **5**, pp. 240-254.

G. Cybenko 1989, "Continuous value neural networks with two hidden layers are sufficient," *Math. Control Signals and Systems* **2**, pp. 303-314.

N.A. Gershenfeld and A.S. Weigend 1993, "The future of time series: learning and understanding, in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A.S. Weigend and N.A. Gershenfeld (SFI Studies in the Sciences of Complexity, Addison-Wesley), pp. 1-70.

J. Hertz, A. Krogh and R. Palmer 1991, "An Introduction to the Theory of Neural Computation," (Addison-Wesley, Redwood City, CA).

M. Jordan 1993, "Neural net architecture for temporal processing," preprint.

M.B. Jorgenson, G.J.M. Aitken and E.K. Hege 1991, "Evidence of a chaotic attractor in star-wander data," *Optics Letters* **16**, pp. 64-66.

K. Lang and G. Hinton 1988, "Time-Delay Neural Network architecture for speech recognition," Tech. rep. CMU-CS-88-152 (Carnegie-Mellon University, Pittsburgh).

A. Lapedes and R. Farber 1987, "Non linear signal prediction using neural networks: prediction and system modelling," preprint, (Los Alamos National Laboratory LA-UR-87-2662).

R. Leighton and B. Conrad 1991, "The autoregressive backpropagation algorithm," *Proceedings of the 1991 International Joint Conference on Neural Networks.*

S. Miyamoto 1990, "Fuzzy Sets in Information Retrieval and Cluster Analysis," (Dordrecht: Kluwer).

M.C. Mozer 1993, "Neural Net architectures for temporal sequence processing," in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A.S. Weigend and N.A. Gershenfeld (SFI Studies in the Sciences of Complexity, Addison-Wesley), pp. 243-265.

F. Murtagh, M. Sarazin and H.M. Adorf 1992, "Statistical prediction of astronomical seeing and of telescope thermal environment," in *ESO Conference on Progress in Telescope and Instrumentation Technologies*, ed. M.H. Ulrich (Garching, Germany), pp. 377.

F. Murtagh, M. Sarazin 1993, "Nowcasting astronomical seeing: a study of ESO La Silla and Paranal," *Publications of the Astronomical Society of the Pacific* **105**, pp. 932-939.

F. Murtagh, A. Aussem and M. Sarazin 1995, "Nowcasting astronomical seeing: towards an operational approach," *Publications of the Astronomical Society of the Pacific*, accepted for publication.

K.S. Narendra and K. Parthasarathy 1991, "Gradient methods for the optimization of linear control systems," IEEE *Trans. on Neural Networks*, pp. 252-262.

E. Ott 1993, Chaos in Dynamical Systems, Cambridge University Press.

S.W. Piché 1994, "Steepest descent algorithms for neural network controllers and filters," IEEE *Trans. on Neural Networks* **5**, pp. 198-212.

F.J. Pineda 1987, "Generalization of back-propagation to recurrent neural networks," *Physical Review Letters* **59**, pp. 2229-2232.

W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling 1988, Numerical Recipes in C, Cambridge University Press.

D.E. Rumelhart, G.E. Hinton and R.J. Williams 1986, "Learning internal representations by error propagation," in *Parallel Distributed Processing Explorations in the Microstructure of Cognition* **1**, eds. D.E. Rumelhart and J.L. McClelland (Cambridge, MA: MIT Press/Bradfords Books), pp. 318-362.

P.S. Sastry, G. Santharam and K.P. Unnikrishnan 1994, "Memory neuron networks for identification and control of dynamical systems," IEEE *Trans. on Neural Networks* **5**, pp. 306-319.

J. Schmidhuber 1992, "A fixed size storage O(n^3) time complexity learning algorithm for fully recurrent continually running networks," *Neural Computation* **4**, pp. 243-248.

B. Srinivasan, U.R. Prasad and N.J. Rao 1994, "Back propagation through adjoints for the identification of nonlinear dynamic systems using recurrent neural networks," IEEE *Trans. on Neural Networks* **5**, pp. 213-228.

C.T. Tsoi and A.D. Back 1994, "Locally recurrent globally feedforward networks: a critical review of architectures," IEEE *Trans. on Neural Networks* **5**, pp. 229-239.

E.A. Wan 1993, "Time series prediction by using a connectionist network with delay lines, in *Time Series Prediction: Forecasting the Future and Understanding the Past*, eds. A.S. Weigend and N.A. Gershenfeld (SFI Studies in the Sciences of Complexity, Addison-Wesley), pp. 195-207.

A.S. Weigend, D.E. Rumelhart and B.A. Huberman 1990, "Predicting the future: a connectionist approach," *International Journal of Neural Systems* **1**, pp. 195-220.

P. Werbos 1974, "Beyond regression: new tools for prediction and analysis in behavioral sciences", PhD thesis, Harvard University.

R.J. Williams and D. Zipser 1989, "Experimental analysis of the real-time recurrent learning algorithm," *Connection Science 1*, pp. 87-111.

Model	AR	NARMA	DRNN
Initial degrees of freedom	36	41	31
Effective degrees of freedom	35	40	19
NMSE	0.467	0.389	0.295

Table 1: Results for single-step predictions for AR, ARMA and DRNN models. A weight-elimination method was adopted to reduce the number degrees of freedom.

estimated \ actual seeing	good	moderate	bad
good	68	42	31
moderate	109	117	104
bad	27	49	58

Table 2: Contingency table exemplifying estimated versus actual seeing values with the k-nearest neighbor method in the overall data set - Same category: 40.2 %, Partially wrong: 50.2 % and Incorrect: 9.6 %

estimated \ actual seeing	good	moderate	bad
good	27	9	2
moderate	11	8	11
bad	1	16	25

Table 3: Contingency table exemplifying estimated versus actual seeing values in prediction set - Output thresholds have been optimally determined from the training set to maximize the hit rate (P1) - Same category: 57.1 %, Partially wrong: 40.0 % and Incorrect: 2.8 %

estimated \ actual seeing	good	moderate	bad
good	22	4	1
moderate	12	13	12
bad	1	15	25

Table 4: Contingency table exemplifying estimated versus actual seeing values in prediction set - Output thresholds have been optimally determined to maximize the good-good conditional probability (P2) - Same category: 57.1%, Partially wrong: 41.0% and Incorrect: 1.9 %

estimated \ actual seeing	good	moderate	bad
good	3	0	0
moderate	31	24	20
bad	0	9	18

Table 5: Contingency table exemplifying estimated versus actual seeing values in prediction set - Output thresholds have been optimally determined to minimize the cost function (P3) - Same category: 42.9%, Partially wrong: 57.1% and Incorrect: 0%

estimated \ actual seeing	good	moderate	bad
good	-1	+1	+1
moderate	+1/2	0	+1/4
bad	+1/2	+1/4	-1/4

Table 6: Costs associated with each possibility. Positive and negative values express failure and success respectively.

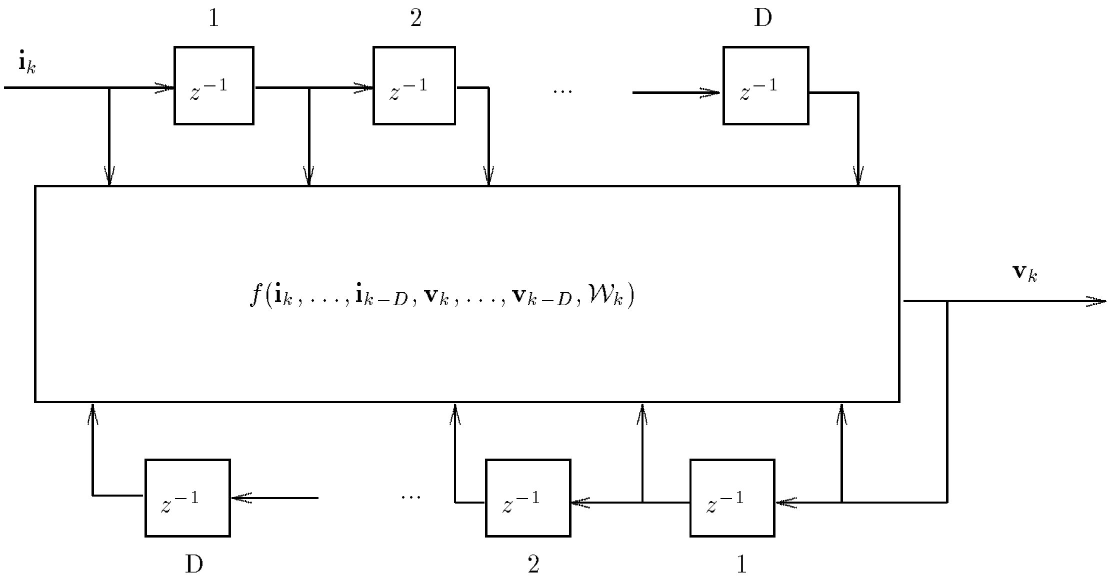

Figure 1: Standard representation of the dynamical recurrent network as a relaxationwise output error model.

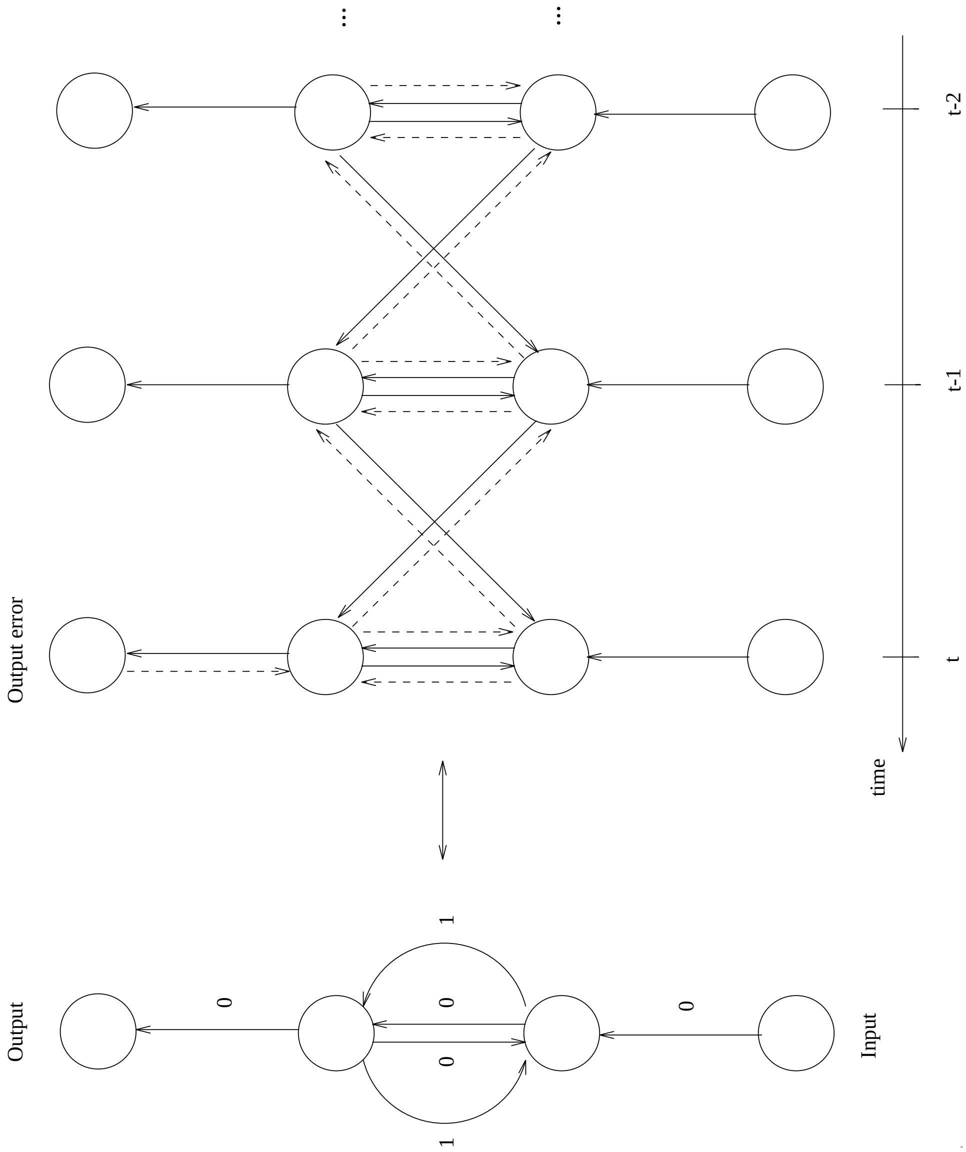

Figure 2: Illustration of a simple DRNN model with its unfolded counterpart.

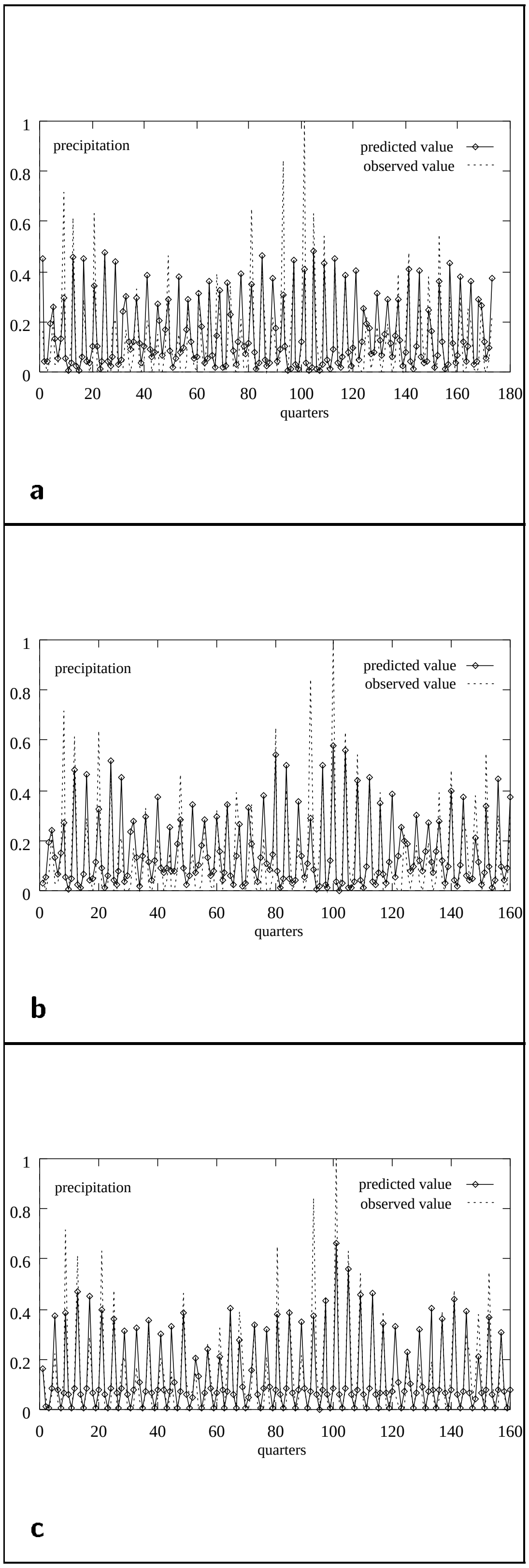

Figure 3: Single-step prediction for the precipitation series as plotted against quarters from 1949 to 1989 with (a) NAR model fed with a time lagged vector of length 5, (b) NARMA model fed with two time lagged vectors of length 5: the past observations and the past network outputs, (c) a DRNN model depicted in Fig. 5.

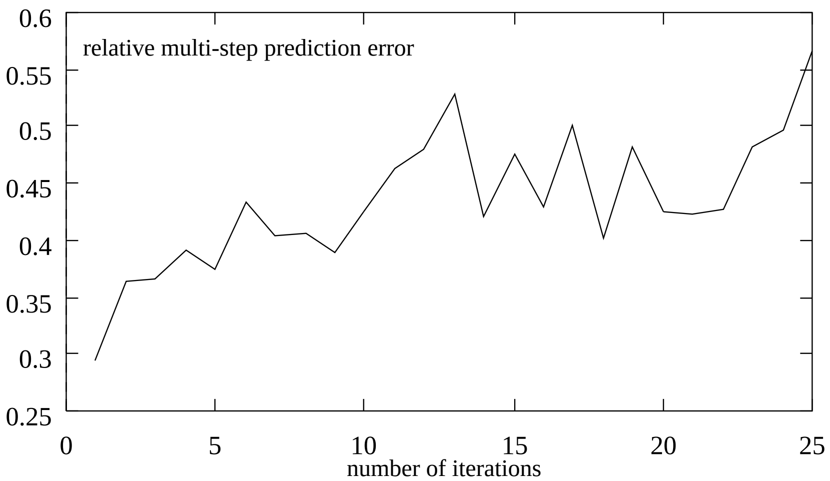

Figure 4: Precipitation multi-step prediction error averaged over the dataset as a function of the quarters into the future.

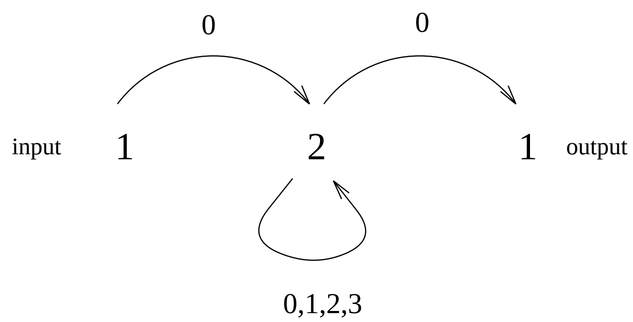

Figure 5: Schematic representation of DRNN model used in the precipitation experiment. The arrows represent a full connectivity between the layers and the associated figures indicate the connection delays. The first layer has 1 input unit; the second layer has 2 hidden units; and the output layer has 1 unit.

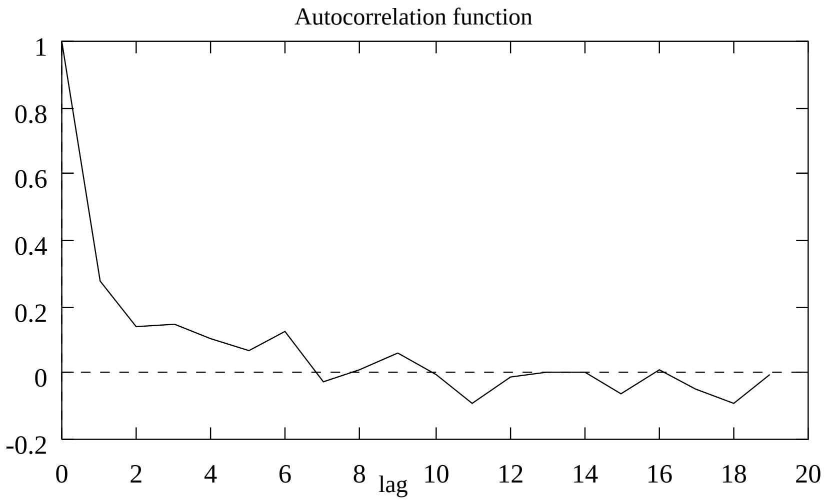

Figure 6: Autocorrelation function of fuzzy seeing sequence.

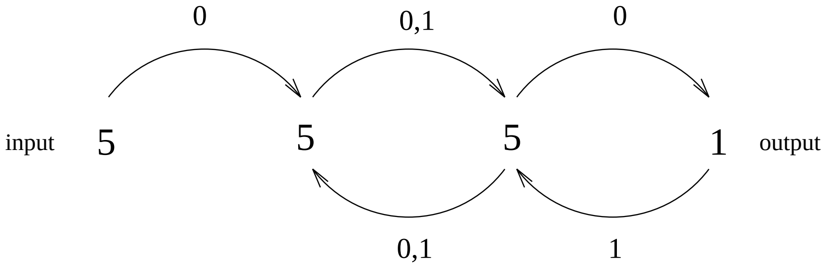

Figure 7: Schematic representation of DRNN model used in the seeing experiment. The arrows represent a full connectivity between the layers and the associated figures indicate the connection delays. The first layer has 5 input units (the 5 meteorological variables); the second and third layers have 5 hidden units each; and the output layer has 1 unit (the predicted seeing).

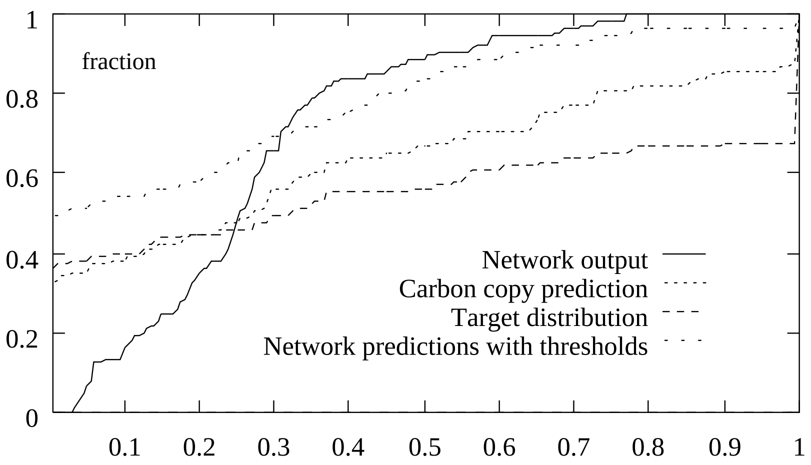

Figure 8: Fuzzily coded seeing cumulative histograms during winter and spring 1992. The plain line corresponds to the raw output of a dynamical recurrent neural network. The dotted line corresponds to the output decoded by mean of two thresholds optimally determined to maximize the hit rate.

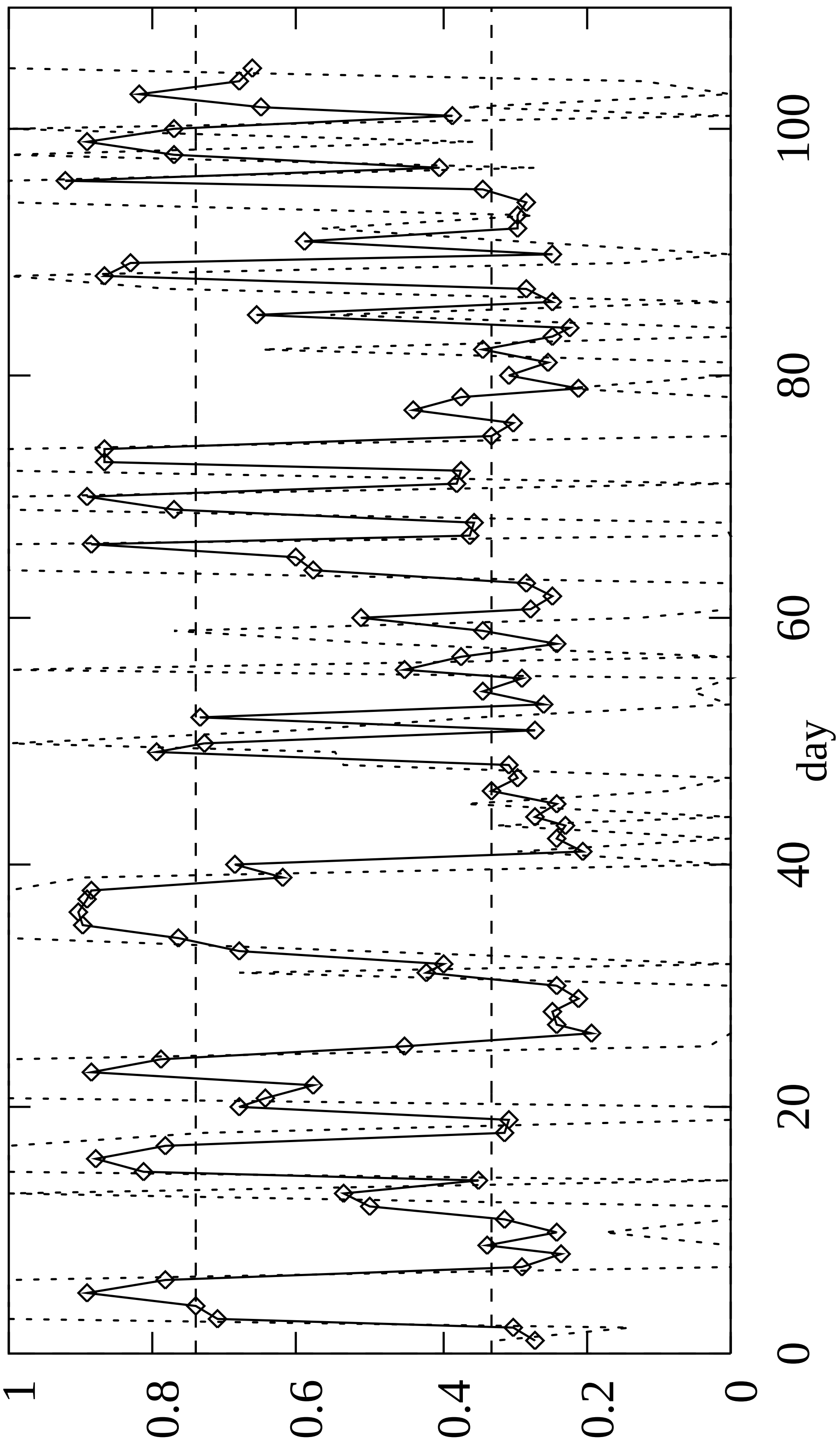

Figure 9: Fuzzy coded seeing (dotted line) during winter and spring 1992 and single-step forecasts (plain line) in the prediction set. Thresholds are shown in dotted line.

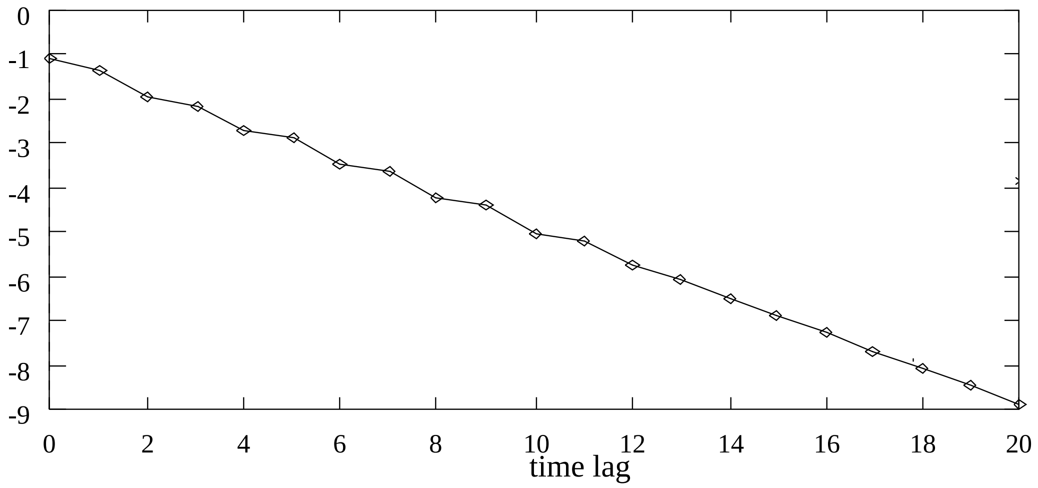

Figure 10: $< \mathrm{Log}_{10}(\|\mathrm{y}^n_{n-j}\|_\infty) >$ is plotted against j, the retrograde time step in the adjoint network. The DRNN used was a four layer $5 \times 5 \times 5 \times 1$ network. The model is depicted in Fig. 7.

List of figure captions

Fig. 1: Standard representation of the dynamical recurrent network as a relaxationwise output error model.

Fig. 2: Illustration of a simple DRNN model with its unfolded counterpart.

Fig. 3: Single-step prediction for the precipitation series as plotted against quarters from 1949 to 1989 with (a) NAR model fed with a time lagged vector of length 5, (b) NARMA model fed with two time lagged vectors of length 5: the past observations and the past network outputs, (c) a DRNN model depicted in Fig. 5.

Fig. 4: Precipitation multi-step prediction error averaged over the dataset as a function of the quarters into the future.

Fig. 5: Schematic representation of DRNN model used in the precipitation experiment. The arrows represent a full connectivity between the layers and the associated figures indicate the connection delays. The first layer has 1 input unit; the second layer has 2 hidden units; and the output layer has 1 unit.

Fig. 6: Autocorrelation function of fuzzy seeing sequence.

Fig. 7: Schematic representation of DRNN model used in the seeing experiment. The arrows represent a full connectivity between the layers and the associated figures indicate the connection delays. The first layer has 5 input units (the 5 meteorological variables); the second and third layers have 5 hidden units each; and the output layer has 1 unit (the predicted seeing).

Fig. 8: Fuzzily coded seeing cumulative histograms during winter and spring 1992. The plain line corresponds to the raw output of a dynamical recurrent neural network. The dotted line corresponds to the output decoded by mean of two thresholds optimally determined to maximize the hit rate.

Fig. 9: Fuzzy coded seeing (dotted line) during winter and spring 1992 and single-step forecasts (plain line) in the prediction set. Thresholds are shown in dotted line.

Fig. 10: $< \mathrm{Log}_{10}(\|\mathrm{y}^{n}_{n-j}\|_{\infty}) >$ is plotted against j, the retrograde time step in the adjoint network. The DRNN used was a four layer $5 \times 5 \times 5 \times 1$ network. The model is depicted in Fig. 7.