RANDOM PROJECTIONS: A REMEDY FOR OVERFITTING ISSUES IN TIME SERIES PREDICTION WITH ECHO STATE NETWORKS

Lucie Daubigney[‡*] *Matthieu Geist*[‡] *Olivier Pietquin*[‡†]

[‡] MaLIS - Supélec (Metz, France)
[†] UMI 2958 - GeorgiaTech/CNRS
[*] Team project Maia - Loria (Nancy, France)

ABSTRACT

Modelling time series is quite a difficult task. The last recent years, reservoir computing approaches have been proven very efficient for such problems. Indeed, thanks to recurrence in the connections between neurons, this approach is a powerful tool to catch and model time dependencies between samples. Yet, the prediction quality often depends on the trade-off between the number of neurons in the reservoir and the amount of training data. Supposedly, the larger the number of neurons, the richer the reservoir of dynamics. However, the risk of overfitting problem appears. Conversely, the lower the number of neurons is, the lower the risk of overfitting problem is but also the poorer the reservoir of dynamics is. We consider here the combination of an echo state network with a projection method to benefit from the advantages of the reservoir computing approach without needing to pay attention to overfitting problems due to a lack of training data.

Index Terms— Time series, echo state network, random projection

1. INTRODUCTION

Physical processes (such as the evolution of the temperatures in a place or the concentration of a gas in the atmosphere), economical processes, *etc.*, are phenomenons which can be modelled as time series. A reasonable question which may arise once the data are collected could be: would it be possible to predict future samples of a time series given those we already know? This is particulary true when important problems, like climate change, are at stake.

A first solution would be to start from a known model (*e.g.*, a differential equation) and adapt its parameters from the collected samples. However, determining a model is not an easy task when little or even nothing is known about the underlying process. Moreover, this way of solving the prediction problem usually provides only with *ad hoc* solutions and the crafting task, which can be time consuming, must be done for each new problem.

Soft computing approaches have thus been proposed to find models able to predict time series. The idea is to propose general models for which no hypotheses are required either about the underlying process or about the time dependencies between samples. The reservoir computing approach seems to be particularly appropriate. Indeed, reservoir computing uses *Recurrent Neural Networks* (RNNs) [1, 2] which can catch dependencies over time because of the recurrent connections among its neurons.

Results have been computed with the InterCell cluster funded by the Région Lorraine. The authors want to thank the ANR project METHODEO for partial funding.

An RNN is made of an input layer, a hidden layer (or reservoir) and an output layer. The hidden layer is supposed to contain a large number of neurons. Each neuron is possibly connected to itself and to any of the other neurons on this layer. Its connection to itself induces a memory effect. This layer is thus responsible for catching the dynamics. Supposedly, the larger the number of neurons in the reservoir, the more complex the caught dynamics is. More details can be found in [1]. The problem while using RNNs is that classic training methods such as backpropagation do not work properly [3].

Because of that, *Echo State Networks* (ESNs) have been introduced [4]. ESNs are RNNs where both the weights of the connections from the input to the reservoir and the weights of the connections inside the reservoir are set up randomly initially. Some aspects for defining good weights remain unclear but good heuristics have been found to build efficient ESNs [5]. The problem of learning the connections inside the reservoir is thus avoided and only the weights of the output layer are learnt. To do so, several algorithms have been proposed, such as a simple linear regression or most commonly least-squares [5]. Especially, an online version of this algorithm, the *Recursive Least Squares* (RLS) has been successfully experienced in [6] on a time series prediction task.

Related to the training of the ESN, the choice of the number of hidden neurons is of major importance when designing the network. One must remind that there are as many weights to learn as the number of neurons in the reservoir. As said previously, a large number increases the ability to represent a complex dynamics. However, if this number is too large (with respect to the amount of training data), the generalisation of the representation is not ensured (this is commonly known as the bias-variance trade-off). In other words, overfitting problems might appear.

The overfiting problem is underlined in [5]. In this tutorial, the authors propose to reduce the size of the reservoir. Yet, this may lead to poor results if the dynamics is complex. Other solutions like regularisation have also been proposed. In [7], a ridge regression is used to solve a time series prediction task. Also, results with several schemes of pruning and regularisation have also been published in [8] and tested on two different problems. Notably, the *Least Angle Regression* (LAR) algorithm [9], which performs an L_1 regularisation, was tested (this approach performs somehow feature selection).

Yet, since the underlying dynamics is rarely known, a large number of neurons in the reservoir is preferable by default. Designing an approach where this number could be large is thus of interest. The work presented here proposes to increase the robustness to overfitting when adding neurons in the reservoir. To do so, a projection of the internal state of the network (the activity of the hidden neurons) on a space of a smaller dimension is added. The projection used is a random projection [10] and the data from the original space are

projected on a random hyperplane. Under particular conditions, the distance between the points from the original space to the new ones are preserved with a given probability. The advantage of such a projection is that the dimension of the new space does not rely on the dimension of the initial space but only on the amount of data.

This new method has the advantage of decreasing the number of parameters to learn while keeping the number of hidden neurons large. A complex dynamics can thus be caught and the training can be made with a reasonnable amount of data. In what follows, a brief review of ESNs and random projections is proposed in Sec. 2. Then, in Sec. 3, the experimental settings are presented. Finally, in Sec. 4, the combination of ESNs and random projections is tested on a time series prediction task. The results are compared to the ones returned by an ESN trained with RLS and LAR algorithms.

2. ESN COMBINED WITH RANDOM PROJECTIONS

2.1. Principle of ESNs

Echo State Networks (ESNs) [4] are Recurrent Neural Networks (RNNs) where only the weights to the output layer are learnt. The other connections are chosen randomly in order for the network to have the *echo state property*. This property states that if the internal state of the network is equal to another one after the network has been fed with two sequences of inputs, then the input sequences are identical. The reservoir of the ESN must be rich enough to match each history of inputs to a different internal state. The ESNs are very interesting networks since they avoid the difficult task of learning the recurrent connections.

Let $\mathbf{u}(k) \in \mathbb{R}^{N_i}$ be a column vector representing the input at time k (N_i being the number of input neurons), $\mathbf{x}(k) \in \mathbb{R}^{N_h}$ be a column vector representing the activity of the hidden neurons (N_h being the number of hidden neurons), and $\mathbf{y}(k) \in \mathbb{R}^{N_o}$ be a column vector representing the activity of the output (N_o being the number of output neurons). Let W^{in} be a $(N_h \times N_i)$-matrix containing the synaptic weights of the connections between the input neurons and the hidden layer, W^{hid} a $(N_h \times N_h)$-matrix containing the weights of the connections in the hidden layer and W^{out} a $(N_o \times N_h)$-matrix containing the weights of the connections from the hidden layer to the output. The activation of the hidden neurons and the output neurons is updated as follows:

$$\begin{aligned} \mathbf{x}(k+1) &= f(W^{hid}\mathbf{x}(k) + W^{in}\mathbf{u}(k+1)) \\ \mathbf{y}(k+1) &= f^{out}(W^{out}(\mathbf{u}(k+1), \mathbf{x}(k+1))) \end{aligned}$$

with f the activation function of the hidden layer (for example $f = tanh$) and f^{out} the activation function of the output layer (for example f^{out} being the identity). The vector $(\mathbf{u}(k+1), \mathbf{x}(k+1))^T$ is the concatenation of the vector $\mathbf{u}(k+1)$ and $\mathbf{x}(k+1)$. The input matrix W^{in} is a possibly sparse random matrix and W^{hid} is a sparse matrix. No precise conditions have been determined to choose the coefficients of the matrix W^{hid} but for the choice of the spectral radius α of W^{hid}. The spectral radius is defined as $\alpha = \max_i |\lambda_i|$, λ_i being the eigenvalues of W^{hid}. A spectral radius greater than one will just induce the echo state property not to be ensured [4]. In practice, a network which W^{hid} matrix has a spectral radius smaller than one seems to often exhibit the echo state property. In Fig. 1 the description of the ESN is summarised.

The main advantage of the ESN approach is that most of the synaptic weights are chosen randomly. Only the output weights are learnt. To do so, a linear regression can be used to learn the W^{out} matrix. A sequence of $n \in \mathbb{N}$ pairs $(\mathbf{u}_{teach}(k), \mathbf{y}_{teach}(k))$ is provided to the network. The training error to be minimised is $\epsilon_{train}(k) = \mathbf{y}_{teach}(k) - W^{out}(\mathbf{u}_{teach}(k), \mathbf{x}(k))$.

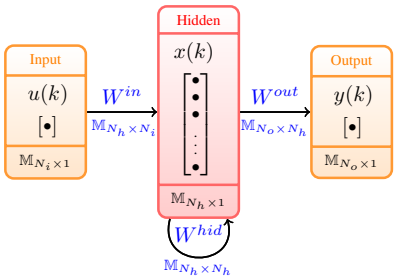

Fig. 1. Structure of an ESN. For the example, $N_i = 1$ and $N_o = 1$.

Here are the steps of the ESN training. First, the input matrix W^{in} and the matrix of hidden connections W^{hid} are randomly built. Then, for all of the examples of the training set, $S = (\mathbf{u}_{teach}(k), \mathbf{y}_{teach}(k))$, $k \in 0..n$, the network is fed with $\mathbf{u}_{teach}(k)$ and the output $\mathbf{y}(k) = W^{out}(n-1)\mathbf{x}(k)$ is computed. Each time an example is provided, the weights are updated following a *Widrow-Hoff* rule:

$$W^{out}(k) = W^{out}(k-1) + K_k(\mathbf{y}_{teach}(k) - \mathbf{y}(k)).$$

The gain K_k is computed so that the empirical cost $\sum_{i=0}^{k} \epsilon_{train}(i)^2$ is minimised. The algorithm used here is the *Recursive Least Squares* (RLS) algorithm.

Since finding a good compromise between the complexity of the model (which memory depth is rarely known) and the amount of training data is not so obvious, it can be of interest to find a new approach to get rid of this constraint. This approach would present the advantage of allowing a large number of neurons in the reservoir without paying attention to overfitting problems. The idea developed here is to project the $\mathbf{x}$ vector into a space of smaller dimension.

2.2. Random projections

As presented in the previous section, choosing the number of hidden neurons may be difficult in situations where neither the amount of training data is big nor a good intuition about the complexity of the dynamics of the system is known. The method proposed here allows the number of neurons to be increased while avoiding overfitting.

The method is a projection into a space of a smaller dimension with some guarantees about the loss of information. The method is called random projection since the coefficients of the projection matrix are chosen randomly (under certain conditions). The weights of the output matrix W^{out} are not learnt direclty from the $\mathbf{x}(\mathbf{k})$ vector anymore but with the projection of the $\mathbf{x}(\mathbf{k})$ vector on this basis.

The results about the random projections are based on a theorem by Dimitri Achiloptas [10] based itself on a lemma by Johnson and Lindenstrauss [11]. The theorem states that it is possible to find a projection with a controlled loss of information.

Theorem *Let M be an arbitrary set of n points in $\mathbb{R}^m$ represented in a $n \times m$ matrix A. Given $\epsilon, \beta > 0$, let $K_{\epsilon,\beta}$ be a constant and $m_0 = K_{\epsilon,\beta} log(n)$.*

For integer $p > p_0$, let R be a $m \times p$ random matrix with $R(i,j) = r_{ij}$, where r_{ij} are independant random variables: $r_{ij} = +\sqrt{3}$ with probability (w. p.) $1/6$, $-\sqrt{3}$ w. p. $1/6$, 0 w. p. $2/3$.

Let $\Pi = \frac{1}{\sqrt{p}}AR$ and let $f : \mathbb{R}^m \rightarrow \mathbb{R}^p$ map the i^{th} row of A to the i^{th} row of Π. W. p. at least $1 - p^{-\beta}$, for all $u, v \in M$:

$$(1-\epsilon)||u-v||^2 \quad \leq \quad ||f(u) - f(v)||^2 \leq (1+\epsilon)||u-v||^2$$

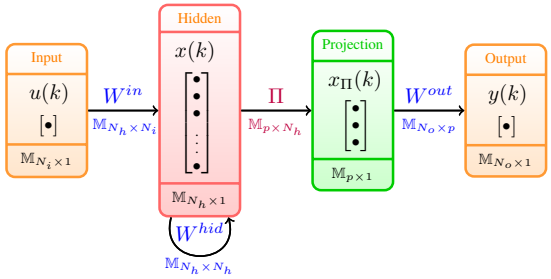

Fig. 2. Combination of the ESN and random projections. With $\mathbf{x}_\Pi(k) = \Pi\mathbf{x}(k)$. For this example, $N_i = 1$, $m = 3$, and $N_o = 1$.

This result states that finding a projection from the initial space to a space with a smaller dimension while losing a bounded amount of information is possible. The information loss depends on the dimension of the projection which itself depends on the amount of data n ($log(n)$ dependency). Such a projection is interesting since its dimension does not depend on the dimension of the initial space.

By means of this theorem, a large number of neurons can be used in the hidden layer. The $\mathbf{x}$ vector is then projected into a space of smaller dimension. Eventually, the coefficients of the model are learnt from the projected data. The computational complexity thus decreases and the risk of overfitting decreases too since the number of parameters, for a same amount of training data, is lower. Fig. 2 sums up how ESNs and random projections are combined.

ESNs combined with RPs are tested in the next section for a time series prediction task, a standard benchmark where ESNs have already been proven efficient [12, 6].

3. EXPERIMENTAL SETTINGS

The time series used for the experiments is described first. ESNs parameters and the metrics for the evaluation of the results follow.

3.1. Description of the time series

A 10^{th} order non-linear NARMA time series is used for the experiments. This series is the same as the one described in [6] and was the first time used in [13]. The system is driven from an independent and identically distributed uniform input e from $[0\,,\,0.5]$:

$$d(k+1) = tanh\left(0.3d(k) + 0.05d(k)\left[\sum_{i=0}^{9} d(k-i)\right] + 1.5e(k-9)e(k) + 0.1\right).$$

To run the experiment, input sequences $(e_{teach}(k), d_{teach}(k))$ have been generated. The first 200 samples have been discarded to let the time series stabilises. An overview of the series and an example of prediction returned by the ESN is given in Fig. 3. Although the series is noisy, the prediction follows the actual series even if not perfectly.

3.2. Description of the ESN

The parameters of the ESN are the following: $N_i = 1$ and $N_o = 1$ (since it is a 1-dimension time series), N_h varies. The input unit u is linked to the reservoir with a random vector W^{in} sampled from a uniform distribution in $[-0.1\,,\,0.1]$. The reservoir matrix W^{hid} has a 5% connectivity and α set to 0.8. The number of hidden neurons varies during the experiments to check the influence of the reservoir

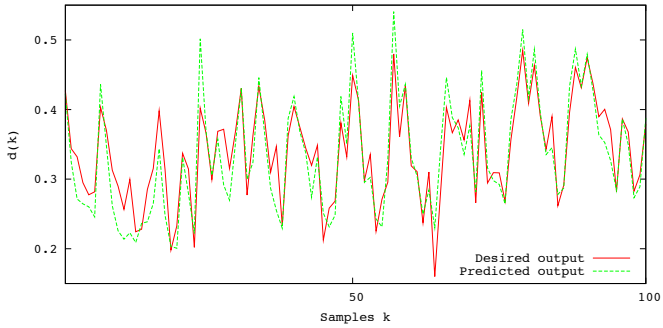

Fig. 3. Overview of the time series and an example of prediction returned by the ESN.

size on the quality of the prediction. The weights between ouput unit y and the reservoir are learnt. When the ESN is fed with new samples, the first 500 data are discarded to let the reservoir of the ESN reach the steady state.

3.3. Evaluation of the performance

The Mean Squared Error (MSE) between the testing data y_{test} and the prediction $\hat{y}$ of the network over several trials serves as a performance metrics:

$$MSE = \frac{1}{T}\sum_{i=1}^{T}(y_{test}(i) - \hat{y}(i))^2.$$

The amount of training data n is set to 100. The amount of testing data T is set to 100. Each of the curves presented is an average over the MSE computed for 50 different training sets and different ESNs (for each new training set, new weights for W^{in} and W^{hid} are generated). The associated variance also appears on the graphs. Both the abscissia and ordinates are in log scale.

4. RESULTS

The following experiments illustrate the advantages of adding a random projection to ESNs to solve a time series prediction task. The first advantage is a better prediction for a given number of parameters. Indeed, since the number of parameters to be learnt is determined by the projection dimension, the number of hidden neurons can be increased. In consequence, a more complex dynamics can be modelled while avoiding overfitting problems. Second, the proposed contribution is an online method which can be of interest while dealing with real-time systems. A comparison to RLS without random projections and to the *Least Angle Regression* (LAR) [9] algorithm is proposed. LAR has been chosen because it particularly fits for problems where the number of training data is smaller than the number of parameters to be learnt. One can notice that an L_2 regularisation is used in the RLS case while it is an L_1 regularisation in the LAR case and that LAR is a *batch* algorithm.

The first experiment presented in Fig. 4 shows how the MSE evolves when the number of parameters p increases. The performance of the RLS is compared to the one of the RLS combined with random projections. In the first case, the number of parameters to learn p is equal to the number of neurons in the reservoir. In the second case, the number of parameters p to learn is equal to the projection dimension. One has to pay attention that the projection dimension is never larger than the number of hidden neurons which explain why some points are missing in the figure. In Fig. 4, the error

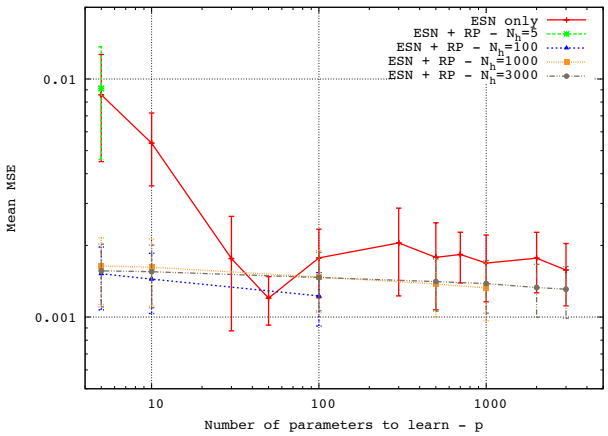

Fig. 4. Comparison of the ESNs performance with the ones of the ESN and RPs versus the number of parameters p to learn. The algorithm used for the learning is RLS.

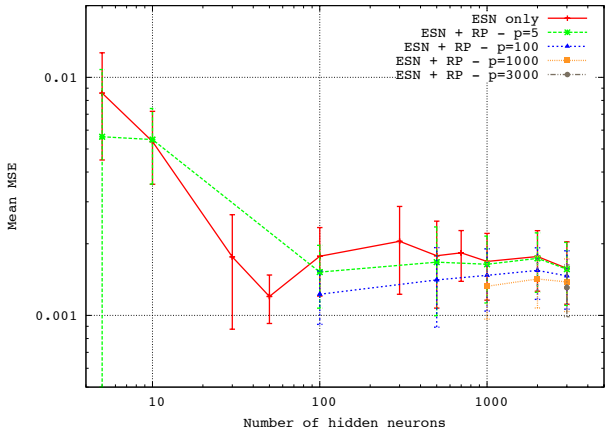

Fig. 5. Comparison of the ESNs performance with the ones of the ESN and RPs versus the number of neurons in the reservoir. Several projection dimensions are tested. The algorithm used for the learning is RLS.

with the RLS algorithm decreases when the number of hidden neurons increases to reach a minimun ($N_h = 50$). Until a reasonnable number of neurons in the reservoir, the representation of the series is richer and thus of better quality. Then, the error increases to stabilise. The regularisation performs quite well since the error does not become huge even when the number of parameters is far larger than the number of training data.

When an RP is added for the approximation, for a fixed dimension, say for example 10, the error decreases when the number of hidden neurons increases to reach a constant level. This is of interest to use RP since for a fixed number of parameters p, when not chosen too large (5 or 10), the random projections allow a better learning. The MSE is lower than when the RLS is direclty ran with the activities of the hidden neurons. One can notice that adding neurons in the hidden layer does not degrade performance.

Using a random projection after the ESN is thus interesting: for an equivalent small number of parameters to learn ($p < 100$), the testing error is lower.

The Fig. 5 presents this aspect. The projection dimension is fixed and the number of hidden neurons increases. Curves are plotted for different projection dimensions. The results got with the RLS algorithm are recalled. It appears that if the projection dimension is sufficient enough ($p > 100$), the quality of the prediction is nearly equivalent: the difference between the best and the worst trial is at most 0.001. Moreover, the number of hidden neurons does not influence the quality of the learning. It can be chosen very large with respect to the amount of training data ($N_h > 1000$) without increasing the error. The overfitting problem is thus avoided.

The two presentations of the results show the interest of using random projections: for a fixed number of parameters to learn, the error is lower and the choice of the number of parameters is made simple. Indeed, this number can be large (to ensure the dynamics of the system to be caught) without overfitting problems.

The third aspect presented is the comparison between LAR and RLS combined with random projections. This comparison is interesting since each of these algorithms somehow performs feature selection. The first one because only the features which bring the most important information are selected. The second one because the data is projected into a space of smaller dimension. The results are shown in Fig. 6. The error made with LAR tends to deacrease when the number of neurons in the reservoir increases until a minimum is

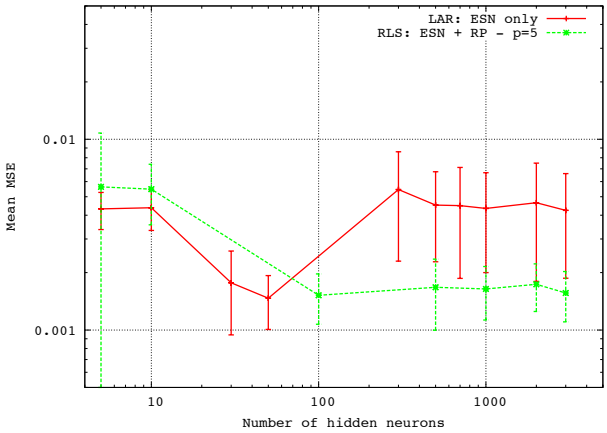

Fig. 6. Performance of the ESN trained with LAR compared to the performance of the ESN combined with random projections trained with RLS. The projection dimension is 5. The number of hidden neurons varies.

reached ($N_h = 50$). Then, when the number goes on increasing, the error increases too until $N_h = 300$ where a plateau is reached. When the random projection is used and the learning is made with the RLS algorithm, the results are of better quality when the number of hidden neurons tends to be large. Moreover, if we compare the LAR and the RLS results (Fig. 4), the L_1 regularisation (LAR) is less efficient than the L_2 regularisation (RLS).

5. CONCLUSION

Some encouraging results concerning time series prediction have been obtained by combining echo state networks and random projections. This method proposes a solution to limitate the overfitting problems. Indeed, the number of parameters to learn is decreased without needing the number of neurons in the reservoir to be decreased too (a large number of hidden neurons guarantees time dependencies to be caught). Consequently, the computational costs are also lowered. Moreover, the proposed approach has the advantage of being online.

6. REFERENCES

[1] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[2] David Verstraeten, Benjamin Schrauwen, Michiel DHaene, and Dirk Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007.

[3] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[4] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks," Tech. Rep., Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

[5] H. Jaeger, *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the Echo State Network approach*, GMD-Forschungszentrum Informationstechnik, 2002.

[6] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Proceedings of the Neural Information Processing Systems Conference (NIPS'03)*, 2003, pp. 593–600.

[7] F. Wyffels, B. Schrauwen, and D. Stroobandt, "Stable output feedback in reservoir computing using ridge regression," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN'08)*, 2008, pp. 808–817.

[8] X. Dutoit, B. Schrauwen, J. Van Campenhout, D. Stroobandt, H. Van Brussel, and M. Nuttin, "Pruning and regularization in reservoir computing," *Neurocomputing*, vol. 72, no. 7, pp. 1534–1546, 2009.

[9] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[10] D. Achlioptas, "Database-friendly random projections: Johnson-Lindenstrauss with binary coins," *Journal of computer and System Sciences*, vol. 66, no. 4, pp. 671–687, 2003.

[11] W.B. Johnson, J. Lindenstrauss, and G. Schechtman, "Extensions of Lipschitz maps into Banach spaces," *Israel Journal of Mathematics*, vol. 54, no. 2, pp. 129–138, 1986.

[12] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[13] A.F. Atiya and A.G. Parlos, "New results on recurrent network training: Unifying the algorithms and accelerating convergence," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 697–709, 2000.