An Empirical Study of the L2-Boost technique with Echo State Networks

Sebastián Basterrech

IT4Innovations
National Supercomputing Center
VSB-Technical University of Ostrava
Ostrava, Czech Republic
Sebastian.Basterrech.Tiscordio@vsb.cz

Abstract:

A particular case of Recurrent Neural Network (RNN) was introduced at the beginning of the 2000s under the name of Echo State Networks (ESNs). The ESN model overcomes the limitations during the training of the RNNs while introducing no significant disadvantages. Although the model presents some well-identified drawbacks when the parameters are not well initialized. The performance of an ESN is highly dependent on its internal parameters and pattern of connectivity of the hidden-hidden weights Often, the tuning of the network parameters can be hard and can impact in the accuracy of the models.

In this work, we investigate the performance of a specific boosting technique (called L_2-Boost) with ESNs as single predictors. The L_2-Boost technique has been shown to be an effective tool to combine "weak" predictors in regression problems. In this study, we use an ensemble of random initialized ESNs (without control their parameters) as "weak" predictors of the boosting procedure. We evaluate our approach on five well-know time-series benchmark problems. Additionally, we compare this technique with a *baseline* approach that consists of averaging the prediction of an ensemble of ESNs.

Keywords: Boosting, L_2-boosting, Echo State Network, Time-series modeling, Reservoir Computing, Ensemble Methods

I. Introduction

Boosting is a general procedure for improving the accuracy of an ensemble of methods. It has been successful used in supervised learning problems since its apparition in the 1990s [1–3]. Several variations of the original Boosting idea have been introduced over the years [4, 5], one of the most popular is called *AdaBoost* [3]. At the beginning, Boosting was used in problems where the output features were label or discrete responses (classification problems). An analogy between AdaBoost and *additive models* was studied in [4]. This connection was essential for the extension of Boosting for solving problems where the output features are continuous variables (regression problems). Bühlmann *et al.* developed a variation of the Boosting technique called L_2-*Boost* that is constructed from an additive model and the functional gradient descent method [5].

Since the early 2000s, a computational paradigm called *Reservoir Computing (RC)* has gained prominence in the Neural Computation community. In a RC model there are two well-separated concepts: a dynamical system and a memory-less function. The purpose of the dynamical system is to encode the spatio-temporal information of the input patterns into a spatial representation. At each time this dynamical system is characterized by its state that is called *reservoir* in the RC literature. This non-linear transformation is most often realized by a *Recurrent Neural Network (RNN)* with a large pool of interconnected neurons. A distinctive principle of a RC model is that the parameters of the dynamical system (the RNN weights) do not participate in the training process. That is, once the reservoir parameters are initialized, they remain fixed during the training process. Another part of the model is a memory-less supervised learning tool called *readout* structure. This part is designed to be robust and fast in the learning process.

The RC models has been applied in the neuroscience area for processing cognitive information in the neural system. [6]. Furthermore, they have proven to be extremely effective tools for time-series problems in the area of Machine Learning. For instance, as far as we know one of the most popular RC models the *Echo State Networks (ESN)* [7], has the best known learning performance on the Mackey-Glass time-series prediction problem [8, 9]. In this article, we will concentrate in the ESN model for solving time-series problems. The reservoir in the ESN model is composed by a RNN with sigmoid neurons, and the readout part of the model is a linear regression. The weight connection between neurons in the reservoir are collected in a matrix that we will call *reservoir matrix*. The main global parameters of the ESN model are: the input scaling factor, the spectral radius of the reservoir matrix and the pattern of connectivity among the reservoir units. The setting of these parameters often requires the human expertise and several empirical trials [10]. As a consequence, the setting procedure can be expensive in computational time. For instance, the time complexity of an algorithm that computes the spectral radius of a $N \times N$ matrix is equal to $O(N^4)$ [11].

The goal of this article is to investigate the performance of an automatic procedure to combine single *weak* ESNs and the L_2-Boost technique. We develop an automatic technique based on L_2-Boost, which combines the prediction of several

random initialized ESNs in order to produce a highly accurate tool. We use the term *weak ESN* for an ESN without checking and computing the spectral radius. We use the terminology *weak* due to the fact that this particular ESN is not optimal. The main advantages of the procedure presented in this article are:

- Descend the computational effort. In order to gain in the computational effort, the approach consists of combining weak ESNs. The procedure avoids to tune the reservoir parameters, which can often have a high computational cost. It uses only a uniform random initialization of the weights. Note that, there are not a control of the spectral radius, then some single weak ESNs can have unstable dynamics.

- The procedure is automatic. The procedure does not require external human expertise for setting the model parameters, and for evaluating the model performance.

- The technique has a new parameter used for *overfitting control*. This parameter which we will that comes from the L_2-Boost technique.

We present empirical results of the procedure introduced in this paper on a wide range of benchmark problems. We compare these performances with the accuracy obtained by a single ESN. Furthermore, we realize a comparison with the accuracy of a *baseline* approach that computes the average among single ESN models. Each single ESN is independently initialized and adjusted during the learning process. There is empirical evidence in the Machine Learning literature that shows that this baseline approach sometimes performs better than other ensemble methods [12].

This work is a revised and expanded version of the article [13].

The structure of this article is organized as follows. In Section II, we start with a specification of supervised learning problems with temporal data. Next, we present an overview about the family of additive models. Subsection II-C introduces the L_2-Boost technique. In Section III, we present the Reservoir Computing paradigm. Particularly, we focus on the Echo State Network model in III-A. In Subsection III-B is presented a formalization of the procedure introduced in this article. Section IV describes the empirical results. This section starts with a description of the benchmark problems. Next, we present the reached results. Finally, last section provides conclusions and future work.

II. Background

In this Section, we start specifying the context where the ESN model and the L_2-Boost technique are applied. Next, we present the additive models and we introduce a description of the L_2-Boost technique.

A. Problem Specification

We begin specifying a supervised learning problem. Given a data set $\mathcal{L} = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) : t = 1, \ldots, T\}$ where the points $\mathbf{x}$ and $\mathbf{y}$ are either a class or a numerical response. We denote by N_x the dimension of the input vector $\mathbf{x}$, and N_y the dimension of the output vector $\mathbf{y}$. We suppose that the mapping between the input $\mathbf{x}$ and the output $\mathbf{y}$ is given by certain unknown function $F(\cdot)$. The goal consists in learning a parametric function $\hat{F}(\mathbf{x}^{(t)}, \mathcal{L})$ such that certain error distance between $\hat{F}(\mathbf{x}^{(t)}, \mathcal{L})$ and $\mathbf{y}^{(t)}$ is minimized for all t. The problem is called *regression problem* when the learning set has output numerical variables. Otherwise, it is named *classification problem*. In the case of regression problems, it is recommended to use the a quadratic distance [14]. Even though we can also use a quadratic distance in classification problems, it is recommendable to use the Kullback-Leibler distance in this domain [14].

An ESN model is mainly used for solving supervised learning tasks, wherein the data set presents temporal dependencies. Although, it can be also used for non-temporal supervised learning problems [9]. In this article we will concentrate only in temporal learning tasks with real output variables ($\mathbf{y}^{(t)} \in \mathbb{R}^{N_\mathrm{Y}}$, for all t). In this work, we perform the models using a standard discrete time. We want to forecast some aspect of the output feature $\mathbf{y}$ at time $t + k$, using some aspect of the information available at current time t, that is given the collection $((\mathbf{x}^{(t)}, \mathbf{y}^{(t)}), (\mathbf{x}^{(t-1)}, \mathbf{y}^{(t-1)}), (\mathbf{x}^{(t-2)}, \mathbf{y}^{(t-2)}), \ldots)$ we would like to predict the value $\mathbf{y}^{(t+k)}$ ($k > 0$) [15]. In this case, the goal consists in estimating a mapping $\hat{F}(\cdot)$ for predicting $\mathbf{y}^{(t+k)}$ for some $k > 0$, such that some distance between $\hat{F}(\cdot)$ and $\mathbf{y}$ is minimized.

B. Additive Models

In [4] was analyzed the Boosting model under the form of an *Additive model*. Given a set of functions $f^{(m)} : \mathbb{R}^{N_\mathrm{X}} \to \mathbb{R}^{N_\mathrm{Y}}$, $m = 1 \ldots, M$ characterized by a set of parameters θ and expansion coefficients β,

$$f^{(m)}(\mathbf{x}) = \beta^{(m)} h(\mathbf{x}, \theta^{(m)}),$$

an additive model has the following form

$$F(\mathbf{x}) = \sum_{m=1}^{M} f^{(m)}(\mathbf{x}). \tag{1}$$

The functions $\{h(x; \theta)\}_1^M$ are named *basis functions*. They are not fixed *a priori* and are selected depending of the cost function used and the data set. An important parameter of the model is the number of basis functions (M) considered in the expression (1). This parameter controls the generalization error of the model. Since the main goal in a learning task is to find a predictor with low generalization error, the parameter M has an important role in the accuracy of an additive model.

C. The L_2-Boost Procedure

A relationship between the gradient descent technique and stage-wise additive expansions was introduced at the beginning of the 2000s [16]. The introduction of the gradient descent algorithm using a boosting approach was an essential contribution in the field of ensemble learning methods [16]. It allowed to start to use boosting in regression problems [5]. A Boost method for regression problems with quadratic error distance was introduced under the name of L_2-Boost in [5]. We present the L_2-Boost technique in Algorithm 1. Other boosting variants were presented for other kind of distances, some of them are described in [4, 5, 16].

We refer by *epoch* to the iteration of the training algorithm through all the patterns in the training set [17]. At each epoch $m + 1$, the basis function $h^{(m+1)}(\cdot, \theta)$ is fitted to the current residuals: $\mathbf{y}^{(i)} - \hat{F}^{(m)}(\mathbf{x}^{(i)})$, for all i. Unlike other boosting techniques such as Adaboost, L_2-Boost does not present any re-weighting. Another difference between L_2-Boost and other boosting methods is that L_2-Boost presents a tendency to over-fit the data [5]. The model with contracting linear learners converge to the fully saturated model [5]. Each boosting epoch contributes to additional overfitting, thus the selection of the weak learners and the parameter M is an essential task for this device. In practice, few boosting iterations are enough to achieve good performances avoiding the overfitting phenomena.

Algorithm 1 The L_2-Boost algorithm.

Require: $\mathcal{L}, M, h(\mathbf{x}, \theta)$

  Fit an initial model using a least squares fit (see [18]): $\hat{F}^{(0)}(\cdot) = h(\cdot, \theta)$;
  for $(m = 1, \ldots M)$ **do**
    Compute the residuals for all pattern i: $\mathbf{e}^{(i)} = \mathbf{y}^{(i)} - \hat{F}^{(m)}(\mathbf{x}^{(i)})$;
    Fit the model $\hat{f}^{(m+1)}(\cdot)$ parametrized as $\hat{f}^{(m+1)}(\mathbf{x}) = h(\mathbf{x}, \theta)$ to the current residuals e using the least squares fit;
    Update: $\hat{F}^{(m+1)}(\cdot) = \hat{F}^{(m)}(\cdot) + \hat{f}^{(m+1)}(\cdot)$;
  end for
  Return the $\hat{F}^{(M)}(\cdot)$ function;

III. Modeling Time-series with Echo State Networks

The *Recurrent Neural Networks (RNNs)* are powerful tools for solving time-series benchmarks. They are computational methods that operate in time. Considering terminology of graphs, in a RNN at least one circuit is presented in its topology. The circuits of the network enable to store temporal information, in order to learn and memorize the input history [9]. Each circuit creates an internal state which makes the recurrent network a discrete time state-space model. At each time, the RNN receives an input pattern. Next, the network update its hidden state via a non-linear activation function using the input pattern and the network state at the precedent time [19]. There is a general consensus in the community that considers the RNN as powerful tool for forecasting and time-series prediction.

In spite of that, in practice the model presents some drawbacks. The most important is that is hard to train a RNN using gradient descent methods [20]. The training methods that use the first differential information have often stability problems and high numerical complexity. As a consequence, much longer training times are necessary to adjust the network weights. In [20] is analyzed the main limitations of the algorithms of the gradient descent type for training RNNs. These drawbacks are identified under the names of *vanishing* and the *exploding* gradient problems [20]. The vanishing gradient phenomena occurs when the norm of the gradient decreases arbitrarily fast to 0. The exploding gradient phe-

nomena refers to the opposite, when the gradient norm large increases during the training process [21]. Recently, an effective algorithm to train RNN was introduced [19], the algorithm uses the *Hessian-free Optimization* for setting the network parameters.

Reservoir Computing (RC) models appear as a good alternative for RNNs. The two pioneering RC models are *Echo State Network (ESN)* [7] and *Liquid State Machine (LSM)* [22]. This computational paradigm covers the main limitations related to learning processes in RNNs obtaining acceptable performance in practical applications [9]. In a RC model there are at least two well-differentiated structures: a dynamical system called *reservoir* and another one called *readout*. The readout is a supervised learning tool for training with non-temporal data. For example: feedforward neural network, linear regression, decision trees, etc. A main characteristic of a RC model is that the weights involved in circuits are deemed fixed during the learning process. Thus, the matrix with the weight between reservoir units (reservoir matrix) is initialized in an arbitrary way and it remains unchanged during the learning process. The training algorithm is restricted to update the weights in the readout structure. Over the last years several kinds of dynamical systems have been used for generating the reservoir state, models include: Backpropagation-decorrelation Recurrent Learning [23], Leaky Integrator Echo State Networks studied [24], Evolino [8], Intrinsic Plasticity [25], Echo State Queueing Networks [26], Reservoir Computing and Extreme Learning [27], and so on.

A. Formalization of the Echo State Network Model

In this work related to the L_2-Boost technique and the RC methods, we only study the L_2-Boost with the ESN model. An ESN reservoir is a RNN from an input space $\mathbb{R}^{N_x}$ into a larger space $\mathbb{R}^{N_s}$ with $N_x \ll N_s$. The connection between input and hidden neurons are collected in a $N_s \times N_x$ weight matrix $\mathbf{w}^{\text{in}}$. The connections among the hidden neurons are represented by a $N_s \times N_s$ weight matrix $\mathbf{w}^r$. A $N_y \times N_s$ weight matrix $\mathbf{w}^{\text{out}}$ represents the readout weights. At any time t, the information from the input pattern and the past is represented in a state vector

$$\mathbf{s}(t) = \tanh(\mathbf{w}^{\text{in}}\mathbf{x}^{(t)} + \mathbf{w}^r\mathbf{s}^{(t-1)}). \tag{2}$$

At any time t, the output prediction $\mathbf{y}^{(t)} \in \mathbb{R}^{N_y}$ is generated using the input pattern and the reservoir state information. Most often is computed using a linear regression:

$$\mathbf{y}^{(t)} = \mathbf{w}^{\text{out}}[\mathbf{x}^{(t)}|\mathbf{s}^{(t)}], \tag{3}$$

where $\cdot|\cdot$ is the vertical concatenation of the vectors. For the sake of the notation simplicity, we omit the bias term, a constant term is included in all the regressions.

In [7] was analyzed the stability of the reservoir dynamics in the ESN model. Under certain algebraic conditions the reservoir state only depends (asymptotically) of the inputs and the network topology. It becomes independent of its initial conditions [9]. These conditions were summarized in the *Echo State Property (ESP)* [7]. In practice, the stability of the ESN is almost always ensured when the spectral radius of the reservoir matrix is less than 1 [9, 28]. As a consequence, the reservoir weights are appropriately scaled in

order to have a spectral radius less than 1. To scale the parameters is necessary to compute the spectral radius of the reservoir matrix. The computation of the spectra requires an important computational effort [11]. Some attempts to generate a procedure for initializing the RC models were introduced in [11, 25, 29–31].

B. L_2-Boost Using the ESN Model for Time-series Processing Information

In this article, we investigate the performance of using L_2-Boost in temporal learning tasks, and we consider as *weak learner predictors* a set of ESNs with random initialization. Given an arbitrary parameter M the procedure is as follows. We initialize an ESN in a random way. The initialization consists in selecting the size of the network as well as the pattern of connectivity. We consider a reservoir with fixed sparse connections. We do not control the spectrum norm of the reservoir weight. A guide about the initialization procedure can be seen from [10]. We expand the input information using the ESN reservoir given by the expression (2), thus we obtain $\mathbf{s}^{(t)}$, $\forall t$. Next, we apply Algorithm 1. Finally, we obtain predictor $F^{(M)}(\cdot)$. The approach is summarized in Algorithm 2. In our experiments we use ridge linear regression for computing the readout weights $\mathbf{w}^{\text{out}}$ [28].

Algorithm 2 The L_2-Boost with the ESN model.

Initialize an ESN following the comments in Subsection III-B;
Compute the temporal expansion of $\mathcal{L}$ using (2);
Generate the set $\{(\mathbf{s}^{(t)}, \mathbf{y}^{(t)}), \forall t\}$;
Apply the Algorithm 1;
Return $\hat{F}^{(M)}(\cdot)$;

In order to evaluate the performance of this procedure, we compare the reached accuracy of the L_2-Boost technique with a simple *baseline* approach [17]. The baseline approach consists in combining K single predictors (in our case the learning predictors are ESNs). We consider random initialized reservoirs, without control of the reservoir spectrum norm. In the baseline method, we train independently each of these single ESNs. The final prediction is the average among the single predictions.

For statistical comparisons between the methods we consider $K = 30$. We remark again that we do not scale the reservoir weights for obtaining the ESP. Even though some ESN models can present good accuracy, other ones can be weak predictors. Additionally, we compare our performances with the performance obtained when single ESNs with "good" tuning of the reservoir parameters are used. For that, we use the results presented in the RC literature.

IV. Empirical Results

We begin this section describing the benchmark problems. Next, we specify the experimental setup. We concludes this section with an analysis of our empirical results.

A. Description of the Benchmark Problems

We use the following range of time-series benchmarks:

- Fixed kth order NARMA. This data set presents a high non-linearity and is widely used in the RC literature. We generate the NARMA serie following the description in [28, 32],

$$b(t+1) = \alpha_1(t) + \alpha_2 b(t) \sum_{i=0}^{k-1} b(t-i)$$
$$+ \alpha_3 s(t-(k-1))s(t) + \alpha_4,$$

where $s(t) \sim Unif[0, 0.5]$ and the constants values are shown in Table 1. In order to evaluate the memorization ability of the model, we consider two simulated series when $k = 10$ and $k = 30$. The task consists to predict

k	α_1	α_2	α_3	α_4
10	0.3	0.05	1.5	0.1
30	0.2	0.004	1.5	0.001

Table 1: Parameters considered for the fixed kth order NARMA serie with $k = 10$ and $k = 30$.

the value $y(t+1)$ based on the history of $y(t)$ up to time t. We used the first 200 samples as initial washout in both the training and test procedure. The regularization parameter used was 0.00001.

- The Santa Fe Laser data set [33]. It is an experimental data that contains the intensity pulsations of a real laser recorded by a LeCroy oscilloscope. The data is a cross-cut through periodic to chaotic intensity laser pulsations. These pulsations more or less follow the theoretical Lorenz model of a two level system [33]. In this problem, the task consists to predict the next measure $y(t+1)$, given the precedent values up to t. The original data only consists of 1000 measurements, we used for training 499 samples and for testing 500 samples. We used a washout of 10 samples. The regularization parameter γ was 0.001.

- Henon Map data set. It is a prototypical invertible map with chaotic solutions proposed in [34]. The data is generated by

$$y(t+1) = 1 - 1.4(y(t))^2 + 0.3y(t-1) + z(t+1),$$

where the noise is $z(t) \sim \mathcal{N}(0, 0.05)$. The data is normalized in $[0, 1]$. The goal is to predict the next value $y(t+1)$ with the past information up to t. We considered a training data with 3995 samples and a test data with 795 samples. The regularization parameter γ used was 0.001. We use an initial washout composed by 100 samples. The network topology has 3 input units set with the last two precedent $y(t)$ values and the noise at current time.

- Freedman's non linear time data set [35]. The data is generated by

$$y(t+1) = g(y(t)),$$

where:

$$g(x) = \begin{cases} 2x, & \text{if } x \leq 0.5, \\ 2 - 2x, & x > 0.5. \end{cases}$$

We consider a very short data set. The length of the training data was 30 and the test size was 19. The initial value is $y(0) = 0.23719$. The initial washout considered only was of 3 samples. The network topology has only one input unit, several reservoir units and one output unit. The regularization parameter γ used was 0.001.

B. Experimental Setup

DATA	Initial Washout	γ	Train samples	Test samples
10th NARMA	200	0.00001	1400	2400
30th NARMA	200	0.00001	1600	2600
Santa Fe Laser	10	0.001	499	500
Henon Map	100	0.001	3995	795
Freedman's	3	0.001	30	19

Table 2: Parameter setting of the benchmark problems. In all cases, we initialize the input weights ($\mathbf{w}^{in}$) using Uniform distribution in $[-0.2, 0.2]$, and we initialize the reservoir weights with an Uniform distribution in $[-0.8, 0.8]$.

We summarize the setting of the main parameters related to the benchmark problems in Table 2. The table presents the initial washout period, the regularization parameter (γ) of the linear ridge regression, and the number of train and test samples for each benchmark problem.
The benchmarks selected have been widely used in the RC literature [7, 28, 31, 36]. In all cases, we use the *Normalized Mean Square Error (NMSE)* as measure of accuracy model [9]. The learning method used for computing the output weight matrix $\mathbf{w}^{out}$ was the offline ridge regression. This algorithm has a regularization parameter γ that we adjust it for each benchmark problem. The pre-processing data step consisted in normalizing the patterns in the interval $[0, 1]$ We investigated the algorithm performance for several reservoir sizes. The range of the reservoir size values is specified for each benchmark problem. The connection between the input and reservoir layer is fully connected with random weights in $[-0.2, 0.2]$. The reservoir matrix is initialized using Uniform distribution in $[-0.8, 0.8]$.

C. Result Analysis

Table 3 shows results reported in the RC literature for these benchmarks when a single ESN model was used as model predictor. Table 4 presents the train set accuracy reached on the Henon Map data set. The columns 2, 3 and 4 show the NMSE reached with L_2-Boost with ESNs for M epochs ($M = 3, 4$ and 5), respectively. Column 5 of Table 3 shows the accuracy of the baseline approach, that it averaging the prediction of 30 ESNs. The columns of the table are written using a scientific notation.
Table 4 illustrates the accuracy of the models during the training. The NMSE corresponds to the training data of the Henon Map data set. We present this table in order to illustrate the tendency of overfitting of L_2-Boost with ESN. The additive model converge very fast to the solution, for this reason the columns 3 and 4 of Table 4 are very similar. The

model with larger M performs better over the train data, but it has problems of generalization. We found this characteristic in all benchmarks. As a consequence, we can affirm that the parameter M has a relevant impact in the control of the overfitting phenomenon. We can found a similar remarks for the L_2-Boost technique in non-temporal learning tasks [5].
Figure 2 illustrates the NMSE reached according the reservoir size for different M values for the 30th NARMA data set. This figure shows the training error, we can see the evolution of the NMSE versus the size of the reservoir. We present few values of reservoir size between 6 till 11. Figures 1 and 3 show the NMSE of the test data versus the reservoir size for the 10th and 30th order NARMA data set, respectively. These figures show 4 curves, the black one (with points represented by dots) corresponds to the baseline method which combines several single ESNs. The other curves correspond to the L_2-Boost-ESN with different number of epochs $M = 6, 8$ and $M = 10$. We can not affirm that the procedure of L_2-Boost with single weak ESNs performs better than *optimal* single ESNs. The accuracy it is also of the same order that results presented in the RC literature using a single well-initialized ESN [28, 31].
Figure 4 illustrates the evolution of the NMSE for the reservoir size of the test data of the Santa Fe Laser benchmark. The error was computed for the L_2-Boost with ESNs for $M = 4, 6$ and $M = 8$ and the baseline approach averaging 30 ESNs. Figure 5 shows the accuracy reached for the models on the Freedman test data set. The graphic shows the evolution of the L_2-Boost with ESNs for $M = 4, 5$ and $M = 5$ and the baseline approach. In all graphics, we can see that when the reservoir increases its size the procedure L_2-Boost with ESNs and the baseline approach decrease their test error. This behavior about the impact of the reservoir size on the accuracy of the model, also happens with single ESNs [10, 28, 31].

DATA	ACCURACY	N_x	REF.
10TH NARMA	0.166 (NMSE)	50	[28]
	0.0425 (NMSE)	200	[28]
30TH NARMA	0.4542 (NRMSE)	100	[30]
SANTA FE LASER	0.0184 (NMSE)	50	[28]
	0.00819 (NMSE)	200	[28]
HENON MAP	0.00975 (NMSE)	50	[28]
	0.00868 (NMSE)	200	[28]
FREEDMAN'S	0.0004302 (MSE)	40	[31]

Table 3: Accuracy of the ESN model for the benchmark problems. Second column shows the accuracy reached by the single ESN, third column refers the reservoir size and the last column shows a bibliographic reference. In the case of the Freedman's non linear time data, the reservoir initialization was done using the Scale Invariant Map method [31], and the Mean Square Error (MSE) was the error measure. In the case of 30th NARMA, the authors initialize the reservoir using permutation matrices. The error measure was the Normalized Root Square Error (NRMSE) [30]. In the other benchmarks problems, the authors control some reservoir parameters such as: spectral radius and reservoir matrix density.

N_x	$M = 3$ (1.0E-12)	$M = 4$ (1.0E-13)	$M = 5$ (1.0E-13)	BAS. ESN (1.0E-12)
6	0.297367	0.131036	0.131036	0.288720
7	0.191883	0.131036	0.131036	0.064412
8	0.142289	0.131036	0.131036	0.572728
9	0.160910	0.131036	0.131036	0.182942
10	0.203845	0.131036	0.131036	0.278237
11	0.272649	0.131036	0.131035	0.524017
12	0.488265	0.131036	0.131036	0.459002

Table 4: Train set performance of the Henon Map data set. First column indicates the number of neurons in the reservoir. The columns 2, 3 and 4 show the NMSE obtained with L_2-Boost with M epochs ($M = 3, 4$ and 5), respectively. Column 5 shows the accuracy of the baseline approach. That is the accuracy average among 30 ESN predictions. The columns are written using a scientific notation.
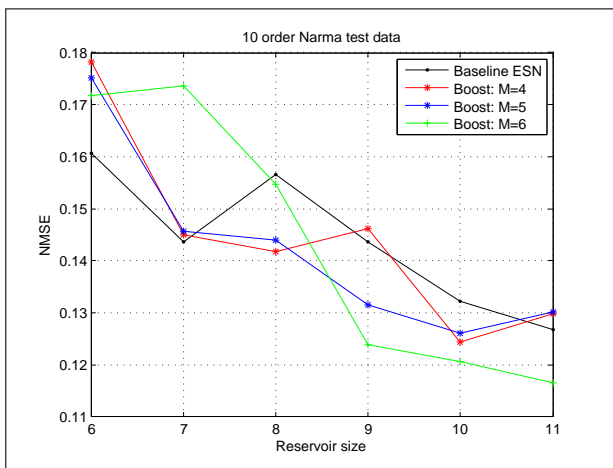
Figure. 1: Test set accuracy reached for the 10th order NARMA data set. The vertical axis of the graph shows the NMSE accuracy, and the horizontal axis presents some values of the reservoir size. We compare the accuracy of L_2-Boost: $M = 4, 5$ and 6 with the baseline approach averaging 30 ESNs.

V. Conclusions and Future Work

At the beginning of the 2000s, an efficient technique to train and design a RNN was developed under the name of Echo State Network (ESN). This approach overcome the limitations to train RNN using the gradient descent method. The performance of an ESN is highly dependent on its parameters and pattern of connectivity of the hidden-hidden weights Besides, the network setting can be computational expensive, in particular to compute the spectral radius of the hidden-hidden weight matrix.

In this article, we investigated boosting ideas with ESNs, in order to built a robust new learning tool. In particular, we studied the utilization of L_2-Boost with random initialized ESNs. We merge a set of *weak* single ESNs. We call *weak* ESNs because they are random initialized, and we do not use extra computational effort for tuning the initial hidden-hidden weights.

In spite of the realization of numerous tests, we can not affirm that L_2-Boost with ESNs performs better than a single
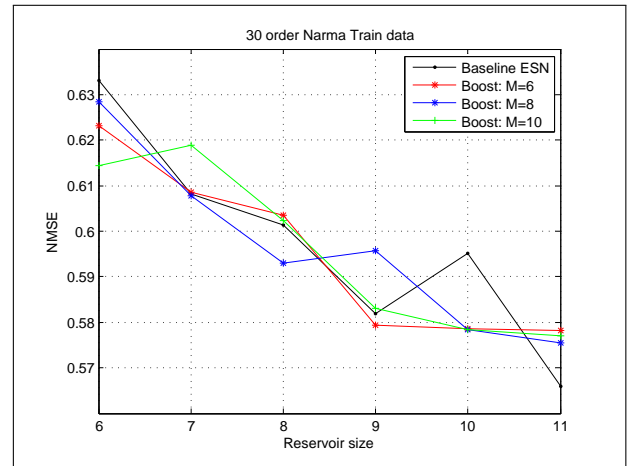
Figure. 2: The accuracy reached on the training set of the 30th order NARMA data. The vertical axis of the graph shows the NMSE accuracy, and the horizontal axis presents some values of the reservoir size. We compare the accuracy of L_2-Boost: $M = 6, 8$ and 10 with the baseline approach averaging 30 ESNs.
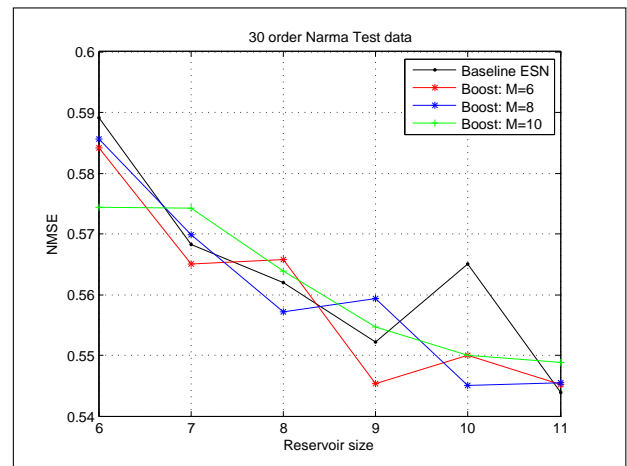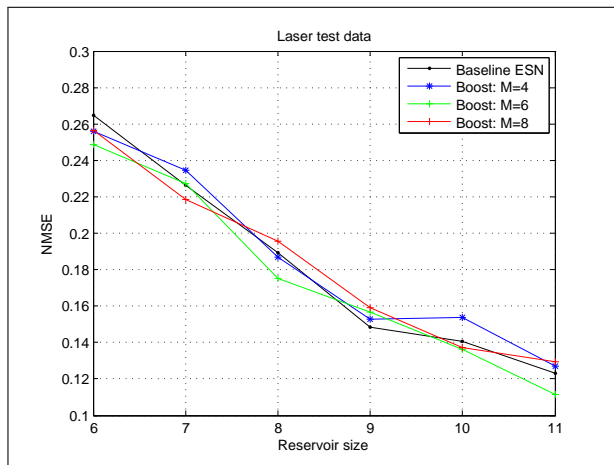

Figure. 3: The accuracy reached for the 30th order NARMA data set. The vertical axis of the graph shows the NMSE accuracy, and the horizontal axis presents some values of the reservoir size. We compare the accuracy of L_2-Boost: $M = 6, 8$ and 10 with the baseline approach averaging 30 ESNs.

well-initialized ESN (according the results presented in the RC literature). However, the main advantage of the L_2-Boost with *weak* ESNs is that the procedure is automatic and does not require the computational effort of computing the spectra of the hidden-hidden weight matrix. Additionally, the procedure has a control parameter for the overfitting phenomena. In a future work we will test the model using another technique for decrease the generalization error, as well as on a more number of benchmark problems. Additionally, we can test the approach using another supervised learning tool for the readout structure.

Acknowledgment

Figure. 4: Test set accuracy of the Santa Fe Laser data. The vertical axis of the graph shows the NMSE accuracy, and the horizontal axis presents some values of the reservoir size. We compare the accuracy of L_2-Boost: $M = 4, 6$ and 8 with a baseline approach averaging 30 ESNs.
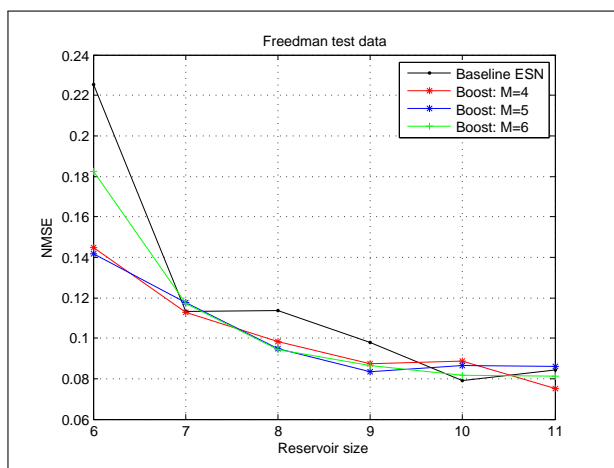
Figure. 5: Test set performance of the Freedman data set. Size of the reservoir versus the NMSE accuracy. We compare the accuracy of L_2-Boost: $M = 4, 5$ and 6 with a baseline approach averaging 30 ESNs.

References

[1] R. E. Shapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, 1990.

[2] Y. Freund, "Boosting a weak learning algorithm by majority," AT&T Bell Laboratories, New Jersey, USA, Tech. Rep., January 1995.

[3] Y. Freund and R. E. Shapire, "Experiments with a new Boosting Algorithm," *Machine Learning: Proceedings of Thirteenth International Conference*, 1996.

[4] J. Friedman, T. Hastie, and R. Tibshirani, "Additive Logistic Regression: a Statistical View of Boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[5] P. Bühlmann and B. Yu, "Boosting with the L2 loss: Regression and Classification," *Journal of the American Statistical Association*, vol. 98, 2003.

[6] W. Maass, "Liquid State Machines: Motivation, Theory, and Applications," in *In Computability in Context: Computation and Logic in the Real World*. Imperial College Press, 2010, pp. 275–296.

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

[8] M. G. J. Schmidhuber, D. Wierstra and F. Gomez, "Training Recurrent Networks by Evolino," *Neural Networks*, vol. 19, pp. 757–779, 2007.

[9] M. Lukoševičius and H. Jaeger, "Reservoir Computing Approaches to Recurrent Neural Network Training," *Computer Science Review*, pp. 127–149, 2009.

[10] M. Lukoševičius, "A Practical Guide to Applying Echo State Networks," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Montavon, G. Orr, and K.-R. Müller, Eds. Springer Berlin Heidelberg, 2012, vol. 7700, pp. 659–686. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35289-8_36

[11] A. A. Ferreira, T. B. Ludermir, and R. R. B. De Aquino, "An Approach to Reservoir Computing Design and Training," *Expert Syst. Appl.*, vol. 40, no. 10, pp. 4172–4182, Aug. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2013.01.029

[12] D. Opitz and R. Maclin, "Popular Ensemble Methods: an Empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[13] S. Basterrech, "An Empirical Study of L2-Boost with Echo State Networks," *IEEE Intelligent Systems Design and Applications (ISDA)*, December 2013.

[14] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of Statistical Learning*, ser. Spring series in statistics. New York, USA: Springer-Verlag, 2001.

[15] Y. Bengio and N. Chapados, "Extensions to metric-based model selection," *Journal of Machine Learning Research*, vol. 3, pp. 1209–1227, 2003.

[16] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[17] H. Schwenk and Y. Bengio, "Boosting Neural Networks," *Neural Computation*, vol. 12, no. 8, pp. 1869–1887, Aug. 2000.

[18] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, UK: Cambridge University Press, 1992.

[19] J. Martens and I. Sutskever, "Learning Recurrent Neural Networks with Hessian-Free Optimization," in *Proceeding of the 28th International Conference on Machine Learning*, 2011, pp. 1033–1040.

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

[21] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," *Proceedings of the 30th International Conference on Machine Learning*, vol. 28, pp. 37–48, 2013.

[22] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: a new framework for a neural computation based on perturbations," *Neural Computation*, pp. 2531–2560, november 2002.

[23] J. J. Steil, "Backpropagation-Decorrelation: online recurrent learning with O(N) complexity," *In Proceedings of IJCNN'04*, vol. 1, 2004.

[24] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of Echo State Networks with leaky-integrator neurons," *Neural Networks*, no. 3, pp. 335–352, 2007.

[25] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil, and D. Stroobandt, "Improving Reservoirs using Intrinsic Plasticity," *Neurocomputing*, vol. 71, pp. 1159–1171, March 2007.

[26] S. Basterrech and G. Rubino, "Echo State Queueing Network: a new Reservoir Computing learning tool," *IEEE Consumer Comunications & Networking Conference (CCNC'13)*, January 2013.

[27] J. B. Butcher, D. Verstraeten, B. Schrauwen, C. R. Day, and P. W. Haycock, "Reservoir Computing and Extreme Learning Machines for Non-linear Time-series Data Analysis," *Neural Networks*, vol. 38, pp. 76–89, feb 2013.

[28] A. Rodan and P. Tiňo, "Minimum Complexity Echo State Network," *IEEE Transactions on Neural Networks*, pp. 131–144, 2011.

[29] M. Lukoševičius, "On self-organizing reservoirs and their hierarchies," Jacobs University, Bremen, Tech. Rep. 25, 2010.

[30] J. Boedecker, O. Obst, N. M. Mayer, and M. Asada, "Studies on Reservoir Initialization and Dynamics Shaping in Echo State Networks," in *Proceedings of the 17th European Symposium On Artificial Neural Networks (ESANN'09)*, Evere, Belgium, Apr. 2009, pp. 227–232.

[31] S. Basterrech, C. Fyfe, and G. Rubino, "Self-Organizing Maps and Scale-Invariant Maps in Echo State Networks," in *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, nov. 2011, pp. 94–99.

[32] S. Dasgupta, F. Worgotter, and P. Manoonpong, "Information dynamics based self-adaptive reservoir for delay temporal memory tasks," *Evolving Systems*, pp. 1–15, 2013.

[33] U. Huebner, N. B. Abraham, and C. O. Weiss, "The Santa Fe Time Series Competition Data," available at: http://goo.gl/6IKBb9, date of access: 12 September 2013.

[34] M. Hénon, "A two dimensional mapping with a strange attractor," *Commun. Math. Phys.*, vol. 50, pp. 69–77, 1976.

[35] R. Hyndman, "Time series data library," available at: http://goo.gl/PZzReR, date of access: 12 September 2013.

[36] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, no. 3, pp. 287–289, 2007.

Author Biographies

Sebastián Basterrech received a M.Sc. degree in Applied Mathematics in 2008 from the Aix-Marseille University, France. From 2009 to 2012 he received a doctoral fellowship from the National Institute for Research in Computer Science and Control (INRIA), France. He obtained the Ph.D. degree in Computer Sciences in November, 2012 from INRIA-Rennes and the University of Rennes 1, France. Since June, 2013 he is a postdoctoral researcher at the *IT4Innovations* National Supercomputing Center, VŠB-Technical University of Ostrava, Czech Republic. The main research interests of S. Basterrech include spatio-temporal data mining, recurrent neural networks and bio-inspired algorithms for engineering applications.