CORE
Metadata, citation and similar papers at core.ac.uk
Provided by Ghent University Academic Bibliography

Can Non-Linear Readout Nodes Enhance the Performance of Reservoir-Based Speech Recognizers?

Fabian Triefenbach Jean-Pierre Martens
Department of Electronics and Information Systems
Ghent University
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
`fabian.triefenbach@elis.ugent.be`

***Abstract*—It has been shown for some time that a Recurrent Neural Network (RNN) can perform an accurate acoustic-phonetic decoding of a continuous speech stream. However, the error back-propagation through time (EBPTT) training of such a network is often critical (bad local optimum) and very time consuming. These problems hamper the deployment of sufficiently large networks that would be able to outperform state-of-the-art Hidden Markov Models. To overcome this drawback of RNNs, we recently proposed to employ a large pool of recurrently connected non-linear nodes (a so-called reservoir) with fixed weights, and to map the reservoir outputs to meaningful phonemic classes by means of a layer of linear output nodes (called the readout nodes) whose weights form the solution of a set of linear equations. In this paper, we collect experimental evidence that the performance of a reservoir-based system can be enhanced by working with non-linear readout nodes. Although this calls for an iterative training, it boils down to a non-linear regression which seems to be less critical and time consuming than EBPTT.**

***Keywords*-Reservoir Computing, Recurrent Neural Networks, Linear Regression, Logistic Regression, On-line Training, Automatic Speech Recognition**

I. Introduction

Using Recurrent Neural Networks (RNN) as classifiers has a long tradition in the domain of machine learning and it is often argued that these systems can be seen as an implementation with a cognitive motivation. Especially the recurrent connections that are used to exchange information between computational nodes also seem to exist in many brain structures. In the early nineties RNNs were also successfully applied to speech recognition [1] but due to the critical training procedure [2], it was difficult to scale them up to reach state-of-the-art results on today's continuous speech recognition benchmarks.

In our previous work we presented an alternative neural implementation where the idea of a RNN is adopted but simplified by imposing architectural constraints which lead to a much simpler training procedure. We were able to show that this paradigm of Reservoir Computing (RC) [2]–[4] can be successfully used for isolated and connected digit recognition in clean and noisy conditions [5] as well as for continuous phoneme recognition [6]. All results were obtained with systems comprising a recurrently connected neural network of nodes with randomly fixed input and recurrent weights (a so-called reservoir) and a layer of linear readout nodes with trained weights. Each readout node computes a linear combination of the reservoir node outputs. The readout weights are designed so that a regularized mean squared error between the computed outputs and the desired outputs for a set of training examples is minimized. Under these constraints, there exists a closed-form solution for the weights which can be obtained by inverting a squared matrix and performing some additional matrix multiplications.

In the present study we systematically investigate how the performance of a reservoir-based system is affected by changing the constraints to which it is subjected. With performance we not only mean accuracy here, but also memory requirements, computational load, training time and scalability to complex problems.

The paper is organized as follows. First we review the concept of traditional RC (Section II). Then we discuss the changes that are necessary to accommodate non-linear readouts, followed by the new opportunities that emerge to further improve the system performance (Section III). And subsequently, we present an experimental assessment of the architectures we investigated (Section IV). We end the paper with some conclusions and ideas for future work.

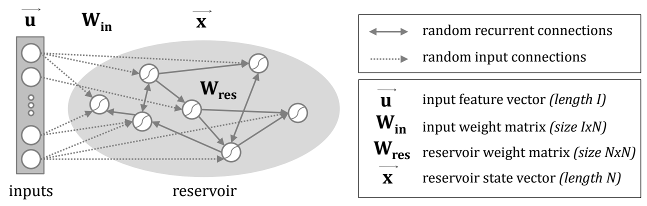
II. Traditional Reservoir Computing

A. The reservoir

A traditional reservoir, originally called an Echo State Network [2], is a dynamical system that is composed of a pool of recurrently connected non-linear computational nodes. The reservoir is stimulated by time varying inputs (see Figure 1). If the inputs at time t are given by $\mathbf{u}[t]$ and if the outputs of the reservoir nodes at time t constitute the reservoir state vector $\mathbf{x}[t]$, then $\mathbf{x}[t]$ is computed as

$$\mathbf{x}[t] = f_{res}(\mathbf{W}_{res}\mathbf{x}[t-1] + \mathbf{W}_{in}\mathbf{u}[t]) \tag{1}$$

In our case, the activation function of the reservoir nodes is either $f_{res}(z) = \text{logistic}(z)$ or $f_{res}(z) = \tanh(z)$. The weight matrices $\mathbf{W}_{in}$ and $\mathbf{W}_{res}$ specify which input-to-reservoir and which internal reservoir connections are present and how important they are.



$\vec{\mathbf{u}}$
$\mathbf{W_{in}}$
$\vec{\mathbf{x}}$
$\mathbf{W_{res}}$
inputs
reservoir
random recurrent connections
random input connections
$\vec{\mathbf{u}}$ input feature vector *(length I)*
$\mathbf{W_{in}}$ input weight matrix *(size IxN)*
$\mathbf{W_{res}}$ reservoir weight matrix *(size NxN)*
$\vec{\mathbf{x}}$ reservoir state vector *(length N)*

Figure 1. A reservoir contains non-linear computational nodes that are stimulated by inputs as well as by reservoir nodes. The interconnections between the nodes are described by two interconnection matrices.

Due to its recurrent connections, a reservoir can capture long-term dynamics of the input data and is therefore an appealing method to create feature vectors that are determined by the recent past. In addition, the reservoir can memorize its node activations by integrating information over time by using leaky integrator nodes [4] rather than the classical memoryless nodes. In that case, Equation 1 is extended to

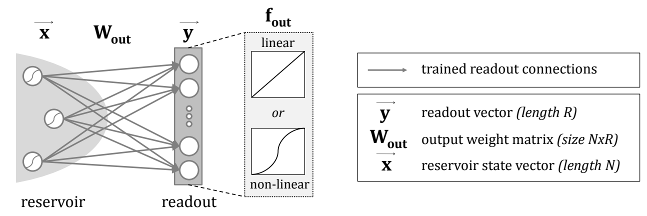
$$\mathbf{x}[t] = (1-\lambda)\mathbf{x}[t-1] + \lambda f_{res}(\mathbf{W}_{res}\mathbf{x}[t-1] + \mathbf{W}_{in}\mathbf{u}[t]) \quad (2)$$

For $\lambda < 1$ the leaky nodes implement a fading memory of the past state vectors.

The reservoir weights are fixed randomly but the weight fixing process uses some control parameters to guarantee a stable reservoir system that analyses the inputs at a desired timescale. One of these parameters is the spectral radius (SR) [3], defined as the largest absolute eigenvalue of the recurrent weight matrix. The SR can easily be controlled by rescaling the recurrent weight matrix. By making it smaller than one, a stable system can be guaranteed. The smaller the SR is, the less impact the recurrent connections will have on the behavior of the reservoir. The parameter λ (leak rate) encodes the integration time constant τ (in time units) of the reservoir nodes via $\lambda = 1 - e^{-1/\tau}$. To control the excitability of the reservoir by the inputs one can change the maximum magnitude of the input weights and the sparseness of the input weight matrix.

B. The readout layer

In order to use reservoirs for e.g. classification, one needs a layer of readout nodes (see Figure 2) that 'read out' the



$\vec{\mathbf{x}}$
$\mathbf{W_{out}}$
$\vec{\mathbf{y}}$
$\mathbf{f_{out}}$
linear
or
non-linear
reservoir
readout
trained readout connections
$\vec{\mathbf{y}}$ readout vector *(length R)*
$\mathbf{W_{out}}$ output weight matrix *(size NxR)*
$\vec{\mathbf{x}}$ reservoir state vector *(length N)*

Figure 2. In order to interpret the reservoir state vector $\mathbf{x}$, a readout layer is trained to classify the state vector into predefined classes. The readout nodes are normally linear.

reservoir state. The readout $\mathbf{y}[t]$ is then computed as

$$\mathbf{y}[t] = f_{out}(\mathbf{W}_{out}\,\mathbf{x}[t]) \quad (3)$$

with $\mathbf{W}_{out}$ characterizing the reservoir-to-output connections and with f_{out} representing the activation function of the readout nodes. In a traditional Echo State Network, this is a linear function $f_{out}(z) = z$.

The readouts are meant to represent meaningful classes and the non-zero elements of the output weight matrix $\mathbf{W}_{out}$ are determined so as to minimize the mean difference between the computed readouts and the desired readouts over a set of labeled training examples. In traditional reservoir systems, the readout nodes are linear and the error criterion is a regularized mean squared error. If the training state vectors (created by the reservoir) form the N_t rows of matrix $\mathbf{X}$ and if the corresponding desired targets form a matrix $\mathbf{D}$, then $\mathbf{W}_{out}$ is given by

$$\mathbf{W}_{out} = \arg\min_{\mathbf{W}} \left(\frac{1}{N_t} \left(||\mathbf{X}\,\mathbf{W} - \mathbf{D}||^2 \right) + \epsilon\ ||\mathbf{W}||^2 \right) \quad (4)$$

The regularization term $\epsilon\ ||\mathbf{W}||^2$ is meant to limit the norm of the output weights in order to prevent over-fitting.

Equation 4 boils down to a Ridge Regression [7] and its closed-form solution is given by

$$\mathbf{W}_{out} = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \epsilon\,\mathbf{I})^{-1}(\mathbf{X}^{\mathbf{T}}\mathbf{D}) \quad (5)$$

with $\mathbf{I}$ representing the unit matrix. The computation of $\mathbf{W}_{out}$ just requires the inversion of a squared matrix and some extra matrix multiplications.

C. Performance issues

The proposed method has two main advantages: it always finds the globally optimal solution (no risk to get stuck in a 'poor' local optimum) and it does so in one step. Nevertheless, the size of $\mathbf{X}^{\mathbf{T}}\mathbf{X}$ and the time needed to construct it are both quadratically proportional to the reservoir size, defined as the number of reservoir nodes. The time to perform the required matrix inversion is even cubically proportional to the reservoir size. There may thus be problems of scalability to very large reservoirs like the ones we need for continuous speech recognition [6].

III. Introducing a non-linear readout layer

If the activation function of the readout nodes is no longer linear (see Figure 2), it is no longer possible to find a closed-form solution for the optimal output weights. In that case one has to apply an iterative training procedure such as e.g. logistic regression [8]. Since we do not want to confine ourselves to logistic (sigmoidal) nodes, we applied the general on-line training method described in [9] which was already used successfully before for the training of big Multi-Layer Perceptrons (MLP) for speech recognition. The method updates the weights after every presentation of a new training example. It combines normalization for rescaling the inputs, gradient-descent for updating the weights and line search for controlling the learning speed [10]. The weights on the connection from node k to node i are updated according to

$$\Delta w_{ik} = -\gamma \, r_{ik} \, \frac{\partial c}{\partial w_{ik}} \tag{6}$$

with c being the criterion one intends to minimize. The quantity r_{ik} is a deterministic function of the activation function of node i, the fan-in of that node and the 90-th percentile of the squared output of node k (see [9] for a motivation and more details). In the special case of a fully connected single layer perceptron (the situation considered here), all nodes have the same fan-in, and since the input normalization equalizes the percentiles of the incoming nodes, all r_{ik} become equal to the same r. Nevertheless, thanks to the dependency of r on the normalization of the inputs and the fan-in of the node, it is possible to start the training with the same $\gamma = 0.2$ for every reservoir under test.

A. Changing the criterion to minimize

Once one follows a gradient descent approach, there is no need to stick to a mean squared error function anymore. Any other differentiable error function is acceptable, provided it will give rise to trained networks whose outputs approximate posterior class probabilities. In [11] it is shown that both the mean squared error (MSE) and cross-entropy (CE) do meet that condition. If y_i is the computed and d_i the desired output of node i at some time, the MSE and CE criteria are given by

$$c_{MSE} = \frac{1}{2} \sum_{i=1}^{N_y} (y_i - d_i)^2 \tag{7}$$

and

$$c_{CE} = -\sum_{i=1}^{N_y} \left[d_i \ln(y_i) + (1 - d_i) \ln(1 - y_i) \right] \tag{8}$$

respectively. Obviously, the CE criterion can only be used in combination with outputs that are confined in the interval (0,1).

B. Changing the activation function

Here, the standard non-linear mapping of a node activation a_i to a node output y_i is the logistic function

$$y_i = f(a_i) = \frac{1}{1 + e^{-a_i}} \tag{9}$$

However, since the training circumstances are never ideal, the outputs of the trained network will not be exact posterior probabilities [11]. Hence it is often suggested to use a soft-max node defined by

$$y_i = f(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}} \tag{10}$$

that at least guarantees outputs with a sum of one.

C. Performance issues

Obviously, non-linear nodes call for an iterative training and the training time will be proportional to the number of iterations that are required to converge to a good solution. On the other hand, the time per iteration and the memory requirements are only proportional to the reservoir size and to the number of outputs, and the latter is normally much smaller than the reservoir size. This makes the method more scalable to higher reservoir sizes.

D. Smart initialization of the weights

Since the iterative training normally starts from randomly created weights, it may take a lot of iterations for the training to converge. However, as long as the reservoir size permits us to apply the normal linear readout training, we can use the solution of that method to bootstrap the iterative on-line training. This approach should lead to faster convergence and reduce the chance of getting stuck in a ‘poor’ local optimum.

Simply taking over the weights resulting from the linear training would be an option, but this would lead to non-linear outputs spanning only a part of the output range from 0 to 1. Therefore, we propose to modify the weights so that each linear activation a_i is transformed to a new activation $\hat{a}_i = g\, a_i + a_0$ which has a broader value range. To estimate proper factors g and a_0, we first measure a_+ as the mean (over all training frames) of the linear readout corresponding to the true class of the frame. Likewise, we measure a_- as the mean (over all training frames and false classes) of the linear readouts corresponding to a false class. Based on these values, we introduce P_1 and P_0 as the probabilities of $a_i \geq a_+$ and $a_i \leq a_-$ indicating that i is the true class. If f_{out} represents the activation function of the non-linear readout, then g and a_0 are determined so that

$$\begin{aligned} y &= f_{out}(g\, a_+ + a_0) = P_1 \\ y &= f_{out}(g\, a_- + a_0) = P_0 \end{aligned} \tag{11}$$

In the case of a logistic activation function, one obtains that

$$\begin{aligned} a_0 &= \frac{1}{2} \left(ln\frac{P_1}{1-P_1} + ln\frac{P_0}{1-P_0} + g a_+ + g a_- \right) \\ g &= \frac{1}{a_+ - a_-} \left(ln\frac{P_1}{1-P_1} - ln\frac{P_0}{1-P_0} \right) \end{aligned} \tag{12}$$

Once these transformation parameters are available, it is easy to convert the original weights w_{ik} and w_{0i} to the new weights $\hat{w}_{ik}$ and $\hat{w}_{i0}$ because

$$\begin{aligned} \hat{a}_i &= g\, a_i + a_0 \\ &= \sum_{k=1}^{N} g w_{ik}\, x_k + g w_{i0} + a_0 \\ &= \sum_{k=1}^{N} \hat{w}_{ik}\, x_k + \hat{w}_{i0} \end{aligned} \tag{13}$$

IV. Experimental Evaluation

In the case of speech recognition we are always in the situation that we have a lot of training examples at our disposal, but that the labels of these examples are not always totally reliable. It is under these circumstances that we want to optimize our reservoir systems.

A. Experimental conditions

All training experiments presented in this section aim at maximizing the accuracy of a frame-wise classification (frames are slices of speech) in terms of the basic sounds of American English (called phones). The trained systems will later on be embedded in a phone recognizer that aims to find the phone sequence that was spoken, without having prior knowledge of the length of that sequence.

The data set is the well-known TIMIT database [12]. We perform fully supervised training with the hand-labeled training targets provided with the recordings and we discern 51 phones. The data is split into 2.80 hours of training data (414 speakers), 0.32 hours of development data (48 speakers), and 0.16 hours of test data (24 speakers). There are approximately one million training vectors and the test set is commonly referred to as the core test set.

We use the basic system architecture presented in [6], but in order to save time, we only consider a single-layer implementation (only one reservoir). In the phone recognition experiments, the decoder first converts the readouts to likelihoods, as explained in [5], and then searches for the most likely phone string with the help of a phonetic bi-gram language model. The reservoir inputs are the traditional 39-dimensional MFCC vectors that are used in most speech recognition systems. The network parameters (spectral radius etc.) are set to the values that were also used in [6].

In order to make fair comparisons between methods, the same reservoirs are used in all experiments conducted for a given reservoir size. Classification accuracy is expressed by means of a frame error rate (FER) measured on the development set. Phone recognition accuracy is expressed by means of a phone error rate (PER) measured on the core test set.

B. Regularization

As the training examples are not entirely reliable, they may be self-regulating, and hence not require any explicit regularization like the one introduced by ridge regression. We therefore examined the effect of the regularization parameter ϵ in Equation 4 on the FER measured on the development set. From Table I it is clear that the self-regularization hypotheses holds and that we can use the pure MSE and CE criteria for our experiments with non-linear output nodes as well.

Table I
Frame Error Rate (FER) using Ridge Regression with different regularization terms for two sizes of reservoirs

regularization term	**625**	**2500**
$\epsilon = 10^{-5}$	42.54%	36.79%
$\epsilon = 10^{-7}$	42.27%	36.36%
$\epsilon = 10^{-9}$	42.27%	36.36%
no regularization	42.27%	36.36%

C. Logistic output nodes

Before starting to work with the on-line training of non-linear readout nodes, we first verified whether the on-line training of linear readout yields a solution that has about the same accuracy as the closed-form solution of Equation 5. A test with a reservoir of 625 nodes confirmed that. We therefore assume that differences between systems with linear and non-linear readout nodes will be due to the choice of the node type and not to differences in the training procedure.

In subsequent experiments, we tested logistic nodes in combination with reservoirs of 625, 1250, 2500 and 5000 nodes. The training started from randomly initialized weights and stopped when the FER on the development set did not drop by more than 0.1% over the last 5 iterations. In Figure 3 the FERs obtained for the different systems are compared with those obtained with systems employing linear readouts.

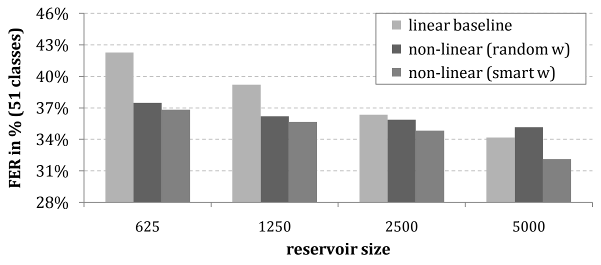


Figure 3. Evaluation of the frame-wise classification error rate (FER) using linear and non-linear readout nodes. In case of non-linear readouts, the initial output weights were either randomly or smartly initialized.

It can be seen that the logistic nodes (with randomly initialized weights) clearly outperform the linear nodes for small reservoir sizes, but the benefit seems to vanish when the systems get larger. For a reservoir size of 5000 the linear nodes are even superior to the non-linear ones.

D. Smart weight initialization

We anticipated that the proposed smart weight initialization might provide a better starting point for the iterative training. We therefore applied the procedure described in Section III with parameters that were retrieved from the outputs of the linear readout nodes on the training data.

Figure 3 (right bars) shows that smart initialization always leads to readouts that outperform the linear ones, even for large reservoir sizes. In terms of training time, Figure 4 shows how the FER evolves in the course of the training. Especially for large reservoir sizes, the improvement is combined with a faster convergence of the training. This complies with the fact that the non-linearities are expected to have a larger impact when the reservoir is smaller, and hence, that for smaller systems the weights derived from the linear solution are further away from the final weights.

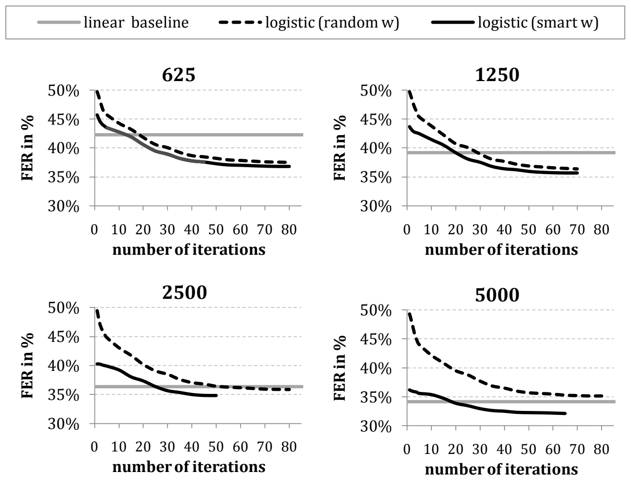


Figure 4. Evaluation of the frame-wise classification error rate (FER) for logistic regression using random and smart weight initialization.

E. Error criteria

Figure 5 shows the classification accuracies that can be reached after smart initialization and subsequent training with MSE or CE as the error criteria. It is clear that cross-entropy persistently outperforms MSE for all reservoir sizes. We therefore maintain CE as the error criterion in the forthcoming experiments. Another finding is that CE-based training always takes a few more iterations than MSE-based training to converge. This is owed to the larger mismatch between the MSE criterion that is involved in the weight initialization process and the CE criterion that is used during training.

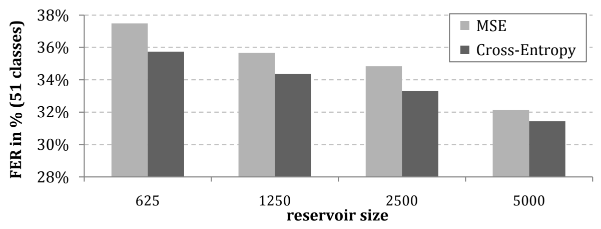


Figure 5. Evaluation of the frame-wise classification error rate (FER) using MSE or Cross-Entropy as cost-function during the logistic regression.

F. Activation functions

In a last training experiment, the classification accuracies reached after training with logistic and soft-max nodes was evaluated. The error criterion was CE and the weights were randomly initialized here. The latter was done in order to avoid any positive bias of the on-line training towards the logistic function. We investigated systems with 625, 1250 and 2500 nodes and found that the soft-max nodes yield an inferior classification performance (0.5% to 1% absolute) in all examined cases.

G. Phone recognition

Since we are not interested in frame-wise classification but in true phone recognition, we have also assessed the phone error rates (PER) of various systems on the core test set (see Table II). The PER measures substitutions, insertions and deletions between the recognized and the correct phone sequence. As it is done traditionally, errors are obtained after having mapped all recognized and correct phones to 39 phone classes first.

Table II
PHONE ERROR RATES (PER) FOR DIFFERENT INVESTIGATED SYSTEMS

model			reservoir size					
type	cost	w_{out}	625	1250	2500	5k	10k	20k
linear	**MSE**	-	**41.6%**	**38.2%**	**34.9%**	**32.4%**	**30.5%**	**29.4%**
logistic	MSE	random	36.7%	35.3%	34.8%	33.9%	-	-
logistic	MSE	smart	36.0%	34.0%	33.7%	30.2%	-	-
logistic	CE	random	34.0%	32.6%	31.7%	31.1%	-	-
logistic	**CE**	**smart**	**33.9%**	**31.9%**	**31.4%**	**29.2%**	29.6%	-
soft-max	CE	random	35.2%	34.1%	31.8%	-	-	-

The PERs correlate very strongly with the FERs. They confirm that CE outperforms MSE as an error criterion and that smart initialization leads to higher recognition rates. The table also confirms that smart initialization is less effective when CE is used and that the soft-max function does not help for sequence decoding.

For systems of the same size we observe that the non-linear system outperforms the linear system by 10 – 18% relative. On the other hand, the non-linear readout accuracy saturates at the same value as the linear readout accuracy. Hence, the only benefit of nonlinear readouts seems to be that they can achieve the maximum attainable accuracy at a much lower reservoir size. In the next section we discuss the positive consequences of this.

H. Needed Resources

From Table II it follows that non-linear systems with N reservoir nodes compete well with linear systems with $4N$ reservoir nodes. Taking this rule into account, we can analyze the training time as a function of the attainable accuracy so to speak. Table III then reveals that the iterative

Table III
TRAINING TIME (ON A SINGLE CORE 3GHZ CPU) AS A FUNCTION OF ACCURACY FOR LINEAR AND NON-LINEAR READOUTS. FOR NON-LINEAR NODES WE DISTINGUISH BETWEEN RANDOM (NO BRACKETS) AND SMART WEIGHT INITIALIZATION (BRACKETS)

needed nodes		training time (in h)	
non-linear	linear	non-linear	linear
1250	5000	17 [18]	3
2500	10000	35 [36]	13
5000	20000	70 [73]	52
7000	**28000**	**97 [103]**	**102**
10000	40000	140 [153]	214
20000	80000	280 [332]	856
40000	160000	560 [764]	3424

training of the non-linear readouts (without or with smart initialization) takes less time than the solution of the linear regression problem as soon as the reservoir size exceeds a threshold of 7000 nodes[1]. Hence, if we want to develop reservoir systems on a much larger database than TIMIT, the accuracy will most probably saturate only for reservoir sizes above that threshold, where the non-linear systems show an advantage.

V. CONCLUSIONS

We have shown that non-linear readout nodes in combination with an on-line learning algorithm are an interesting alternative to the linear nodes used in traditional Reservoir Computing. The use of logistic nodes in combination with a Cross-Entropy error criterion leads to a significant improvement (10 – 18% relative) attainable with a predefined reservoir size.

Non-linear readout nodes seem to be more powerful classifiers than the linear nodes, especially when the size of the reservoir is constrained. Only when the reservoir is made big enough, linear and non-linear classifiers can reach the same level of accuracy. However, the computation time that must be spent to reach this level can be quite different for the two cases. The main benefit of the non-linear solution seems to be that it requires less reservoir nodes to attain a certain accuracy. Hence, it permits to reduce the computation load during real operation and it offers better perspectives for the development of speech recognition systems on much larger benchmarks than the TIMIT benchmark that was used here. Systems with linear readouts and a reservoir of e.g. 40,000 nodes cannot be solved with linear regression anymore, due to the very large squared matrix that has to be inverted. But competitive systems with non-linear readout nodes stay well within reach.

One of our priorities for the future is to prove our hypotheses that non-linear readouts will result in better recognition accuracy on larger benchmarks.

[1]The CPU times for the very large reservoir sizes are no actual measurements but extrapolations that could be made on the basis of our knowledge of how the computational load depends on the reservoir size.

ACKNOWLEDGMENT

The work presented in this paper is funded by the EC FP7 project ORGANIC (FP7-231267).

REFERENCES

[1] A. Robinson, "An application of recurrent neural nets to phone probability estimation," *IEEE Trans. on Neural Networks*, vol. 5, pp. 298–305, 1994.

[2] H. Jaeger, "Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the echo state network approach," German National Research Center for Information Technology, Tech. Rep., 2002. [Online]. Available: http://www.faculty.jacobs-university.de/hjaeger

[3] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, pp. 391–403, 2007.

[4] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural Networks*, vol. 20, pp. 335–352, 2007.

[5] A. Jalalvand, F. Triefenbach, D. Verstraeten, and J.-P. Martens, "Connected digit recognition by means of reservoir computing," in *Procs. of INTERSPEECH*, 2011, pp. 1725–1728.

[6] F. Triefenbach, A. Jalalvand, B. Schrauwen, and J.-P. Martens, "Phoneme recognition with large hierarchical reservoirs," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2010, pp. 2307–2315.

[7] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," *The American Statistician*, vol. 29, no. 1, pp. pp. 3–20, 1975.

[8] M. Schumacher, R. Rossner, and W. Vach, "Neural networks and logistic regression: Part i," *Computational Statistics and Data Analysis*, vol. 21, no. 6, pp. 661–682, 1996.

[9] J.-P. Martens and N. Weymaere, "An equalized error backpropagation algorithm for the on-line training of multilayer perceptrons," *IEEE Trans. on Neural Networks*, vol. 13, pp. 532–541, 2002.

[10] R. Golden, *Mathematical methods for neural network design and analysis*. Cambridge, MIT Press, 1996.

[11] M. Richard and R. Lippmann, "Neural net classifiers estimate posterior probabilities," *Neural Computation*, vol. 3(4), pp. 461–483, 1991.

[12] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren, "The DARPA TIMIT acoustic-phonetic continuous speech corpus cd-rom," National Institute of Standards and Technology, Tech. Rep., 1993.