

# Improving Recurrent Neural Network Performance Using Transfer Entropy

Oliver Obst<sup>1,2</sup>, Joschka Boedecker<sup>3,4</sup>, and Minoru Asada<sup>3,4</sup>

<sup>1</sup> CSIRO ICT Centre, Adaptive Systems, P.O. Box 76, Epping, NSW 1710, Australia

<sup>2</sup> School of Information Technologies, The University of Sydney, NSW 2006, Australia

<sup>3</sup> Department of Adaptive Machine Systems, Osaka University, Suita, Osaka, Japan

<sup>4</sup> JST ERATO Asada Synergistic Intelligence Project, Suita, Osaka, Japan

Oliver.Obst@csiro.au

joschka.boedecker@ams.eng.osaka-u.ac.jp

asada@ams.eng.osaka-u.ac.jp

**Abstract.** Reservoir computing approaches have been successfully applied to a variety of tasks. An inherent problem of these approaches, is, however, their variation in performance due to fixed random initialisation of the reservoir. Self-organised approaches like intrinsic plasticity have been applied to improve reservoir quality, but do not take the task of the system into account. We present an approach to improve the hidden layer of recurrent neural networks, guided by the learning goal of the system. Our reservoir adaptation optimises the information transfer at each individual unit, dependent on properties of the information transfer between input and output of the system. Using synthetic data, we show that this reservoir adaptation improves the performance of offline echo state learning and Recursive Least Squares Online Learning.

**Keywords:** Machine learning, recurrent neural network, information theory, reservoir computing, guided self-organisation.

## 1 Introduction

Reservoir Computing (RC) is a recent paradigm in the field of recurrent neural networks (for a recent overview, see [1]). RC computing approaches have been employed as mathematical models for generic neural microcircuits, to investigate and explain computations in neocortical columns (see e.g. [2]). A key element of reservoir computing approaches is the randomly constructed, fixed hidden layer – typically, only connections to output units are trained. Despite their impressive performance for some tasks (e.g. [3]), their fixed random connectivity can lead to significant variation in performance [4]. To address this issue, approaches like Intrinsic Plasticity (IP) [5, 6] can help to improve randomly constructed reservoirs. IP is based on the idea to maximise available information at each internal unit in a self-organised way by changing the behaviour of individual units. This is contrast to, for example, Hebbian learning [7], which strengthens connections between two units if their firing patterns are temporally correlated.

Both adaptation of individual units as well as adaptation of connections are phenomena that occur in biological units.

IP learning has been used as an approach to optimise reservoir encoding specific to the input of the network [6]. It is, however, *only* dependent on the input data, and does not take the desired output of the system into account, i.e., it is not guaranteed to lead to optimised performance with respect to the learning task of the network [4]. Ideally, we would like to retain the principle of a self-organised approach to optimise reservoirs, but to guide self-organisation [8] based on the overall learning goal.

The approach presented in this paper for the first time leads to a method that optimises the information transfer at each individual unit, dependent on properties of the information transfer between input and output of the system. The optimisation is achieved by tuning self-recurrent connections, i.e., the means to achieve this optimisation can be viewed as a compromise between Hebbian and IP learning. Using synthetic data, we show that this reservoir adaptation improves the performance of offline echo state learning, and is also suitable for online learning approaches like backpropagation-decorrelation learning [9] or recursive least squares (RLS, see e.g. [10]).

## 2 Echo State Networks

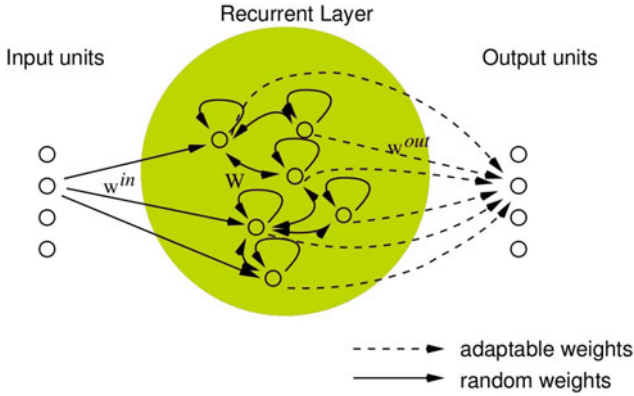
ESN provide a specific architecture and a training procedure that aims to solve the problem of slow convergence [11, 3] of earlier recurrent neural network training algorithms. ESN are normally used with a discrete-time model, i.e. the network dynamics are defined for discrete time-steps  $t$ , and they consist of inputs, a recurrently connected hidden layer (also called *reservoir*) and an output layer (see Fig. 1).

We denote the activations of units in the individual layers at time  $t$  by  $\mathbf{u}_t$ ,  $\mathbf{x}_t$ , and  $\mathbf{o}_t$  for the inputs, the hidden layer and the output layer, respectively. The matrices  $\mathbf{w}^{\text{in}}$ ,  $\mathbf{W}$ ,  $\mathbf{w}^{\text{out}}$  specify the respective synaptic connection weights. Using  $f(x) = \tanh x$  as output nonlinearity for all hidden layer units, the network dynamics is defined as:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{W}\mathbf{x}_{t-1} + \mathbf{w}^{\text{in}}\mathbf{u}_t) \quad (1)$$

$$\mathbf{o}_t = \mathbf{w}^{\text{out}}\mathbf{x}_t \quad (2)$$

The main differences of ESN to traditional recurrent network approaches are the setup of the connection weights and the training procedure. To construct an ESN, units in the input layer and the hidden layer are connected randomly. Only connections between the hidden layer and the output units are trained, usually with a supervised offline learning approach using linear regression. Here, the output weights  $\mathbf{w}^{\text{out}}$  are calculated using the collection of desired output states  $\mathbf{D}$ , and the pseudoinverse of a matrix  $\mathbf{S}$  collecting the states of the system over a number of steps as  $\mathbf{w}^{\text{out}} = \mathbf{S}^\dagger\mathbf{D}$  (see [11] for details). An online learning procedure, like RLS, adapts the output weights while training input is fed into



**Fig. 1.** In echo state networks, only output weights (represented by dashed lines) are trained, all other connections are setup randomly and remain fixed. The recurrent layer is also called a *reservoir*, analogously to a liquid, which has fading memory properties in response to perturbations (like e.g. ripples caused by a rock thrown into a pond).

the network, i.e., no states need to be collected. The RLS update rule can be described with the following set of equations:

$$\alpha_t = \mathbf{d}_t - \mathbf{w}_{t-1}^{\text{out}} \cdot \mathbf{x}_t, \quad (3)$$

$$\mathbf{g}_t = \mathbf{p}_{t-1} \cdot \mathbf{x}_t / (\lambda + \mathbf{x}_t^T \cdot \mathbf{p}_{t-1} \cdot \mathbf{x}_t), \quad (4)$$

$$\mathbf{p}_t = (\mathbf{p}_{t-1} - \mathbf{g}_t \cdot \mathbf{x}_t^T \cdot \mathbf{p}_{t-1}) / \lambda, \quad (5)$$

$$\mathbf{w}_t^{\text{out}} = \mathbf{w}_t^{\text{out}} + (\alpha_t \cdot \mathbf{g}_t^T), \quad (6)$$

where  $\alpha_t$  represents the *a priori error* vector between desired output  $\mathbf{d}_t$  and current input,  $\mathbf{p}_t$  the inverse of the autocorrelation, and  $\lambda$  is close to 1 and is an exponential forgetting factor. RLS has been applied to ESN learning in [12].

Even though the reservoir weights are randomly initialised and remain fixed, these connections cannot be completely random; they are typically designed to have the *echo state property*. The definition of the echo state property has been outlined in [11] and is summarised in the following section.

## 2.1 The Echo State Property

The Echo State Property is reflected in the following definition. In simple terms, the system has echo state property if different initial states converge to each other for all inputs. Consider a time-discrete recursive function:

$$\mathbf{x}_{t+1} = \mathbf{F}(\mathbf{x}_t, \mathbf{u}_{t+1}) \quad (7)$$

that is defined at least on a compact sub-area of the vector-space  $\mathbf{x} \in R^n$ , with  $n$  the number of internal units. The  $\mathbf{x}_t$  are to be interpreted as internal states and  $\mathbf{u}_t$  is some external input sequence, i.e. the stimulus.

**Definition 1.** Assume an infinite stimulus sequence:  $\bar{\mathbf{u}}^\infty = \mathbf{u}_0, \mathbf{u}_1, \dots$  and two random initial internal states of the system  $\mathbf{x}_0$  and  $\mathbf{y}_0$ . From both initial states  $\mathbf{x}_0$  and  $\mathbf{y}_0$  the sequences  $\bar{\mathbf{x}}^\infty = \mathbf{x}_0, \mathbf{x}_1, \dots$  and  $\bar{\mathbf{y}}^\infty = \mathbf{y}_0, \mathbf{y}_1, \dots$  can be derived from the update equation Eq. (7) for  $\mathbf{x}_{t+1}$  and  $\mathbf{y}_{t+1}$ . The system  $F(\cdot)$  will have the echo state property if, independently of the set  $\mathbf{u}_t$ , for any  $(\mathbf{x}_0, \mathbf{y}_0)$  and all real values  $\epsilon > 0$ , there exists a  $\delta(\epsilon)$  for which  $d(\mathbf{x}_t, \mathbf{y}_t) \leq \epsilon$  for all  $t \geq \delta(\epsilon)$ , where  $d$  is a square Euclidean metric.

### 3 Transfer Entropy

To improve the reservoir based on the learning goal, we are interested in detecting the characteristics of the information transfer between input and desired output of the system. Transfer Entropy [13] is an information-theoretic measure for the information provided by a source about the next state of the destination which was not already contained in its own history. It is similar to mutual information [see e.g. 2], but asymmetric (i.e. directed), and takes the dynamics of information transfer into account. The transfer entropy from a source node  $Y$  to a destination node  $X$  is the mutual information between previous  $l$  states of the source  $y_n^{(l)}$  and the next state of the destination  $x_{n+1}$ ,

$$T_{Y \rightarrow X} = \lim_{k, l \rightarrow \infty} \sum_{\mathbf{u}_n} p(x_{n+1}, x^{(k)}, y_n^{(l)}) \log_2 \frac{p(x_{n+1} | x_n^{(k)}, y_n^{(l)})}{p(x_{n+1} | x_n^{(k)})}. \quad (8)$$

where  $\mathbf{u}_n$  is the state transition tuple  $(x_{n+1}, x^{(k)}, y_n^{(l)})$ .

For our purposes,  $T_{Y \rightarrow X}(k, l)$  represents finite  $k, l$  approximation.

### 4 Reservoir Dynamics

In the following, we consider the case where we have an one-dimensional input vector  $\mathbf{u}$ . The learning goal for our system is a one step-ahead prediction of an one-dimensional output vector  $\mathbf{v}$ . Departing from the usual reservoir dynamics described above, we use

$$\mathbf{x}(k+1) = \text{diag}(\mathbf{a})\mathbf{W}\mathbf{y}(k) + (\mathbf{I} - \text{diag}(\mathbf{a}))\mathbf{y}(k) + \mathbf{w}^{\text{in}}\mathbf{u}(k) \quad (9)$$

$$\mathbf{y}(k+1) = \mathbf{f}(\mathbf{x}(k+1)), \quad (10)$$

where  $x_i$ ,  $i = 1, \dots, N$  are the neural activations,  $\mathbf{W}$  is the  $N \times N$  reservoir weight matrix,  $\mathbf{w}^{\text{in}}$  the input weight,  $\mathbf{a} = (\alpha_1, \dots, \alpha_N)^T$  a vector of local decay factors,  $\mathbf{I}$  is the identity matrix, and  $k$  the discrete time step. In this work, we use  $\mathbf{f}(\mathbf{x}) = \tanh(\mathbf{x})$ .

The  $\alpha_i$  represent a decay factor, or coupling of a unit's previous state with the current state; they are computed as:

$$\alpha_i = \frac{2}{1 + m_i}, \quad (11)$$

where  $m_i$  represents the memory length of unit  $i$  ( $m_i \in \{1, 2, 3, \dots\}$ ). All memory lengths are initialised to  $m_i = 1$ , so that  $\alpha_i = 1$ , i.e. the reservoir has the usual update rule. Increasing individual  $m_i$  during an adaptation will increase the influence of a unit's past states on its current state.

## 5 Adaptation of Information Transfer

Adaptation of the reservoir to the learning goal introduces two extra steps to the learning procedure. In a first step, we determine the required history size  $l$  to maximise the information transfer from input  $\mathbf{u}$  to output  $\mathbf{v}$ , i.e. a first idea may be to look for a value

$$l_{\max} = \arg \max_l T_{\mathbf{u} \rightarrow \mathbf{v}}(1, l) .$$

Using increasingly larger history sizes may, however, always increase the transfer entropy (by possibly smaller and smaller values). To optimise the information transfer, we will instead be looking for the smallest value  $\hat{l}$  that does not increase the transfer entropy  $T_{\mathbf{u} \rightarrow \mathbf{v}}(1, \hat{l} - 1)$  by more than a threshold  $\epsilon$ , i.e.

$$T_{\mathbf{u} \rightarrow \mathbf{v}}(1, \hat{l} + 1) \leq T_{\mathbf{u} \rightarrow \mathbf{v}}(1, \hat{l}) + \epsilon \quad \mathbf{and} \quad (12)$$

$$T_{\mathbf{u} \rightarrow \mathbf{v}}(1, l) > T_{\mathbf{u} \rightarrow \mathbf{v}}(1, l - 1) + \epsilon \quad \mathbf{for\ all\ } l < \hat{l} . \quad (13)$$

From this first step, we learn the contribution of the size of the input history to the desired output (the learning goal of the system): some input-output pairs may require a larger memory of the input history to be informative about the next output state, other outputs may be more dynamic, and be dependent on the current input state only.

We take this information into the second step, which consists of a pre-training of the reservoir. Here, the local couplings of the reservoir units are adapted so that the transfer entropy from the input of each unit to its respective output is optimised *for the particular input history length  $\hat{l}$* . The idea behind this step is to locally adjust the memory at each unit to approximate the required memory for the global task of the system. Pre-training is done in epochs of length  $\ell$  over the training data. Over each epoch  $\theta$ , we compute, for each unit  $i$ , the transfer entropy from activations  $x_i^{(\ell)}$  to output  $y_i^{(\ell)}$ :

$$te_i^\theta = T_{x_i^{(\ell)} \rightarrow y_i^{(\ell)}}(1, \hat{l}) . \quad (14)$$

If the information transfer during the current epoch  $\theta$  exceeds the information transfer during the past epoch by a threshold (i.e.,  $te_i^\theta > te_i^{\theta-1} + \epsilon$ ), the local memory length  $m_i$  is increased by one. Likewise, if  $te_i^\theta < te_i^{\theta-1} - \epsilon$ , the local memory length is decreased by one, down to a minimum of 1.

After each epoch, all  $m_i$  and  $\alpha_i$  are adapted according to this rule, and used to compute activations over the next epoch. Once the training data is exhausted,

pre-training of the reservoir is finished and the  $\alpha_i$  are fixed. For the subsequent training we compute the output weights by linear regression with data as used in the pre-training. In additional experiments, we use RLS online learning, where adaptation and training of output weights were run in the same loop.

## 6 Experimental Results

We tested our method using a one-step ahead prediction of unidirectionally coupled maps, and a one-step ahead prediction of the Mackey-Glass time series.

### 6.1 Prediction of Autoregressive Coupled Processes

As first experiments we studied our approach using a one-step ahead prediction of two unidirectionally coupled autoregressive processes:

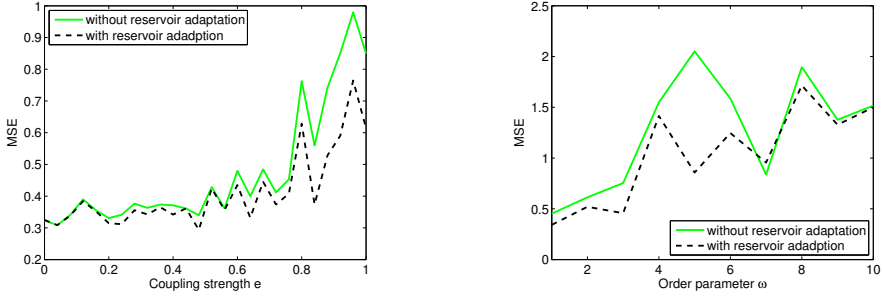
$$u_{t+1} = 0.7 u_t + 0.7 \cos(0.3t) + n_t^x(0, \sigma^2) \quad \text{and} \quad (15)$$

$$v_{t+1} = 0.7 v_t + e u_{t-\omega+1} + n_t^y(0, \sigma^2) , \quad (16)$$

where the parameter  $e \in [0, 1]$  regulates the coupling strength,  $\omega \in \{0, 1, 2, \dots\}$  an order parameter, and  $n_t^x(0, \sigma^2)$  and  $n_t^y(0, \sigma^2)$  are independent Gaussian random processes with zero mean and standard deviation  $\sigma = 0.4$ . For each trial, we generated time series  $\mathbf{u}$  and  $\mathbf{v}$  (random initial conditions; time series divided into 10000 values for training and 1200 values for testing; the first 200 values of both training and testing were used to prime the reservoir), where the task of our system was a one-step ahead prediction of  $\mathbf{v}$  using  $\mathbf{u}$ . The reservoir was initialised using a random, sparse recurrent weight matrix ( $|\lambda| = 0.95$ ), with 40 internal units. Figure 2 (a) displays the mean square errors of the prediction over the test data for different coupling strengths and fixed  $\omega = 0$  for both echo state learning with and without adaptation of information transfer in the reservoir. All values are averaged over 50 trials; for each individual trial the same reservoir and time series have been used once with and without adaptation. The prediction using the reservoir adaptation is better over almost the entire range of  $e$ , with the improvement becoming more significant as the influence of the input time series becomes larger. Figure 2 (b) is a plot of the mean square error for different  $\omega$  using a fixed coupling of  $e = 0.75$ . In all but one cases the reservoir adaptation improves results.

### 6.2 Prediction of Mackey-Glass Time Series

A further experiment was prediction of the widely used Mackey-Glass time series (see e.g. [11, 14, 6]) with parameter  $\tau$  set to 17. The first task using this time series was again a one-step ahead prediction using a reservoir size of 40 units. For this task, the transfer entropy between input and output time series is maximised already for smaller values of  $\ell$  compared to our first experiment ( $\ell$  was typically around 2 for Mackey-Glass one-step ahead prediction), i.e., the information used from the previous state to predict the next state is already quite high. The



**Fig. 2.** (a) Left: mean square errors of the prediction over the test data for different coupling strengths and fixed  $\omega = 0$ . (b) Right: mean square error for different  $\omega$  using a fixed coupling of  $e = 0.75$ . Reported results are averages over 50 runs.

reservoir adaptation lead to an average improvement of the MSE (averaged over 50 runs) from  $0.4530 \cdot 10^{-6}$  to  $0.0751 \cdot 10^{-6}$ . Individually, in 48 of the 50 runs, the same reservoir performed better with adaptation than without adaptation.

Instead of offline learning, we also used RLS in the same loop with our reservoir adaptation. To less consider data from earlier stages of the adaptation, we used a forgetting factor  $\lambda = 0.995$ . Again, the adaptation improved performance, from  $9.1 \cdot 10^{-6}$  to  $7.2 \cdot 10^{-6}$ ; a fine-tuning of  $\lambda$  may further improve the results.

## 7 Conclusions

We presented an information-theoretic approach to reservoir optimisation. Our approach uses a local adaptation of a units internal state, based on properties of the information transfer between input and desired output of the system. The approach has shown to improve performance in conjunction with offline echo-state regression, as well as with RLS online learning. In our experiments we have used only a small number of internal units – our goal was to show the capability of our approach compared to standard echo state learning. In first additional experiments (not reported here), we have shown that for a larger number of units our adaptation leads to an even larger improvement compared to echo state learning without adaptation. A further investigation of statistical properties of coding in the reservoir obtained by our adaptation may provide useful insights. Moreover, other information-theoretic measures such as the active information storage [15] may be useful to further improve the local adaptation rule.

**Acknowledgments.** The Authors thank the Australian Commonwealth Scientific and Research Organization’s (CSIRO) Advanced Scientific Computing group for access to high performance computing resources used for simulation and analysis.

## References

- [1] Lukosevicius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. *Computer Science Review* 3(3), 127–149 (2009)
- [2] Maass, W., Natschläger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14(11), 2531–2560 (2002)
- [3] Jaeger, H., Haas, H.: Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science* 304(5667), 78–80 (2004)
- [4] Boedecker, J., Obst, O., Mayer, N.M., Asada, M.: Initialization and self-organized optimization of recurrent neural network connectivity. *HFSP Journal* 3(5), 340–349 (2009)
- [5] Triesch, J.: A gradient rule for the plasticity of a neuron’s intrinsic excitability. In: Duch, W., Kacprzyk, J., Oja, E., Zadrozny, S. (eds.) *ICANN 2005. LNCS*, vol. 3696, pp. 65–70. Springer, Heidelberg (2005)
- [6] Steil, J.J.: Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Networks* 20(3), 353–364 (2007)
- [7] Hebb, D.O.: *The organization of behavior: a neuropsychological theory*. Lawrence Erlbaum Associates, Mahwah (1949)
- [8] Prokopenko, M.: Guided self-organization. *HFSP Journal* 3(5), 287–289 (2009)
- [9] Steil, J.J.: Backpropagation-decorrelation: Recurrent learning with  $O(N)$  complexity. In: *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, vol. 1, pp. 843–848 (2004)
- [10] Hayes, M.H.: Chapter 9.4 Recursive Least Squares. In: *Statistical Digital Signal Processing and Modeling*. Wiley, Chichester (1996)
- [11] Jaeger, H.: The “echo state” approach to analysing and training recurrent neural networks. Technical Report 148, GMD – German National Research Institute for Computer Science (2001)
- [12] Jaeger, H.: Adaptive nonlinear systems identification with echo state networks. In: *Advances in Neural Information Processing Systems*, pp. 609–615 (2003)
- [13] Schreiber, T.: Measuring information transfer. *Physical Review Letters* 85(2), 461–464 (2000)
- [14] Hajnal, M., Lőrincz, A.: Critical echo state networks. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) *ICANN 2006. LNCS*, vol. 4131, pp. 658–667. Springer, Heidelberg (2006)
- [15] Lizier, J.T., Prokopenko, M., Zomaya, A.Y.: Detecting non-trivial computation in complex dynamics. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) *ECAL 2007. LNCS (LNAI)*, vol. 4648, pp. 895–904. Springer, Heidelberg (2007)