Improving reservoirs using Intrinsic Plasticity

Benjamin Schrauwen [a,*] Marion Wardermann [a]
David Verstraeten [a] Jochen J. Steil [b] Dirk Stroobandt [a]

[a] *Department of Electronics and Information Systems, Ghent University*
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium

[b] *Neuroinformatics Group, Bielefeld University*
Universitätsstraße 25, 33615 Bielefeld, Germany

Abstract

The benefits of using Intrinsic Plasticity (IP), an unsupervised, local, biologically inspired adaptation rule that tunes the probability density of a neuron's output towards an exponential distribution – thereby realizing an information maximization – have already been demonstrated. In this work, we extend the ideas of this adaptation method to a more commonly used nonlinearity and a Gaussian output distribution. After deriving the learning rules, we show the effects of the bounded output of the transfer function on the moments of the actual output distribution. This allows us to show that the rule converges to the expected distributions, even in random recurrent networks. The IP rule is evaluated in a Reservoir Computing setting, which is a temporal processing technique which uses random, un-trained recurrent networks as excitable media, where the network's state is fed to a linear regressor used to calculate the desired output. We present an experimental comparison of the different IP rules on three benchmark tasks with different characteristics. Furthermore, we show that this unsupervised reservoir adaptation is able to adapt networks with very constrained topologies, such as a 1D lattice which generally shows quite unsuitable dynamic behavior, to a reservoir that can be used to solve complex tasks. We clearly demonstrate that IP is able to make Reservoir Computing more robust: the internal dynamics can autonomously tune themselves – irrespective of initial weights or input scaling – to the dynamic regime which is optimal for a given task.

Key words:
Reservoir Computing, Intrinsic Plasticity, Information maximization

* Corresponding author.
Email address: Benjamin.Schrauwen@UGent.be (Benjamin Schrauwen).
URL: http://snn.elis.ugent.be (Benjamin Schrauwen).

1 Introduction

In machine learning, feed-forward structures like artificial neural networks or kernel methods have been studied extensively for the processing of non-temporal input. These architectures are well understood in many respects because of their feed-forward and non-dynamic nature. Many real-world tasks, however, are temporal in nature, such as prediction (weather, dynamic systems, financial data), system control or identification, adaptive filtering, noise reduction, gait generation, planning or movement control in robotics, and vision and speech (recognition, processing, production).

Feed-forward approaches usually incorporate the temporal domain by means of extending the input to incorporate a finite time window. This technique is motivated by the famous Takens theorem [1], which states that the (hidden) state of the dynamical system can be reconstructed using an adequate delayed embedding. This explicit embedding converts the temporal problem into a spatial one. Disadvantages of this approach are the artificially introduced time horizon, the need for many parameters when a long delay is introduced, and the artificial way to represent time in the spatial domain.

A possible solution to this problem is adding recurrent connections to the feed-forward architecture and to transfer the respective learning algorithms into the spatio-temporal domain. Then the goal is to train all the weights in a fully (or sparsely) connected recurrent neural network based on a standard error minimization approach. This is the purpose of the extension of back-propagation to back-propagation-through-time for recurrent networks by Werbos [2,3] (which was later rediscovered in [4]).

Theoretically, RNNs are very powerful tools for solving complex temporal machine learning tasks. Nonetheless, the application of these rules to real-world problems is not always feasible due to the high computational training costs and slow convergence [5] and while it is in principle possible to achieve state-of-the-art performance, this is often suitable only for experts in the field [6]. Another significant and principal problem for recurrent learning algorithms is the so-called *fading gradient,* which means that the error gradient vanishes or gets distorted when it is propagated many time steps backward, such that only short time series examples are usable for training. One possible solution to this is a specially constructed Long Short Term Memory (LSTM) architecture [7], which nonetheless does not always outperform time delayed neural networks.

A way to train recurrent structures while reducing the computational burden and circumventing the fading gradient problem has independently been discovered by Jaeger [8] as the Echo State Network (ESN) and Maass [9] as the Liquid State Machine (LSM). The idea is to train only parts of the network

using a simple classification/regression technique and leave large parts of the network fixed. We refer to such an approach as Reservoir Computing (RC). Recent advances in the overall field have also been published in [10].

The Echo State Network consists of a randomly connected recurrent network of analog neurons – the reservoir – that is driven by a (one- or multi-dimensional) temporal input signal. On the output level, the activations of the entire network are treated as high-dimensional spatio-temporal input features for a linear classification/regression algorithm – the linear readout. The ESN was introduced as an improved way to use the computational power of RNNs without training of the internal weights. From a formal viewpoint, the reservoir acts as a complex nonlinear dynamic filter that transforms the input signals using a high-dimensional temporal mapping, not unlike the operation of an explicit, temporal kernel function. It is even possible to solve several classification tasks on a single input signal simultaneously by adding multiple readouts to one reservoir.

The Liquid State Machine by Maass [9] was originally presented as a general framework to perform real-time computation on temporal signals, but most of its realizations are based on an abstract cortical micro-column model. Here a 3D structured locally connected network of spiking neurons is randomly created using biologically inspired parameters and excited by external input spike trains. The responses from all neurons are projected to the next cortical layer where the actual training is performed. This projection is usually modeled using a simple linear regression function.

A different line of research by Steil [11,12] showed that the on-line version of a current state-of-the-art learning rule for RNNs (Athya-Parlos recurrent learning [13]), which is derived as a classical error minimization method, leads to similar weight adaptation dynamics as were proposed by Jaeger and Maass: the output weights are trained and adapt quickly while internal weights are changing on a much slower time-scale and in a highly coupled way. These changes can even be ignored if the weights are well scaled initially. This shows that RC is both an attractive idea because of the simplicity of its training scheme and theoretically motivated from an approximation to an error minimizing learning algorithm.

Several successful applications of reservoir computing to both synthetic data and real world engineering applications have been reported in the literature. The former include dynamic pattern classification [14], autonomous sine generation [8] or the computation of highly nonlinear functions on the instantaneous rates of spike trains [15]. In robotics, LSMs have been used to control a simulated robot arm [16], to model an existing robot controller [17], to perform object tracking and motion prediction [18,19], event detection [20,21] or several applications in the Robocup competitions (mostly motor control)

[22,23,24]. ESNs have been used in the context of reinforcement learning [25]. Also, applications in the field of Digital Signal Processing (DSP) have been quite successful, such as speech recognition [26,27,28] or noise modeling [29]. In [30], an application in Brain-Machine interfacing is presented. And finally, the use of reservoirs for chaotic time series generation and prediction have been reported in [14,31,32,33]. In many area's such as chaotic time series prediction and isolated digit recognition, RC techniques already outperform state-of-the-art approaches. A striking case is demonstrated in [29] where it is possible to predict the Mackey-Glass chaotic time series several orders of a magnitude farther in time than with classical techniques.

Various ways of constructing reservoir topologies and weight matrices have been described (see [34] for a brief overview), but for ESNs, one usually creates a network with a certain sparsity and assigns random weights drawn from a normal or uniform distribution or from a discrete set. Next, the weight matrix is globally scaled to set the spectral radius (largest absolute eigenvalue) to a certain value. While Jaeger proposes an optimal spectral radius of around 0.9 for certain small-scale problems, this is not generally applicable [32,35]. Optimization of a reservoir for applications is typically based on experience and heuristics and partly on a brute-force search of the parameter space. Moreover, the variance of the performance across different reservoirs with the same spectral radius is still quite substantial, which is clearly undesirable. Obviously, a computationally simple way to adapt the reservoirs to the task at hand without requiring a full parameter sweep or hand tuning based on experience would be welcome.

In this line, it was recently shown [36,37,38] that the performance of off-line and on-line learning for ESN networks with fermi transfer functions can be improved by using an unsupervised and local adaptation rule based on information maximization, called Intrinsic Plasticity (IP). This rule was first introduced in [39] as a formal model of processes known in neurobiology as homeostatic plasticity. Since this kind of plasticity is quasi omnipresent in biological neurons it is natural to investigate its formal counterpart in combination with standard artificial network learning algorithms. In this context, Triesch has also shown that in combination with Hebbian learning IP can drastically change the behavior of Hebbian networks towards finding heavy-tail directions in arbitrary input distributions [40]. The interplay between these two unsupervised adaptation rules was further investigated in [41]. There, it was shown that the combination of IP and STDP enhances the robustness of the network against small perturbations and aids the networks in discovering temporal patterns present in the input signals. The results from that work suggest that a fundamental underlying link between IP, STDP and unsupervised information maximization exists, and that these rules operating strictly on the local neuron-level succeed in steering the dynamics of the entire network towards a computationally desirable regime.

In this contribution, we further investigate different versions of IP learning extending and generalizing the idea. Whereas previous work on IP has focused on the fermi transfer function and an exponential target distribution, we derive the corresponding formalism here for the hyperbolic tangent transfer function and a Gaussian output distribution. We also derive a formalism to compute the target means and variances the IP learning rule is supposed to converge to. Simulations show that in practice these targets are reached surprisingly well despite the interference between neurons introduced by the recurrency in the network.

The simple, local IP rule effectively makes reservoirs significantly more robust: It empowers the reservoirs with the ability to autonomously and robustly adapt their internal dynamics, irrespective of initial weight setting, input scaling or topology, to a dynamic regime which is suited for the given task. The rule is purely input driven, and adapts the reservoir in an unsupervised way.

This work is organized as follows. We first derive the Intrinsic Plasticity learning rule and show how it performs in practice in Section 2. Next, in Section 3, we show the effect of using the different IP rules to reservoir computing on three benchmark applications of different types. Section 4 demonstrates that IP can also help to use very sparse, structured reservoir topologies such as 1D lattices. We conclude and present future work in Section 5.

2 Derivation of generalized Intrinsic Plasticity

Intrinsic Plasticity (IP) as introduced in [39] models a well-known phenomenon observed in a variety of biological neuron models called *homeostatic plasticity:* biological neurons tend to autonomously adapt to a fixed average firing rate for physiological reasons. In [39] it was shown that such a mechanism, when the neurons have an exponential output firing rate distribution, are effectively maximizing information. While the biological mechanisms are not yet known precisely, it is very plausible that every single neuron tries to balance the conflicting requirements of maximizing its information transmission while at the same time obeying constraints on its energy expenditure. IP formalizes these hypotheses by assuming the following three principles:

(1) *information maximization*: the output of the neuron should contain as much information on the input as possible. This is achieved by maximizing the entropy of the output firing rates;
(2) *constraints on the output distributions*: these are first of all the limited output range of the neuron [42], but can be also the limited energy available [43];
(3) *adapt the neurons intrinsic parameters*: a biological neuron is only able

to adjust its internal excitability, and not the individual synapses (see [44], [45] for reference of this common effect in biological neurons).

The original work of Triesch assumes a constraint on the mean of the output distribution and derives a gradient descent learning rule from these principles for fermi non-linearities. In this work, we extend the formalism and study the effects on two types of transfer functions:

(1) the fermi function: $y = f(x) = \frac{1}{1+\exp(-x)}$,
(2) and the hyperbolic tangent: $y = f(x) = \tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$.

The output of the fermi non-linearity is in the range $[0, 1]$, while the output of the tanh function is in the range $[-1, 1]$. We define intrinsic parameters for these non-linearities by adding a gain a and a bias b:

$$f_{gen}(x) = f(ax + b).$$

Following Triesch' original line of argument the information maximization principle corresponds to a maximization of the entropy of the output distribution of each neuron. In combination with the second principle this leads to maximum entropy (ME) distributions with certain fixed moments. The ME distribution for a given mean (first moment) and support in the interval $[0, \infty]$ is the exponential distribution. Likewise, the ME distribution for a given mean and standard deviation with support in $[-\infty, \infty]$ is the Gaussian. Therefore, to target maximum entropy output distribution in formal neurons, we use for the first moment case fermi neurons having a positive output range $[0, 1]$ and for the second order case tanh neurons with output range $[-1, 1]$. We express the amount the empirical output distribution differs from the desired ME distribution using the Kullback-Leibler divergence:

$$D_{KL}(\widetilde{p}, p) = \int \widetilde{p}(y) \log \left(\frac{\widetilde{p}(y)}{p(y)} \right) dy,$$

where $\widetilde{p}(y)$ is the actual probability density of the neuron's output activity and $p(y)$ the desired probability density function. The learning rule for fermi neurons and targeted exponential output distribution

$$p_{\exp}(y) = \frac{1}{\mu} \exp \left(-\frac{y}{\mu} \right)$$

was derived in [39] and is given here for the sake of completeness. The neuron's gain a and bias b are updated at each time step with the following factors:

$$\Delta b = \eta \left(1 - (2 + \frac{1}{\mu})y + \frac{y^2}{\mu}\right)$$
$$\Delta a = \frac{\eta}{a} + \Delta b x$$

To extend the formalism to include the second moment, we need to minimize the Kullback-Leibler divergence to a desired Gaussian distribution

$$p_{\text{norm}}(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right).$$

Following the line of reasoning by Triesch, this yields for the Kullback-Leibler divergence:

$$\begin{aligned} D_{KL}(\widetilde{p}, p) &= \int \widetilde{p}(y) \log\left(\frac{\widetilde{p}(y)}{\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y-\mu)^2}{2\sigma^2}}}\right) dy \\ &= \int \widetilde{p}(y) \log(\widetilde{p}(y)) dy + \int \widetilde{p}(y) \frac{y^2}{2\sigma^2} dy - \int \widetilde{p}(y) \frac{2\mu y}{2\sigma^2} dy + C \\ &= E\left(\log(\widetilde{p}_x(x)) - \log\left(\frac{\partial f_{gen}}{\partial x}\right) + \frac{1}{2\sigma^2} y^2 - \frac{\mu}{\sigma^2} y\right) + C, \end{aligned}$$

where C are constant terms, the relation $p_y(y)\partial y = p_x(x)\partial x$ is used, $y = f_{gen}(x) = \tanh(ax + b)$ and $p_x(x)$ represents the actual probability density of the input to the neuron.

Deriving this w.r.t. for instance the parameter b yields:

$$\begin{aligned} \frac{\partial D_{KL}}{\partial b} &= \frac{\partial E}{\partial b}\left(\log(\widetilde{p}_x(x)) - \log(\frac{\partial f_{gen}}{\partial x}) + \frac{1}{2\sigma^2} y^2 - \frac{\mu}{\sigma^2} y\right) \\ &= E\left(0 - \frac{\frac{\partial^2 f}{\partial x^2}}{\frac{\partial f}{\partial x}} + \frac{1}{2\sigma^2}\frac{\partial y^2}{\partial b} - \frac{\mu}{\sigma^2}\frac{\partial y}{\partial b}\right). \end{aligned}$$

In this case,

$$\frac{\partial^2 f}{\partial x^2}(\frac{\partial f}{\partial x})^{-1} = \frac{-2y(1-y^2)}{1-y^2} = -2y,$$

so this yields:

$$\frac{\partial D_{KL}}{\partial b} = E\left(-\frac{\mu}{\sigma^2} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y)\right).$$

Similar steps also yield the derivative w.r.t a:

$$\frac{\partial D_{KL}}{\partial a} = E\left(-\frac{\mu x}{\sigma^2} + \frac{xy}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y) - \frac{1}{a}\right).$$

From this, we get the following on-line learning rule with stochastic gradient descent using a learning rate η:

$$\begin{aligned}\Delta b &= -\eta\left(-\frac{\mu}{\sigma^2} + \frac{y}{\sigma^2}(2\sigma^2 + 1 - y^2 + \mu y)\right)\\ \Delta a &= \frac{\eta}{a} + \Delta b x.\end{aligned}$$

The learning rule for the Gaussian/tanh combination bears similarity to the exponential/fermi rule. In particular, the relation between the Δa and Δb terms is identical (in fact, this relation is independent of the desired distribution or non-linearities).

2.1 Effects of bounded activation values

We now want to evaluate whether the IP learning rules are able to converge to the desired output distributions. There is however a problem due to the bounded nature of the output of the neurons we use: They are unable to output arbitrary values. The non-linearities we consider in this contribution are bounded to $[0, 1]$ for fermi and $[-1, 1]$ for tanh. Due to these bounds, when we impose a certain constraint on the moments of the infinite distribution, these moments will not be accurately approximated by a learning rule controlling a neuron with finite output range. The effects of these bounds on the moments of the actual output distribution can be computed as follows.

A probability density function truncated to $[k, l]$ can be derived from its infinite-support version by renormalization: :

$$\widehat{p}(y) = \frac{p(y)\mathcal{H}(y-k)\mathcal{H}(l-y)}{\int_k^l p(y)dy},$$

where $\mathcal{H}(x)$ represents the Heaviside function, which is 0 when $x < 0$ and 1 otherwise. Given the "desired" moments of the distribution p which we use to train, we can now calculate the actual moments that are reached.

The general derivation of these relations is quite complex, so we will only give the explicit relation for the mean on the exponential distribution and the fermi neuron. The exponential distribution truncated to $[0, 1]$ is given by

Figure 1. Statistics of the finite distributions compared to that of infinite ones. The left figure shows mean and standard deviation of the finite exponential distribution against μ of the infinite distribution. They are identical for small μ (linear relation), but then saturate. In the middle and right figures, the standard deviation and mean of the finite normal distribution against variance of the infinite distribution for several settings of μ is shown. The σ of finite and infinite distributions is equal for small σ values an then saturates. The mean should stay constant, but goes to zero for large σ values.

Figure 2. These plots show a comparison between the expected and the estimated output probability density for a reservoir of 100 neurons during 1000 steps. The estimated distributions, shown by the dots, are generated by binning the neurons's output in 200 bins. The expected distributions are shown by a dashed line. For both the exponential and the Gaussian distribution, two different setting for the expected mean are shown.

$$\begin{aligned}\widehat{p}_{\exp}(y) &= \frac{p_{\exp}(y)\mathcal{H}(y-k)\mathcal{H}(l-y)}{\int_0^1 p_{\exp}(y)dy} \\ &= \frac{\frac{1}{\mu}\exp(-y/\mu)\mathcal{H}(y-k)\mathcal{H}(l-y)}{\int_0^1 \frac{1}{\mu}\exp(-y/\mu)dy} \\ &= \frac{\exp(-y/\mu)\mathcal{H}(y-k)\mathcal{H}(l-y)}{\mu(\exp(1/\mu)-1)\exp(-1/\mu)}.\end{aligned}$$

From this we can calculate the actual mean:

$$\begin{aligned}\widehat{\mu}_{\exp} &= \int y\widehat{p}_{\exp}(y)dy \\ &= \frac{\mu\exp(1/\mu)-\mu-1}{\exp(1/\mu)-1}.\end{aligned}$$

We did this also for the standard deviation and for the Gaussian distribution. The theoretical results can be seen in Figure 1.

The degree to which these moments differ from the infinite ones depends on how much of the probability mass of the infinite distribution is cut-off at k and l, as is shown by an example in Figure 2. For small values of μ and σ with respect to k and l, $\widehat{p}$ is a good approximation of p. But for larger values, $\widehat{p}$ approximates a uniform distribution, which has a mean of $(l+k)/2$ and a variance of $((l-k+1)^2-1)/12$.

Experimental validation of these theoretical results can be seen in Figure 3. The setup used for these experiments is the one described in the following section. We see that the theoretical prediction for the mean of the exponential distribution is a good fit. The second moment is however less well predicted. This is mainly due to the "edge effect" of the non-linearity. At the edges of the non-linearity's interval, increasingly high input values are needed to reach the

Figure 3. These plots show the theoretically derived relation between the desired moments and the actual moments (denoted with a dashed line, and can also be seen in Figure 1) of the exponential and Gaussian distribution, and the experimentally measured moments (full lines). The left plot shows the mean for the exponential distribution, the middle plot shows the variance for the exponential distribution, and the right plot shows the variance for the Gaussian distribution. The mean for the Gaussian distribution was set to 0.

edges of the interval. Since the IP learning rule converges, the a values remain bounded. The input to the non-linearity is thus also bounded and the outer edges of the non-linearity can thus never be reached. This can for example be seen in Figure 2. For the Gaussian, we also see a good correspondence, but which gets worse for increasing variance. Here again, edge effects start influencing the result.

2.2 Convergence in the presence of recurrency

Since all the derived rules are based on assuming independence of the neurons' input distributions, it is not self-evident that in a recurrent network, where neurons are coupled, the learning rules will succeed in driving the neurons toward the desired output distributions. Furthermore, each transfer function has only two degrees of freedom. Thus it is worth investigating to what degree IP manages to approximate the desired output distributions in the presence of interfering recurrency. For the single neuron case, Triesch [39] has already demonstrated that IP is able to adjust the bias and gain in a non-recurrent single neuron for approximating an exponential function.

Figure 2 shows the theoretical and actual probability densities for both the fermi/exponential and tanh/Gaussian case, for two different settings of the desired moments. The estimation has been done by scaling a histogram with 200 bins of all neuron outputs over 1000 time steps such that its mass equals 1 (more details on the experimental setup can be found in the next section). The results show that even in the case of using IP in a highly connected recurrent network, we get a very good approximation of the theoretical distributions, with notable deviations only near the bounds of the nonlinearity (which is expected).

It has already qualitatively been shown in a previous publication by Steil (such as [32]) that even though the IP learning rule actually only tunes the temporal distributions of single neurons, when they are coupled together in a recurrent network, the same distribution can be perceived spatially (even on a single time step if the reservoir is large enough)! A sparse temporal distribution thus results in a sparse spatial code. This ergodicity has yet never been proven theoretically to be always reached.

3 Experiments

In this section, we will study the impact of both exponential and Gaussian IP on the performance of RC on several benchmark applications. The standard way to optimize a reservoir is by setting the spectral radius of the interconnection matrix to a given value. This method has however some drawbacks such as that the spectral radius is only a good measure for stability/dynamic regime if the network is not driven by input [46]. There is furthermore a quite large variance on the performance when creating several random reservoirs with the same spectral radius. And finally, the spectral radius stability measure cannot be used on for example fermi nodes, or very constrained topologies. We will show that IP can help with regard to all three issues.

The three benchmarks we use in this section to evaluate the performance of the different IP rules are chosen to span a wide range of desired features we expect from reservoirs. First we use a task which purely evaluates the short-term memory performance of the system. We do this using the Memory Capacity (MC) introduced by Jaeger in [14]. The input of the reservoir consists of a temporal signal $u(t)$ which is drawn from a uniform distribution in the interval $[-.8, .8]$. The outputs consist of an infinite number of outputs $y_s(t)$ which try to reconstruct the delayed input $u(t-s)$ for $s = 0...\infty$. In practice, we take the number of outputs twice as high as the size of the reservoir which is a good approximation since the recall performance will drop drastically when we try to predict more time steps back in time than there are nodes in the network. The total MC is defined as the normalized correlation between the outputs and their associated delayed input:

$$MC = \sum_{s=0}^{\infty} \frac{\langle \widetilde{y}_s(t) u(t-s) \rangle_t}{\left\langle \left(\widetilde{y_s}(t) - \langle \widetilde{y_s}(t) \rangle_t\right)^2 \right\rangle_t \left\langle \left(u(t) - \langle u(t) \rangle_t\right)^2 \right\rangle_t}.$$

Note that it was already shown theoretically in [14] that for this task linear nodes perform optimally and they alone are capable of attaining the theoretical upper bound of $MC = N$ with N the size of the reservoir, and this with a spectral radius very close to one.

Next we use a widely used benchmark for time-series-prediction techniques, a 30th-order NARMA system introduced in [13]. The equation of this system is:

$$y(t+1) = 0.2y(t) + 0.04y(t)\sum_{j=0}^{29} y(t-j) + 1.5u(t-29)u(t) + 0.001$$

where $u(t)$ is uniformly drawn from the interval $[0, 0.5]$. This is a difficult, highly non-linear task which requires a relatively long memory. Performance

is evaluated using the Normalised Root Mean Square Error (NRMSE) which is equal to

$$NRMSE = \sqrt{\frac{\left\langle (\tilde{y}(t) - y(t))^2 \right\rangle_t}{\left\langle (y(t) - \langle y(t) \rangle_t)^2 \right\rangle_t}},$$

where $y(t)$ is the desired output and $\tilde{y}(t)$ is the actual output.

As a last task, we use a real-world classification task where the goal is to perform isolated digits recognition from speech, for which state-of-the-art classification performance was already demonstrated using RC [47]. We use almost the same setup as in [47]: a subset of the TI46 speech corpus is used, where the samples are first transformed into a frequency decomposition using a choclear ear model, which is then resampled 128 times so the reservoir can cope better with the dynamics (for a detailed elaboration on this, we refer to [48]). The classification is performed by training 10 readout functions, each representing one of the spoken digits. The reservoir has to classify the spoken digit at every time step, but the final classification is made by taking the winner-take-all of the temporal mean of all the ten readouts.

3.1 Experimental Settings

In this work we will use the following Echo State Network setup. The reservoir states at time step t, $\mathbf{x}(t) \in \mathbb{R}^{Nx1}$, are calculated by

$$\begin{aligned} \mathbf{x}(0) &= \mathbf{0} \\ \mathbf{x}(t+1) &= \mathbf{f}(\mathbf{W}^{res}_{res}\mathbf{x}(t) + \mathbf{W}^{res}_{inp}\mathbf{u}(t), \mathbf{a}, \mathbf{b}), \end{aligned}$$

where N is the number of network nodes, $\mathbf{W}^{res}_{res} \in \mathbb{R}^{NxN}$ is the matrix of network weights, $\mathbf{f}(.)$ is the vector-valued version of the generalized transfer function $f(.)$, $\mathbf{a}, \mathbf{b} \in \mathbb{R}^N$ are the vectors of gain and bias parameters in $\mathbf{f}(.)$, M is the number of external inputs to the network at each time step, $\mathbf{u}(t) \in \mathbb{R}^{Mx1}$ the external input at time step t and the weights connecting these to the reservoir nodes $\mathbf{W}^{res}_{inp} \in \mathbb{R}^{NxM}$. Note that above equation can be rewritten as

$$\begin{aligned} \mathbf{x}(t+1) &= \mathbf{f}(\hat{\mathbf{W}^{res}_{res}}\mathbf{x}(t) + \hat{\mathbf{W}^{res}_{inp}}\mathbf{u}(t) + \mathbf{b}, \mathbf{1}, \mathbf{0}) \\ \hat{\mathbf{W}}^{res}_{res} &= \mathrm{diag}(\mathbf{a})\mathbf{W}^{res}_{res} \\ \hat{\mathbf{W}}^{res}_{inp} &= \mathrm{diag}(\mathbf{a})\mathbf{W}^{res}_{inp}. \end{aligned}$$

This technique allows us to compare reservoirs that have been adapted by IP directly with networks which have not been adapted. We will call the spectral radius of $\hat{\mathbf{W}}^{res}_{res}$ after applying IP, the effective spectral radius.

The output of a reservoir is calculated as a linear regression of the reservoir state at every time step:

$$\mathbf{y}(t+1) = \mathbf{W}_{res}^{out}\mathbf{x}(t+1) + \mathbf{b}_{out}.$$

It is possible to add linear terms from the input to the output, but in the experiments in this work we did not choose to do this because we want to evaluate the influence of IP on the reservoir, and adding the linear terms can cloud the effects of IP.

We train a memoryless readout in a one-shot fashion for the complete dataset. This can be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{x}(0) & \mathbf{b}_{out} \\ & \vdots \\ \mathbf{x}(T) & \mathbf{b}_{out} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \mathbf{y}(0) \\ \vdots \\ \mathbf{y}(T) \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} \mathbf{W}_{res}^{out} & \mathbf{1} \end{bmatrix}$$

$$\mathbf{AV} \approx \mathbf{B},$$

where $\mathbf{V}$ has to be optimized, given some loss function. We use a squared loss function and solve this using a least squares minimization with a regularization term to keep the linear regressor from over-fitting, which is often called ridge regression:

$$\mathbf{V} = \arg\min_{\mathbf{V}} \left(\mathbf{AV} - \mathbf{B}\right)^2 + \lambda \left\|\mathbf{V}\right\|.$$

This minimization can be efficiently calculated using this matrix solution:

$$\mathbf{V} = \left(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}\right)^{-1}\mathbf{A}^T\mathbf{B}.$$

The λ parameter is the regularization parameter which we determine using a five-fold cross-validation scheme with grid search on an alternating validation set. All the results in this paper are the test results after performing five-fold cross-validation on the total dataset.

Some parameters of the network are kept fixed over all experiments. Network size is always 100 - this could have been optimized, but we care here just for comparison between the two techniques. The $\mathbf{W}_{res}^{res}$ matrix is always created by initializing all the weights uniformly distributed between -1 and 1, which is then scaled to a given spectral radius, whereas the input weights are set with equal probability to -.1 or .1. The IP parameters $\mathbf{a}$ and $\mathbf{b}$ are initialized to $\mathbf{1}$ and $\mathbf{0}$, respectively.

All results in plots and tables are averages and standard deviations over 30 experiments. I.e. for each training setup, 30 experiments were conducted. For training the linear readout, the first 100 time steps of the reservoir dynamics

Figure 4. Results for all three benchmarks for tanh with spectral radius ranging (left column), exponential IP for fermi nodes (middle column) and Gaussian IP for tanh nodes (right column).

Table 1
Best results for the three different benchmarks. IP is better than ranging the spectral radius, both in average performance as in standard deviation (except for one case), denoted between brackets. Some tasks perform better with a Gaussian distribution, others with an exponential distribution.

	fermi - specrad	tanh - specrad	fermi - exp. IP	tanh - Gauss. IP
MC	17.41 (1.48)	29.78 (1.87)	20.32 (0.77)	**31.31 (1.93)**
NARMA	0.77 (0.012)	0.52 (0.050)	0.74 (0.019)	**0.46 (0.042)**
Speech	0.070 (0.018)	0.069 (0.015)	**0.060 (0.012)**	0.069 (0.015)

for every presented example were disregarded to allow the network to "warm up" (forget its initial state) sufficiently.

For pre-training a reservoir, the IP rule is applied with a learning rate of 0.0005 for 100000 time steps (equal to 10 epochs when we use 10 time series, used for the cross-validation, each consisting of 1000 time steps). To check whether IP has had sufficient time to adapt after this time, we verified that $\mathbf{a}$ and $\mathbf{b}$ had converged to small regions and compared the expected probability density with the one estimated from the reservoir's output.

3.2 Results

The three benchmark experiments are now evaluated using different experimental setups. We first use the standard way of scaling reservoirs by ranging over the spectral radius. We do this for both fermi and tanh nodes. Next, we evaluate the use of IP for fermi nodes with an exponential distribution (ranging over μ), and tanh nodes with a Gaussian distribution (ranging over σ). For the Gaussian distribution we only use $\mu = 0$ since for other values, experiments show a considerable performance drop.

The results are shown in Figure 4, where the first row shows the results for the memory task, the second row shows the results for the NARMA task, and the bottom row shows the result for the isolated speech. The columns show the use of ranging the spectral radius for the tanh node type, exponential IP for fermi nodes, and Gaussian IP for tanh nodes. The results for ranging the spectral radius for the fermi nodes are not shown since the performance of this reservoir type is very poor. This is expected since the spectral radius is not a good measure for the dynamic regime when using fermi nodes. The best values for all settings are summarized in Table 1 where the results of ranging

the spectral radius for the fermi nodes are added as a reference.

Memory The best achievable memory capacity of reservoirs without IP is quite different for networks using tanh nodes and those using fermi nodes (see 1). Note that for memory capacity, higher is better. While tanh-networks can have a capacity of up to 31, fermi networks have little more than half that memory, 17. In tanh-networks, the drop in performance for spectral radii smaller than optimal is not drastic. When using IP, the largest change in results can be found in fermi-networks. While the best achievable memory capacity increases by one-fourth, i.e. 4, the variance of the results for one and the same parameter setting, i.e. μ, gets very small, only 0.77, contrasting to setting the spectral radius, where this was 1.48. In tanh-networks, the difference between using spectral radius scaling and IP pre-adaptation was not as pronounced. The best memory capacity achievable there, also increased by using IP from 29.78 to 31.31, but the variance is a little bit worse if using IP. Furthermore, the performance drop for suboptimal parameter settings is equally pronounced for σ as for spectral radii.

NARMA With tanh-networks, the best error achieved was 0.52, at a spectral radius of 0.95. For spectral radii of smaller 0.9 or larger than 1, performance drops considerably. When using Gaussian IP, the performance improves considerably, up to 10%. But, in contrast to the memory task, where the performance drop looks qualitative similar for changes in spectral radius and σ, the optimal value of σ is very small, and the performance decreases steadily for increasing its value. fermi-node networks perform very poorly on this task. When using exponential IP, the performance slightly increases, but is still considerably worse than tanh nodes. Interestingly, if μ was chosen larger than 0.3, its setting did not yield any differences in the results, and the variance drops to very small values of 0.019.

Speech Using IP slightly improves both the average and standard deviation of the performance for both exponential and Gaussian distributions. For this task, the exponential distribution seems to perform best. Another effect is the extreme drop in performance when ranging the spectral radius to high values, where the reservoir becomes unstable. When using IP, this instability is never reached.

When using fermi-neurons, and imposing an exponential distribution, the optimal settings for μ differ for the three different tasks. Increasing μ increases the linear memory present in the network, i.e. the best performance could be observed for $\mu = 0.1$. In NARMA, the opposite was true: Up to $\mu = 0.1$, the variance of results was too high to be usable.

But when using tanh-neurons and imposing a Gaussian distribution, for σ, the optimal setting was almost the same for both tasks, besides the performance drop in memory capacity for $\sigma = 0.1$, smaller settings were better.

An interesting fact can be discovered if comparing optimal spectral radius and the effective spectral radius of the optimal σ: Where the σ was optimal, the effective spectral radius was equal to the optimal spectral radius of a network without IP (see Figure 5). Notice the relatively small variance of the relation between moments and effective spectral radius. Imposing certain output distributions on the nodes of the reservoir is thus actually a precise way of controlling the dynamics in the reservoir. Note that the effective spectral radius for fermi nodes can not at all be related to those of tanh neurons, which are normally used. For tanh neurons, we see that when varying σ over it complete range, we actually vary the effective spectral radius in its most important range: between 0.8 and 1.1.

Nnote that IP and spectral radius influence each other in two ways. Firstly, the initial weight matrix size can alter the learning behavior of IP, because the adjustment factors of the intrinsic parameters by the rule depend on the amount of activity present in the network. Secondly, changing the gain of the transfer function corresponds to scaling all incoming weights, and therefore changing the spectral radius of the weight matrix. Thus, the effective spectral radius after applying IP will be different from the one the network was initialized to.

It is known that the memory capacity is highest for linear reservoirs that have a spectral radius close to one [14]. We see a similar effect when using IP: the best performance is attained for Gaussian distributions with small variance, where to bulk of the dynamics is thus in the linear part of the tanh non-linearity. The optimal $\sigma = 0.2$ which we can clearly relate to an effective spectral radius of 1 in Figure 5. The NARMA task also seems to prefer reservoirs that for the most part operate in their linear regime. This can be explained due to the relatively large memory that is needed to solve the 30th-order NARMA task. The speech task on the other hand appears to be a task that can take advantage of the non-linearities when operating the bulk of the dynamics in the non-linear part of the fermi neurons due to the exponential distribution.

Note that for fermi-node networks, only IP with exponential target distribution is studied here. When not pre-adapting fermi networks, the performance is always very bad. But, when using IP pre-adaptation, fermi neurons can already get the best performance for the speech task. The bad performance on the memory and NARMA task might suggest that the fermi neurons are intrinsically flawed. But when pre-adapting fermi nodes with Gaussian IP, which is possible, but not thoroughly investigated in this work, the results seem to be qualitative similar to the ones achieved with tanh-node networks. The above results thus mainly relate to the distributions, and not to the node types.

Note that in previous work of Steil [32,38] a good performance was achieved

Figure 5. In these plots we show the relation between the mean and standard deviation of the exponential and Gaussian distribution respectively, and the effective spectral radius which is attained after pre-training the reservoir using IP. These plots are generated from the Memory Capacity task, but look almost identical for the other tasks. This shows that there is a clear relation between moments and effective spectral radius, which is task independent.

Figure 6. Example ring topology, where each node is connected to its nearest-neighbors, and the input is only fed to a single neuron.

for fermi nodes with exponential IP on a related NARMA task. This was however accomplished by additional features in the overall reservoir architecture that were deliberately left out in this work to be better able to discern the actual influence of the reservoir: not only the current time step was fed to the reservoir, but also a time delayed version of the input, and both these inputs were also directly fed to the linear output. Since the NARMA task depends on a long input history, and has a large linear component, these two features allow the reservoir to perform better. The optimal value for μ in this setup is actually quite low, which suggests that the reservoir only has to solve the non-linear part of the NARMA task, while the linear, time delayed connection takes care of the linear part. When the reservoir has to solve all these parts by itself, we see that the exponential distribution is not the best option since it is not able to generate a long enough memory of its input, as is suggested by the results on the Memory Capacity task.

4 Constrained topologies

When constructing reservoirs for RC, in most cases completely random and sparse connectivity matrices are used. These type of topologies have very good reservoir properties because the dynamic regime can be easily changed by globally scaling all the weights up or down. Due to this scaling, the dynamic regime can be precisely and progressively varied from very damped to highly unstable dynamics. The topologies are however not easy to be implement on, for example, a planar substrate such as a silicon chip.

When we look at very constrained topologies such as 1D and 2D lattices, which are the easiest topologies to implement on planar substrates, they behave very badly as reservoirs: for example a 1D lattice where the input is only fed to one of the nodes (see Figure 6) will have a very sudden transition from order (only the node which is fed with the input is active, and the activity of all the other nodes is orders of magnitude smaller) to a wildly oscillating reservoir when you scale up the weights. The boundary of stability is very narrow in this case. These topologies can thus not easily be used in the usual sense when

Figure 7. Results for a ring topology, on the left for ranging across the spectral radius without IP, on the right for adjusting the desired standard deviation with IP.

just globally scaling the weights.

We will now demonstrate that imposing distributions on the reservoir dynamics is the key to using these constrained topologies as reservoirs, and this in a very robust and reproducible way. We repeated the memory capacity and NARMA task with the same settings as in the previous section, but now using the constrained topology shown in Figure 6 which is a 1D lattice consisting of 50 neurons which are only connected to their nearest neighbors. For this experiment we only look at tanh nodes and IP with a Gaussian distribution. The results can be seen in Figure 7. We did not do this experiment for the speech task, since this task has multiple inputs (one input for each frequency component in the cochlear model), and in this experiment we only want to evaluate if IP is able to create usable dynamics in a ring topology where only a single neuron initially has some activity.

We first evaluate the performance of this ring topology when just scaling the spectral radius. For both the memory capacity and the NARMA task, scaling the spectral radius performs very poorly. Especially when scaling up the reservoir for the NARMA task, we see a drastic increase in error and variance of error, which is due to the unstable regime which is reached.

When using Gaussian IP to pre-adapt this ring topology, we see an increase in performance for both the memory capacity and NARMA task. This clearly demonstrates that IP is very capable of enforcing a desired dynamic regime on a reservoir, even if its topology is very constrained.

Using this IP rule thus allows to pre-adapt simple, constrained topologies prior to hardware implementation. Due to the pre-adaptation, these very sparsely and regularly connected reservoirs can be actually used to perform tasks in an efficient way.

5 Conclusions and future work

We have presented and derived a reservoir adaptation rule that maximizes the information content of the reservoir states based on a constraint on the mean and variance of the state distribution. This results in exponential and normal distributed reservoir states. We have shown that IP learning is capable of driving the neurons in larger networks to the theoretically derived renormalized mean and variance of the desired exponential and Gaussian output distributions. We have evaluated reservoir performance and effective spectral

radius on several benchmarks with very different characteristics.

The results show that for the three tasks we always get an improvement in performance when pre-adapting the reservoir using IP. We also see a steady decrease in variance, which shows that the spread on the performance when creating random reservoirs can be decreased using a very simple local, unsupervised learning rule which is only used to pre-adapt the reservoir weights.

For tasks that mainly need linear responses (as the memory task and NARMA), the Gaussian distribution performs best, while on tasks that are non-linear (like the speech recognition), the exponential distribution performs best.

Another important effect of IP is that it makes it possible to use node types and topologies which normally perform very poorly as reservoir. When for example training fermi nodes with Gaussian distribution (which was only briefly investigated), we get the same results as when using tanh distribution. The very special ring-topology can, through the use of IP, also be used as a real reservoir. This has far reaching consequences when we implement reservoirs in hardware: a reservoir can be built using hardware friendly nearest neighbour connections, and can still be effectively used as reservoir.

As future work we plan to extend the idea of information maximization to different output distributions, more parameters that can be trained and other node types. We will especially focus on distributions with a high kurtosis since these lead to sparse codes. It has been shown that sparse codes increase signal-to-noise ratio, and improve the detection of coincidences [49]. Distributions such as the Laplace distribution which have a kurtosis of 3 will be investigated. The Gaussian distribution investigated in this work only has a kurtosis of 0.

We will further show that IP can be an enabler for using very constrained topologies in RC, as already hinted in this work. Extending the work to spiking neurons will also show this for LSM-type reservoirs which are currently very difficult to tune, but by using IP can be controlled using a single parameter. A clear link between information maximization, and the influence of it on dynamics must also be further studied (why is there a clear link between parameters of the distribution and for example the spectral radius, which is a measure of stability).

The idea of autonomously regulated robustness of dynamics is a powerful new concept. It was shown that a certain dynamic regime in reservoirs leads to good performance for a given task. The IP rule allows the reservoirs to autonomously perceive and adapt their dynamics to this specific regime, irrespective of disturbances, initial weights or input scaling. Reservoirs were already robust in the sense that the performance variance for random reservoirs is small. This robustness even improves when adding IP, since reservoirs can now autonomously form the correct dynamic properties. In future work,

this idea of robust computing using self-regulating dynamic systems will be further investigated.

Acknowledgment

This work partially funded the Research Foundation - Flanders project G.0317.05, and the *Photonics@be* Interuniversity Attraction Poles program (IAP 6/10), initiated by the Belgian State, Prime Minister's Services, Science Policy Office.

References

[1] F. Takens, D. A. Rand, Y. L.-S., Detecting strange attractors in turbulence, in: Dynamical Systems and Turbulence, Vol. 898 of Lecture Notes in Mathematics, Springer-Verlag, 1981, pp. 366–381.

[2] P. J. Werbos, Beyond regression: New tools for prediction and analysis in the behavioral sciences, Ph.D. thesis, Applied Mathematics, Harvard University, Boston, MA (1974).

[3] P. J. Werbos, Backpropagation through time: what it does and how to do it, Proc. IEEE 78 (10) (1990) 1550–1560.

[4] D. Rumelhart, G. Hinton, R. Williams, Parallel Distributed Processing, MIT Press, Cambridge, MA, 1986, Ch. Learning internal representations by error propagation.

[5] B. Hammer, J. J. Steil, Perspectives on learning with recurrent neural networks, in: Proceedings of the European Symposium on Artificial Neural Networks (ESANN), 2002.

[6] G. V. Puskorius, F. L. A., Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks, IEEE Transactions on Neural Networks 5 (1994) 279–297.

[7] J. Schmidhuber, S. Hochreiter, Long short-term memory, Neural Computation 9 (1997) 1735–1780.

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

[9] W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural Computation 14 (11) (2002) 2531–2560.

[10] Special issue on Echo State Networks and Liquid State Machines, Neural Networks 20 (3).

[11] J. J. Steil, Backpropagation-Decorrelation: Online recurrent learning with O(N) complexity, in: Proceedings of the International Joint Conference on Neural Networks (IJCNN), Vol. 1, 2004, pp. 843–848.

[12] U. D. Schiller, J. J. Steil, Analyzing the weight dynamics of recurrent learning algorithms, Neurocomputing 63C (2005) 5–23.

[13] A. F. Atiya, A. G. Parlos, New results on recurrent network training: Unifying the algorithms and accelerating convergence, IEEE Transactions on Neural Networks 11 (2000) 697.

[14] H. Jaeger, Short term memory in echo state networks, Tech. Rep. GMD Report 152, German National Research Center for Information Technology (2001).

[15] W. Maass, T. Natschläger, M. H., Fading memory and kernel properties of generic cortical microcircuit models, Journal of Physiology 98 (4-6) (2004) 315–330.

[16] P. Joshi, W. Maass, Movement generation and control with generic neural microcircuits, in: Proc. of BIO-AUDIT, 2004, pp. 16–31.

[17] H. Burgsteiner, Training networks of biological realistic spiking neurons for real-time robot control, in: Proceedings of the 9th International Conference on Engineering Applications of Neural Networks, Lille, France, 2005, pp. 129–136.

[18] H. Burgsteiner, On learning with recurrent spiking neural networks and their applications to robot control with real-world devices, Ph.D. thesis, Graz University of Technology (2005).

[19] W. Maass, R. A. Legenstein, H. Markram, A new approach towards vision suggested by biologically realistic neural microcircuit models, in: Proc. of the 2nd Workshop on Biologically Motivated Computer Vision, Lecture Notes in Computer Science, Springer, 2002.

[20] H. Jaeger, Reservoir riddles: Sugggestions for echo state network research (extended abstract), in: Proceedings of the International Joint Conference on Neural Networks, 2005, pp. 1460–1462.

[21] J. Hertzberg, H. Jaeger, F. Schönherr, Learning to ground fact symbols in behavior-based robots, in: Proceedings of the 15th European Conference on Artificial Intelligence, 2002, pp. 708–712.

[22] M. Oubbati, P. Levi, M. Schanz, T. Buchheim, Velocity control of an omnidirectional robocup player with recurrent neural networks, in: Proceeding of the Robocup Symposium, 2005, pp. 691–701.

[23] P. G. Plöger, A. Arghir, T. Günther, R. Hosseiny, Echo state networks for mobile robot modeling and control, in: RoboCup 2003: Robot Soccer World Cup VII, 2004, pp. 157–168.

[24] M. Salmen, P. G. Plöger, Echo state networks used for motor control, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 1953–1958.

[25] K. Bush, C. Anderson, Modeling reward functions for incomplete state representations via echo state networks, in: Proceedings of the International Joint Conference on Neural Networks, Montreal, Quebec, 2005.

[26] W. Maass, T. Natschläger, H. Markram, A model for real-time computation in generic neural microcircuits, in: Proceedings of NIPS, Vol. 15, MIT Press, 2003, pp. 229–236.

[27] D. Verstraeten, B. Schrauwen, D. Stroobandt, J. Van Campenhout, Isolated word recognition with the liquid state machine: a case study, Information Processing Letters 95 (6) (2005) 521–528.

[28] M. D. Skowronski, J. G. Harris, Minimum mean squared error time series classification using an echo state network prediction model, in: IEEE International Symposium on Circuits and Systems, 2006.

[29] H. Jaeger, H. Haas, Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication, Science 308 (2004) 78–80.

[30] Y. Rao, S.-P. Kim, J. Sanchez, D. Erdogmus, J. Principe, J. Carmena, M. Lebedev, M. Nicolelis, Learning mappings in brain machine interfaces with echo state networks, in: Proceedings of the 2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005, pp. 233–236.

[31] H. Jaeger, Adaptive nonlinear system identification with echo state networks, in: Advances in Neural Information Processing Systems, 2003, pp. 593–600.

[32] J. J. Steil, Online stability of backpropagation-decorrelation recurrent learning, Neurocomputing 69 (2006) 642–650.

[33] J. J. Steil, Memory in backpropagation-decorrelation O(N) efficient online recurrent learning, in: Proceedings of the International Conference on Artificial Neural Networks (ICANN), 2005.

[34] D. Verstraeten, B. Schrauwen, M. D'Haene, D. Stroobandt, A unifying comparison of reservoir computing methods, Neural Networks 20 (2007) 391–403.

[35] D. Verstraeten, B. Schrauwen, D. Stroobandt, Reservoir-based techniques for speech recognition, in: Proceedings of the World Conference on Computational Intelligence, 2006, pp. 1050–1053.

[36] J. J. Steil, Several ways to solve the mso problem, in: Proceedings of the European Symposium on Artificial Neural Networks (ESANN), 2007, pp. 489–494.

[37] M. Wardermann, J. J. Steil, Intrinsic plasticity for reservoir learning algorithms, in: Proceedings of the European Symposium on Artificial Neural Networks (ESANN), 2007, pp. 513–518.

[38] J. J. Steil, Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning, Neural Networks 20 (3) (2007) 353 – 364.

[39] J. Triesch, A gradient rule for the plasticity of a neuron's intrinsic excitability, in: Proc. of the Int. Conf. on Artificial Neural Networks (ICANN), 2005.

[40] J. Triesch, Synergies between intrinsic and synaptic plasticity mechanisms, Neural Computation 19 (2007) 885–909.

[41] A. Lazar, G. Pipa, J. Triesch, Fading memory and times series prediction in recurrent networks with different forms of plasticity, Neural Networks 20 (3) (2007) 312–322.

[42] J. J. Atick, Could information theory provide an ecological theory of sensory processing?, Network: Computation in Neural Systems 3 (2) (1992) 213–251.

[43] R. Baddeley, L. F. Abbott, M. C. A. Booth, F. Sengpiel, T. Freeman, E. A. Wakeman, E. T. Rolls, Responses of neurons in primary and inferior temporal visual cortices to natural scenes, in: Proceedings in Biological Sciences, Vol. 264, 1997, pp. 1775 – 1783.

[44] W. Zhang, D. J. Linden, The other side of the engram: Experience-driven changes in neuronal intrinsic excitability, Nature Reviews Neuroscience 4 (2003) 885–900.

[45] A. Destexhe, E. Marder, Plasticity in single neuron and circuit computations, Nature 431 (2004) 789–795.

[46] M. C. Ozturk, D. Xu, J. C. Principe, Analysis and design of echo state networks, Neural Computation 19 (2006) 111–138.

[47] D. Verstraeten, B. Schrauwen, D. Stroobandt, Isolated word recognition using a liquid state machine, in: Proceedings of the 13th European Symposium on Artificial Neural Networks (ESANN), 2005, pp. 435–440.

[48] B. Schrauwen, J. Defour, D. Verstraeten, J. Van Campenhout, The introduction of time-scales in reservoir computing, applied to isolated digits recognition, in: Proceedings of the International Conference on Artificial Neural Networks (ICANN), 2007.

[49] D. J. Field, What is the goal of sensory coding?, Neural Computation 6 (1994) 559–601.