Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2013, Article ID 891501, 9 pages
http://dx.doi.org/10.1155/2013/891501

Research Article

A Novel Method for Training an Echo State Network with Feedback-Error Learning

Rikke Amilde Løvlid

Department of Computer and Information Science, Norwegian University of Science and Technology, Sem Sælands vei 7-9, 7491 Trondheim, Norway

Correspondence should be addressed to Rikke Amilde Løvlid; rikke-amilde.lovlid@ffi.no

Received 31 May 2012; Revised 10 December 2012; Accepted 19 February 2013

Academic Editor: Ralf Moeller

Echo state networks are a relatively new type of recurrent neural networks that have shown great potentials for solving non-linear, temporal problems. The basic idea is to transform the low dimensional temporal input into a higher dimensional state, and then train the output connection weights to make the system output the target information. Because only the output weights are altered, training is typically quick and computationally efficient compared to training of other recurrent neural networks. This paper investigates using an echo state network to learn the inverse kinematics model of a robot simulator with feedback-error-learning. In this scheme teacher forcing is not perfect, and joint constraints on the simulator makes the feedback error inaccurate. A novel training method which is less influenced by the noise in the training data is proposed and compared to the traditional ESN training method.

1. Introduction

A recurrent neural network (RNN) is a neural network with feedback connections. Mathematically RNNs implement dynamical systems, and in theory they can approximate arbitrary dynamical systems with arbitrary precision [1]. This makes them "in principle promising" as solutions for difficult temporal tasks, but in practice, supervised training of RNNs is difficult and computationally expensive.

Echo state networks (ESNs) were proposed as a cheap and fast architectural and supervised learning scheme and are therefore suggested to be useful in solving real problems [2]. The basic idea is to transform the low dimensional temporal input into a higher dimensional *echo state*, and then train the output connection weights to make the system output the desired information. The idea was independently developed by Maass [3] and Jaeger [4] as liquid state machine (LSM) and echo state machine (ESM), respectively.

LSMs and ESMs, together with the more recently explored Backpropagation Decorrelation learning rule for RNNs [5], are given the generic term reservoir computing [6]. Typically large, complex RNNs are used as reservoirs, and their function resembles a tank of liquid. One can think of the input as stones thrown into the liquid, creating unique ripples that propagate, interact, and eventually fade away. After learning how to read the water's surface, one can extract a lot of information about recent events, without having to do the complex input integration. Real water has successfully been used as a reservoir [7].

Because only the output weights are altered, training is typically quick and computationally efficient compared to training of other recurrent neural networks.

We are investigating how to use an ESN to learn internal models of a robot's motor apparatus. An internal model is a system that mimics the behavior of a natural process. In this paper we will talk about inverse models, which transform preplanned trajectories of desired perceptual consequences into appropriate motor commands.

The inverse model is often divided into a kinematic and a dynamic model. An inverse kinematic model transforms a trajectory in task space (e.g., cartesian coordinates) to a trajectory in actuator space (e.g., joint angles), and an inverse dynamic model transforms the joint space trajectory into the sequence of forces that will actually move the limbs. The

robot simulator in our experiments is controlled by the joint angle velocities directly, thus we are only concerned with kinematics.

It is common to use analytical internal models, and deriving such a model for our simulator would be easy. Despite this, we want to explore using an ESN as an inverse model, because as robots become more complex, with springy joints, light limbs and many degrees of freedom, acquiring analytical models will become more and more difficult [8]. Oubbati et al. also argue that substituting the analytical models with a recurrent neural networks might be beneficial in general, as it can make the inverse model more robust against noise and sensor errors [9].

To acquire an accurate inverse model through learning is, however, problematic, because the target motor commands are generally unavailable. What is known is the target trajectory in task space. Three schemas have been suggested for training the inverse model: directly by observing the effect of different motor commands on the controlled object [10], with a forward model as a distal teacher [11], or with an approach called feedback-error learning (FEL) [10]. Direct modeling was excluded because it cannot handle redundancies in the motor apparatus and therefore will not scale to real problems [11]. FEL was chosen over distal teacher because it is a natural extension of using an analytical model, and because it is biologically motivated due to its inspiration from cerebellar motor control [12]. Another advantage, which we will not exploit here, is that FEL can be used for control during learning.

The objective in this paper is to investigate how an ESN can be trained within this FEL scheme. The traditional ESN learning method falls short in this setup due to inaccurate teacher forcing and target estimation. We propose a novel training method, which is inspired by gradient decent methods and shows promising results on this problem. Preliminary studies of this training method can be found in a related work [13]. The current paper includes further studies of why this new method works so well.

2. Learning to Imitate YMCA

In this paper an ESN is trained to execute an arm movement on a simple robot simulator by computing the inverse kinematics of that movement. The ESN is only tested on the movement it was trained on, which means that we do not verify whether the ESN has actually learned the inverse model or merely to execute this particular trajectory. We have earlier investigated the benefit of learning the inverse model by training on one movement with certain properties [14]. Here we have a slightly more complex inverse problem and encountered a problem when trying to learn the training sequence itself. The solution to that problem is the main point in this paper.

2.1. Training Data. The movement data is a recording of the dance to the song YMCA by the Village People. It was gathered with a Pro Reflex 3D motion tracking system by Tidemann and Öztürk [15]. The system is able to track the position of fluorescent balls within a certain volume by using five infrared cameras. The sampling frequency of the Pro Reflex is 200 Hz. In the experiments we used every fourth sample, meaning the position trajectory consisted of 50 samples/sec, resulting in a sequence with 313 steps.

The tracking of the balls yields cartesian coordinates of the balls in three dimensions. The result was projected down to two dimensions, and the position of each arm was expressed as the x and z coordinates of the elbow relative to the shoulder and the wrist relative to the elbow. The coordinates were normalized to be in the interval $\langle -1, 1\rangle$. The position in each time step was thus represented by 8 signals, that is, $(x_{\text{elbow}}, z_{\text{elbow}}, x_{\text{wirst}}, z_{\text{wrist}})$ for each arm.

2.2. Simulator. For the simulations we used a fairly simple 2D simulator with four degrees of freedom (DOFs), one in each shoulder and one in each elbow. The simulated robot was controlled by the joint angle velocities directly, which means that the problem of translating the velocities into torques was not considered. The ESN was trained to output the joint angle velocities that would keep the elbows and wrists on the desired trajectory. The velocities were scaled to be in the interval $\langle -1, 1\rangle$ and will be referred to as the *motor commands*.

The range of motion was constrained to be between 0° and 180° for all 4 DOFs, and if the motor command implied moving the limb further, the limb stopped at the limit and the overshooting motor command was ignored.

The maximum joint angle velocity for each DOF was set to twice the maximum velocity registered in the recorded movement, which meant that a joint angle velocity equal to 1 moved the joint less than 180 degrees. Limited joint velocity is realistic, and it also makes large errors in motor commands lead to smaller position errors, making the movements look smoother.

2.3. Control Architecture. The ESN is trained to compute motor commands that will move the simulated arms from the current position to the next position in the target trajectory. The target motor commands needed for training are not available; what is available is the target positions.

The FEL scheme, illustrated in Figure 1, includes a feedback controller that estimates the error in motor command from the position error. The motor error computed by the feedback controller is used both to train the ESN and to adjust the motor command from the inverse model before it is sent to the arm simulator. In the current setup the transformation from position error to motor error is simple enough to be done analytically, but using the result will still not be perfect as the simulator is noisy and the calculation does not take into consideration any excess motor commands that were potentially ignored if the limbs were moved to their limits.

How much influence the feedback controller has on the final motor command is regulated by the feedback gain, K. To facilitate learning and force the feedback controller to become redundant, the feedback gain was linearly reduced from 1 to 0 during several rounds of training.

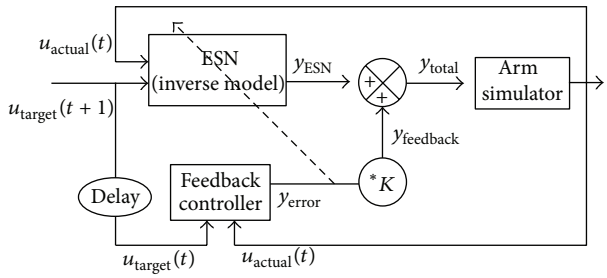


Figure 1: The figure illustrates the feedback-error-learning (FEL) architecture used to training the ESN. The input to the ESN is the actual position at the current time step ($u_{\text{actual}}(t)$) and the next position in the target position trajectory ($u_{\text{target}}(t+1)$). The ESN learns to calculate the motor command which will move the simulated arms from the current position to the next position in the target trajectory. The motor command from the ESN is called y_{ESN} and is adjusted by the motor command from the feedback controller, y_{feedback}, before it is used to move the simulated arms. The feedback controller estimates the error of this total motor command (y_{error}) by comparing the resulting position with the corresponding target position. This error is used to train the ESN and to compute the feedback motor command in the next time step. The feedback gain, K, determines how much the feedback controller can influence the total motor command.

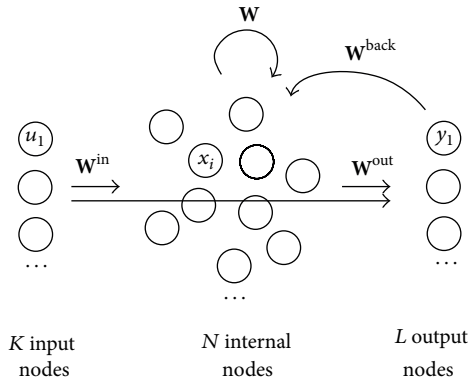


Figure 2: The figure illustrates a basic ESN.

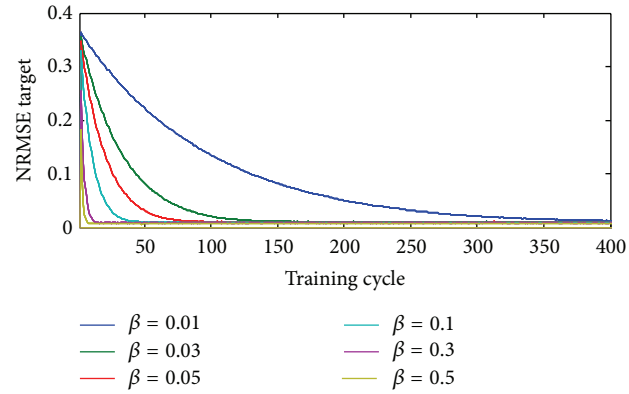


Figure 3: The plot shows the difference between the true target and the used target in each training cycle for different values of β when target estimation and teacher forcing are perfect. The result is used to deduce how many extra cycles of training are needed for different values of β. Note that with $\beta = 1$, the used target and the true target will be equal from the start, and only one cycle of training is needed.

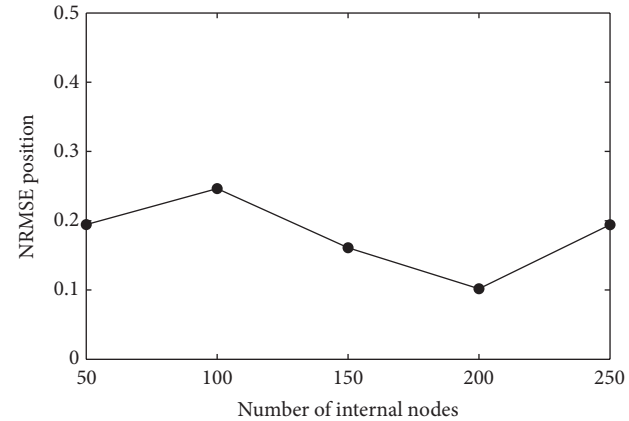


Figure 4: To determine the optimal reservoir size, ESNs with different numbers of internal nodes were trained with the original training method within the FEL scheme. The YMCA movement was repeated 5 times in the training sequence, and the internal noise level was 0.02. The figure shows the mean position errors during testing for 10 repetitions of each experiment.

3. Training an Echo State Network

A basic echo state network is illustrated in Figure 2. The activation of the internal nodes is updated according to

$$\mathbf{x}(t) = f\left(\mathbf{W}^{\text{in}}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t-1) + \mathbf{W}^{\text{back}}\mathbf{y}(t-1)\right) + v(t-1), \quad (1)$$

where f is the node's activation function, and v are white Gaussian noise. The output of the network is computed according to

$$\mathbf{y}(t) = f^{\text{out}}\left(\mathbf{W}^{\text{out}}(\mathbf{u}(t), \mathbf{x}(t))\right). \quad (2)$$

A general task is described by a set of input and desired output pairs, $[\langle \mathbf{u}(1), \mathbf{y}_{\text{target}}(1)\rangle, \langle \mathbf{u}(2), \mathbf{y}_{\text{target}}(2)\rangle, \ldots, \langle \mathbf{u}(T), \mathbf{y}_{\text{target}}(T)\rangle]$, and the solution is a trained ESN whose output $\mathbf{y}(t)$ approximates the teacher output $\mathbf{y}_{\text{target}}(t)$, when the ESN is driven by the training input $\mathbf{u}(t)$.

3.1. Original Training Method. Training the ESN using the original training methods is done in three steps. First, a random RNN with the echo state property is generated [4]. Second, the training sequence is run through the network once. If there are feedback connections, teacher forcing is used, meaning $\mathbf{y}(t)$ is replaced by $\mathbf{y}_{\text{target}}(t)$ when computing $\mathbf{x}(t+1)$ and $\mathbf{y}(t+1)$. After the first T_0 time steps, which are used to wash out the initial transient dynamics, the states of each input and internal node in each time step are stored in a state collection matrix, $\mathbf{M}$. Assuming tanh is used as output activation function, $\tanh^{-1}(\mathbf{y}_{\text{target}}(t))$ is collected row-wise

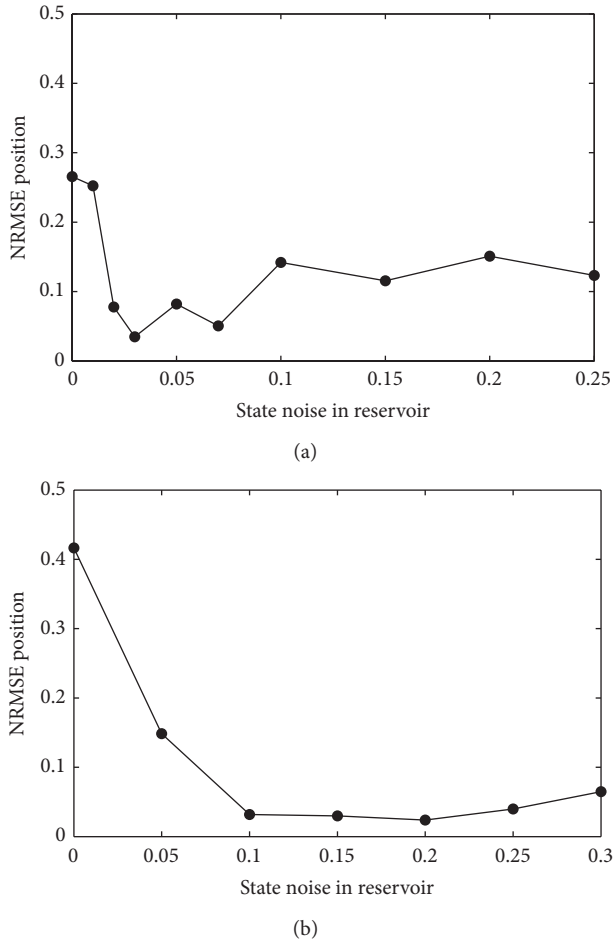


Figure 5: The optimal choice for the level of internal noise in the reservoir was significantly different for the two training methods. The figures show the mean position error during testing for different noise levels. (a) The networks were trained with the original method with the training sequence consisting of 5 repetitions of the YMCA movement. (b) The corresponding results when the networks were trained with the new method with $\beta = 0.1$ and the movement sequence repeated once. All experiments were run 10 times, and the number of internal nodes was 200 in all the networks. Based on the results we chose noise level 0.03 for the original method and 0.2 for the new method.

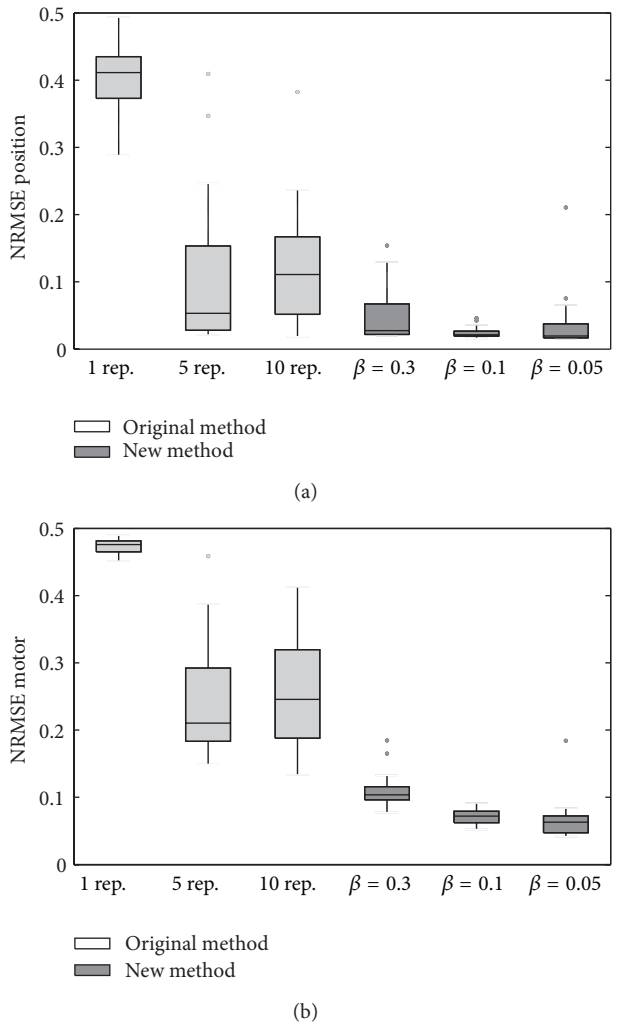


Figure 6: The figures show a box and whisker plots for 20 runs of each of the 6 experiments. Plot (a) illustrates the position error during testing and plot (b) the motor error during testing. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.

into a target collection matrix $\mathbf{S}$. Equation (2) can then be written as

$$\mathbf{S} = \mathbf{M}\left(\mathbf{W}^{\text{out}}\right)^{T}. \tag{3}$$

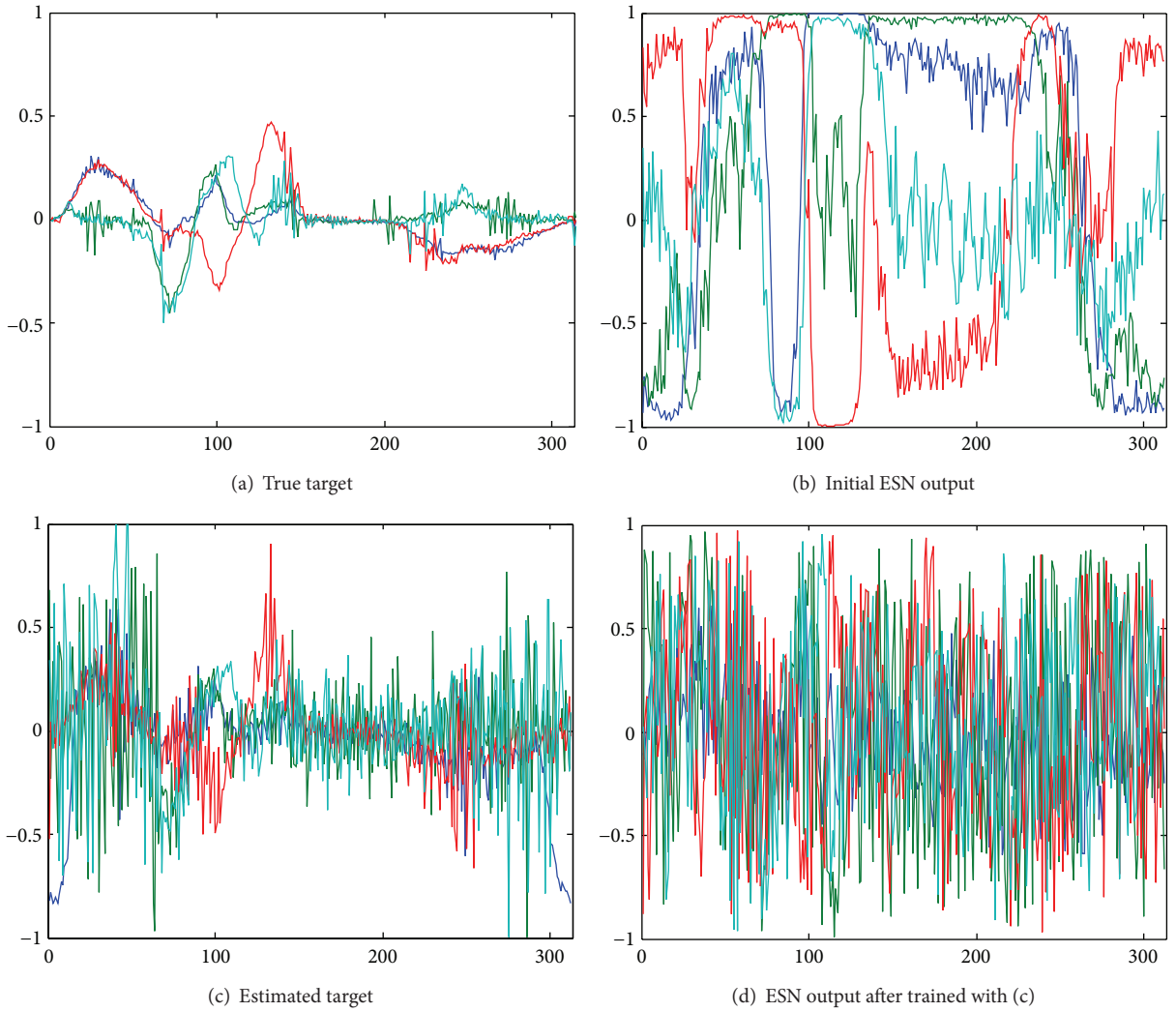
Third, the output weights are computed by using the Moore-Penrose pseudoinverse to solve (3) with regard to $\mathbf{W}^{\text{out}}$:

$$\left(\mathbf{W}^{\text{out}}\right)^{T} = \mathbf{M}^{+}\mathbf{S}. \tag{4}$$

3.2. New Proposed Training Method. In the original training method the training sequence is run through the network once, and the output weights are updated based on the target collection matrix and the state collection matrix as shown in (4). This does not work well with our training architecture, because teacher forcing and target estimation are far from perfect. We therefore suggest *running the training sequence through several times* for each value of the feedback gain. For each of these cycles the output weights are calculated based on the state collection matrix and something in between the estimated target and the actual output from the ESN model. One has

$$\mathbf{y}^{i}_{\text{used target}}(t) = \beta \mathbf{y}_{\text{estimated target}}(t) + \left(1-\beta\right)\ \mathbf{y}_{\text{ESN}}(t). \tag{5}$$

The vector $\mathbf{y}^{i}_{\text{used target}}(t)$ is the target used to generate the target matrix $\mathbf{S}$ for computing $\mathbf{W}^{\text{out}}$ in cycle i, and $\mathbf{y}_{\text{estimated target}}(t)$ is an estimate of the target, as the true target is not available. Note that $\beta = 1$ corresponds to the original training method.



(a) True target

(b) Initial ESN output

(c) Estimated target

(d) ESN output after trained with (c)

FIGURE 7: The plots illustrate why the original method without repetitions (experiment 1) fails. Compared to the true target (a), the estimated target in the first epoch (c) is very noisy. It has the general shape of the true target, but when training the initial, random ESN (b) with this noisy estimate, the result is a network which outputs mostly noise (d). This only gets worse in the succeeding epochs. Plotted are motor commands (joint angle velocities) for the 4 DOFs at each time step in the training sequence.

We hypothesize that this new proposed training method will improve learning. However, the training time increases as β decreases because additional cycles of training are needed. To test how many cycles are needed to converge for each value of β, the network was trained with the true target and perfect teacher forcing for 400 cycles. The true target was found by using an analytical inverse model. Figure 3 illustrates the difference between the true target, $\mathbf{y}_{\text{target}}$, and the used target, $\mathbf{y}_{\text{used target}}$, in each cycle, i. To compensate for this extra computation time, we will try reducing the length of the training sequence when applying this training method.

4. Experiments

The performance of the new proposed method is compared to the performance of the original method through different experiments. Our main hypothesis is that the new method will provide the same or better performance as the original at a smaller computational cost.

In all the experiments the ESN was trained to execute the YMCA movement. It was trained with feedback-error learning with the feedback gain linearly being decreased from 1 to 0 during 10 epochs of training. During testing the ESN was run without the feedback controller and the performance was measured as how accurately the ESN was able to reproduce the training sequence.

The original training method was used on training sequences with varying number of repetitions of the YMCA movement. We hypothesize that training on longer sequences, where the movement is repeated several times, will increase the performance. However, a longer training sequence leads to longer training time.

The new training method was investigated by conducting experiments for three different values of β. All trained

Table 1: The table summarizes the experiment details, including the value of β ($\beta = 1$ means the original method), the number of cycles per epoch, and the number of repetitions of the YMCA movement constituting the training sequence. In all the experiments the ESN was trained for 10 epochs with decreasing feedback gain.

Experiment number	β	# cycles per epoch	# rep. movement
Exp. 1	1	1	1
Exp. 2	1	1	5
Exp. 3	1	1	10
Exp. 4	0.3	2	1
Exp. 5	0.1	3	1
Exp. 6	0.05	10	1

on just one repetition of the YMCA movement, but the sequence had to be presented several times for each epoch to make it possible for the used target to converge during the 10 training epochs. The number of cycles used for each epoch was the approximate number of cycles needed for convergence according to Figure 3, divided by the number of epochs.

Table 1 holds the details of the different experiments.

4.1. Parameters. The ESN had 8 input nodes, corresponding to the x and z coordinates of the shoulders and elbows, and 4 output nodes, one for each DOF of the simulator. We used 200 nodes in the internal network, which was optimized for the original training method as illustrated in Figure 4.

When implementing the ESN, we used the simple matlab toolbox provided by Jaeger et al. [16]. The spectral radius was 0.5 and tanh was used as output function. The reservoir noise level was set to 0.03 when using the original method and 0.2 when using the new method. These noise levels are justified in Figure 5. All other network parameters used were the default in the toolbox. Gaussian noise with mean 0 and standard deviation 0.01 was added to the output from the arm simulator.

4.2. Training and Testing. The feedback controller was only used during training, and the feedback gain was reduced from 1 to 0 during 10 epochs. Before each epoch the ESN was reinitialized by setting the internal states to 0 and running the training sequence through once without learning. The epoch continued with one cycle of training when using the original training method and several cycles of training when $\beta < 1$. One last circle without training (but with use of the feedback controller) was run in each epoch to evaluate the performance at that stage.

After training the network was again reinitialized and tested on the training sequence by running it through once without the feedback controller.

To evaluate the performance we use the Root Mean Square Error (RMSE) of the resulting position sequence normalized over the range of the output values:

$$\mathrm{MSE}\left(\mathbf{y}, \mathbf{y}_{\text{true target}}\right) = \frac{\sqrt{\mathrm{MSE}}}{y_{\max} - y_{\min}} = \frac{\sqrt{\mathrm{MSE}}}{2}. \quad (6)$$

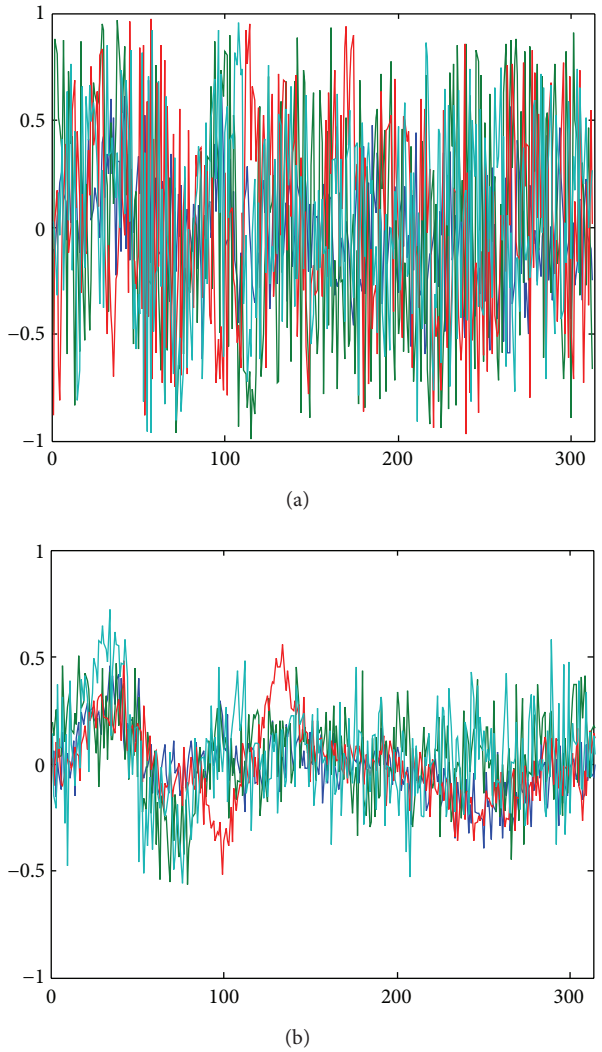


Figure 8: Adding more repetitions of the movement in the training sequence makes the output of the ESN seem less noisy. Plot (a) shows the output of the ESN after training with one repetition and plot (b) the ESN output after training on 5 repetition of the YMCA movement.

The NRMSE for each run was averaged over all time steps and DOFs. A NRMSE = 0 means no error, a random solution would have NRMSE ≈ 0.5, and NRMSE = 1 means opposite solution.

5. Results

Each of the six experiments were repeated 20 times, and the results are summarized in Table 2 and illustrated in Figure 6.

The motor error of experiment 1 is close to 0.5, which means that using the original training method on one repetition of the YMCA sequence results in a network that does not perform better than a random network. Repeating the movement in the training sequence (experiments 2 and 3) helps, but note that the variance is pretty large.

Using the new training method makes a larger improvement with a lower additional computational cost. From the box and whisker plot in Figure 6(b) we see that the worst ESN obtained by using the new method with $\beta = 0.1$ (experiment 5) performed better than the best ESN obtained with the original method trained on 5 repetitions of the YMCA (experiment 2). Due to the computation time of the pseudo-inverse calculations, the training time of a sequence of length $m * n$ is longer than training a sequence of length m n times [17]. This implies that the running time of experiment 5 (sequence of 313 steps run $3 * 10$ times) is also shorter than the running time of experiment 2 (sequence of $5 * 313$ steps run 10 times).

5.1. Why the New Method Outperforms the Original. To understand the effects of the different experimental setups we trained the same initial network with the setups in experiments 1 (original, 1 rep.), 2 (original, 5 rep.), and 5 (new, $\beta = 0.1$) and studied how the ESN output, the actual position sequence, the estimated target, and the target used for weight calculation evolved during the training epochs.

Figure 7 shows why experiment 1 fails. The estimated target sequence is too noisy, and with the short training sequence without any repetitions, the output from the ESN becomes even noisier.

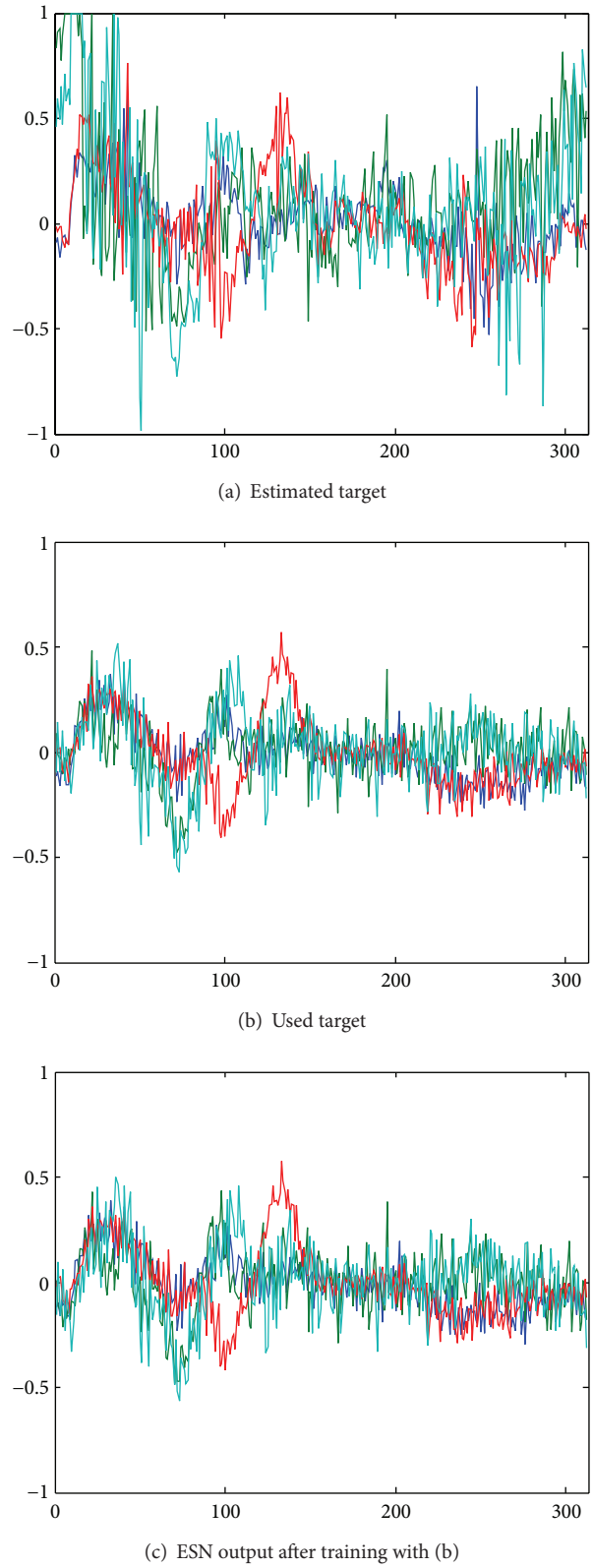
The output from the ESN after training becomes significantly less noisy when the movement is repeated several times in the training sequence, as illustrated in Figure 8. In this setup the target sequence does have a repeating pattern, and since the error in each repetition will differ, the weight calculation will average over these slightly different representations.

When using the new training method, the approach for making a smoother target is different. The new method is apparently able to keep the smoothness of the output of the first, random network and just gradually drives that solution toward the target. As illustrated in Figure 9 the used target, that is, the best target estimate combined with the previous ESN output, appears much less noisy than the target estimate alone.

The new method also results in better teacher forcing. Figure 10 illustrates the quality of the teacher forcing for the three selected experiments.

6. Discussion and Conclusion

This paper investigates using feedback-error learning to train an ESN to learn the inverse kinematics of an arm movement. When applying feedback-error learning, teacher forcing is not perfect, and joint constraints on the simulator make the feedback error inaccurate. A novel training method is suggested, which uses a combination of the previous ESN output and the estimated target to train the network. This presumably keeps much of the smoothness of the output from the initial, random network and avoids the unstable output obtained when training with the estimated target directly.



(a) Estimated target

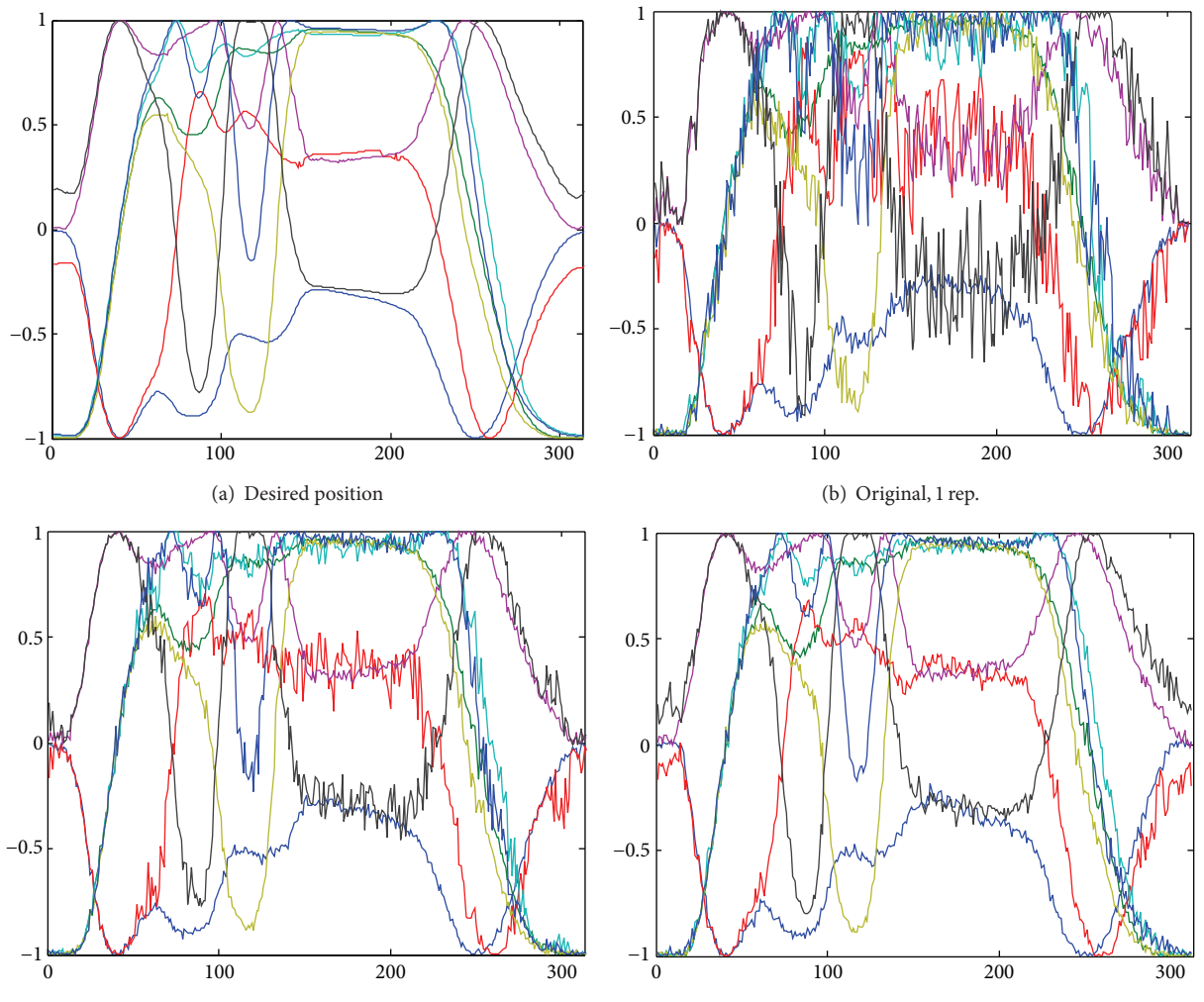
(b) Used target

(c) ESN output after training with (b)

Figure 9: In experiment 5 the network is trained on one repetition of the YMCA movement with $\beta = 0.1$. The plots show (a) the estimated target, (b) the used target, and (c) the ESN output after training with (b). All the plots are from epoch 5, where the used target is starting to look like the true target. Notice that the used target appears less noisy than the estimated target.

Table 2: The mean NRMSE and variance for 20 repetitions of each experiment. All the networks were tested on one repetition of the YMCA movement.

Experiment	Position error	Var	Motor error	Var
1 Orig. method, 1 rep.	0.4000	0.0024	0.4737	$1.4E-04$
2 Orig. method, 5 rep.	0.1088	0.0125	0.2435	0.0071
3 Orig. method, 10 rep.	0.1193	0.0081	0.2500	0.0066
4 New method, $\beta = 0.3$	0.0494	0.0018	0.1100	$6.9E-04$
5 New method, $\beta = 0.1$	0.0245	$7.0E-05$	0.0717	$1.4E-04$
6 New method, $\beta = 0.05$	0.0385	0.0020	0.0669	$9.2E-04$



(a) Desired position

(b) Original, 1 rep.

(c) Original, 5 rep.

(d) New, $\beta = 0.1$

Figure 10: The figure illustrates the quality of the teacher forcing in experiments 1, 2, and 5. For each of these experiments the position sequences in epoch 5 are plotted as the 8 coordinate values at each time step for one repetition of the YMCA movement.

The new method requires extra training cycles to converge, but we showed that this can be compensated by using a shorter training sequence.

For benchmark sequences like generation of the figure-eight [18] or a chaotic attractor like the Mackey-Glass system [19], it will be interesting to see whether this new method could be faster than the original method, as it can get the same performance by training on a shorter training sequence. Preliminary results on the generation of the figure-eight verify that a shorter training sequence is needed with the new method, but the potential computational benefits are not yet extensively tested.

References

[1] K. Doya, "Universality of fully connected recurrent neural networks," Tech. Rep., University of California, San Diego, Calif, USA, 1993, Submitted to: *IEEE Transactions on Neural Networks*.

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

[3] T. Natschläger, W. Maass, and H. Markram, "The "liquid computer": a novel strategy for real-time computing on time series," *Special Issue on Foundations of Information Processing of TELEMATIK*, vol. 8, no. 1, pp. 39–43, 2002.

[4] H. Jaeger, "A tutorial on training recurrent neural networks, covering bppt, rtrl, and the echo state network approach," Tech. Rep., Fraunhofer Institute for Autonomous Intelligent Systems, Sankt Augustin, Germany, 2002.

[5] J. J. Steil, "Backpropagation-decorrelation: online recurrent learning with O(N) complexity," in *Proceedings of IEEE International Joint Conference on Neural Networks (IJCNN '04)*, pp. 843–848, July 2004.

[6] B. Schrauwen, D. Verstraeten, and J. van Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks*, vol. 4, pp. 471–482, 2007.

[7] C. Fernando and S. Sojakka, "Pattern recognition in a bucket," in *Advances in Artificial Life*, Lecture Notes in computer Science, pp. 588–597, Springer, Berlin, Germany, 2003.

[8] D. Nquyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[9] M. Oubbati, M. Schanz, and P. Levi, "Kinematic and dynamic adaptive control of a nonholonomic mobile robot using a RNN," in *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '05)*, pp. 27–33, June 2005.

[10] M. Kawato, "Feedback-error-learning neural network for supervised motor learning," in *Advanced Neural Computers*, R. Eckmiller, Ed., pp. 365–372, Elsevier, Amsterdam, The Netherlands, 1990.

[11] M. I. Jordan and D. E. Rumelhart, "Forward models: supervised learning with a distal teacher," *Cognitive Science*, vol. 16, no. 3, pp. 307–354, 1992.

[12] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion in Neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[13] R. A. Løvlid, "Learning to imitate YMCA with an ESN," in *Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning (ICANN '12)*, Lecture Notes in Computer Science, pp. 507–514, Springer, 2012.

[14] R. A. Løvlid, "Learning motor control by dancing YMCA," *IFIP Advances in Information and Communication Technology*, vol. 331, pp. 79–88, 2010.

[15] A. Tidemann and P. Öztürk, "Self-organizing multiple models for imitation: teaching a robot to dance the YMCA," in *IEA/AIE*, vol. 4570 of *Lecture Notes in Computer Science*, pp. 291–302, Springer, Berlin, Germany, 2007.

[16] H. Jaeger et al., "Simple toolbox for esns," 2009, http://reservoir-computing.org/software.

[17] F. Toutounian and A. Ataei, "A new method for computing Moore-Penrose inverse matrices," *Journal of Computational and Applied Mathematics*, vol. 228, no. 1, pp. 412–417, 2009.

[18] F. Wyffels, B. Schrauwen, and D. Stroobandt, "Stable output feedback in reservoir computing using ridge regression," in *Proceedings of the 18th International Conference on Artificial Neural Networks, Part I (ICANN '08)*, pp. 808–817, Springer, 2008.

[19] H. Jaeger, "The echo state approach to analysing and training recurrent neural networks," Tech. Rep., GMD, 2001.