LETTER

RNN with a Recurrent Output Layer for Learning of Naturalness

Ján Dolinský and Hideyuki Takagi

Kyushu University, 4-9-1 Shiobaru, Minami-ku, Fukuoka, 815-8540 JAPAN
E-mail: jan@plazma.sk, takagi@design.kyushu-u.ac.kr

(Submited on December 12, 2007)

Abstract – The behavior of recurrent neural networks with a recurrent output layer (ROL) is described mathematically and it is shown that using ROL is not only advantageous, but is in fact crucial to obtaining satisfactory performance for the proposed naturalness learning. Conventional belief holds that employing ROL often substantially decreases the performance of a network or renders the network unstable, and ROL is consequently rarely used. The objective of this paper is to demonstrate that there are cases where it is necessary to use ROL. The concrete example shown models naturalness in handwritten letters.

Keywords – recurrent output layer, RNN, ESN, naturalness learning, handwritten letters

1. Introduction

In engineering, recurrent neural networks (RNN) have not been often proposed as a promising solution. The difficulties with training a RNN have been overcome, and recent theoretical advances in the field have made training a RNN quicker and easier [6]. Recurrent connections have not been found to increase the approximational capabilities of the network [1]. Nevertheless, we may obtain decreased complexity, network size, etc. while solving the same problem.

In some applications - such as speech recognition or object detection or prediction - our classification / prediction at time t should be more accurate if we can account for what we saw at earlier times. The most common approach to model such systems is to use a suitably powerful feed-forward network and feed it with a finite history of the inputs and optionally the outputs through a sliding window. Early attempts to improve this, often tedious, technique resulted in various network architectures based on the feed-forward topology with the recurrent connections being set to fixed values to ensure that the backpropagation rule can be used [7][2]. Many works have been done on autonomous Hopfield networks as well as on training algorithms that can be applied to a RNN in a feed-forward fashion (i.e. BPTT). Recent theoretical advances in RNN research such as Echo State Networks (ESN) afford the modeling of fully general topologies, which were difficult to train directly with the former techniques.

An interesting example is a topology where output units have connections from not only the internal units but also the input units and output units, yielding a recurrent output layer - ROL. Although connections from the input units are often used, connections from the output layer are rare. In the following chapters, we explain what behavior ROL implies, introduce the concept of our proposed naturalness learning, and show that using ROL not only increases the performance but is actually an intrinsic part of modeling with the proposed naturalness learning.

One of the earliest RNN, where the output activation values from the previous step were used to compute the output activation values in the next step, was the Jordan network [2][3]. In the Jordan network, the activation values of the output units are fed back into the input layer through a set of extra input units called the state units. This type of network is called output-recurrent network. Various modifications to output-recurrent networks have been proposed and successfully used for modeling difficult non-linear tasks [8]. RNN with ROL, in contrast to output-recurrent network, uses the output activation values of the previous step directly to compute the output in the next step. The output activation values of the previous step can be thought of as a extra hidden units. We are aware of no papers discussing applications of RNN with ROL.

2. Dynamics of RNN with a Recurrent Output Layer

Adopting a standard perspective of system theory, we view a deterministic and discrete-time dynamical system as a function **G** which yields the next system output, given an input and the output history:

$$\mathbf{y}(n+1) = \mathbf{G}(..., \mathbf{u}(n), \mathbf{u}(n+1); ...\mathbf{y}(n-1), \mathbf{y}(n)) \tag{1}$$

where $\mathbf{u}(n)$ is the input vector and $\mathbf{y}(n)$ is the output vector for the time step n.

The echo-state approach enables us to approximate systems represented by **G** directly, without the necessity to convert a time series into static input patterns by the sliding window technique [4].

Consider a discrete-time ESN [6] consisting of K input units with an activation vector $\mathbf{u}(n) = (u_1(n), ..., u_K(n))^t$, N internal units with an activation vector $\mathbf{x}(n) = (x_1(n), ..., x_N(n))^t$, and L output units with an activation vector $\mathbf{y}(n) = (y_1(n), ..., y_L(n))^t$, where ^{t} denotes the transpose. The corresponding input, internal and output connection weights are collected in the $N \times K$, $N \times N$, $L \times (K + N + L)$ weight matrices $\mathbf{W}^{in}$, $\mathbf{W}$, $\mathbf{W}^{out}$ respectively. Optionally, a $N \times L$ matrix $\mathbf{W}^{back}$ may be used to project the output units back to the internal units.

The internal units' activation is computed according to

$$\mathbf{x}(n+1) = \mathbf{f}(\mathbf{W}^{in}\mathbf{u}(n+1) + \mathbf{W}\mathbf{x}(n) + \mathbf{W}^{back}\mathbf{y}(n)) \tag{2}$$

where **f** denotes the component-wise application of the transfer (activation) function to each internal unit. The output is computed as

$$\mathbf{y}(n+1) = \mathbf{f}^{out}(\mathbf{W}^{out}(\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n)) \tag{3}$$

where $(\mathbf{u}(n+1), \mathbf{x}(n+1), \mathbf{y}(n))$ represents the concatenated vector consisting of input, internal and output activation vectors. The concatenated vector often consits only of input and internal activations or internal activation only. Fig. 1 shows the architecture of an ESN. See [6] for further details concerning the training of ESN.

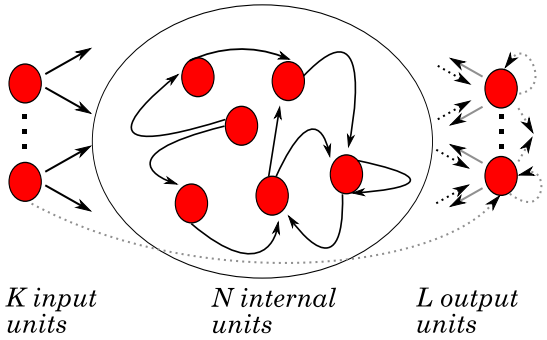

Figure 1: Echo-state network: the dotted lines plot connections which can be trained, the gray lines plot connections which are optional.

A closer look at Eq. (3) reveals that a system output $\mathbf{y}(n+1)$ is constructed from the given input and output history via two distinct mechanisms: from the activation vectors of the internal units $\mathbf{x}(n)$ indirectly (by computing $\mathbf{x}(n+1)$ via Eq. (2)) and optionally from the activation vectors of the input units $\mathbf{u}(n+1)$ and/or output units $\mathbf{y}(n)$ directly.

The internal units' activation $\mathbf{x}(n+1)$ is computed using the input and output activation $\mathbf{u}(n+1), \mathbf{y}(n)$ and the activation $\mathbf{x}(n)$ of the internal units from the previous step which recursively reflects the influence of input and output activations from previous steps. We can therefore rewrite Eq. (2) as

$$\mathbf{x}(n+1) = E(..., \mathbf{u}(n), \mathbf{u}(n+1); ..., \mathbf{y}(n-1), \mathbf{y}(n)) \tag{4}$$

where E depends on the history of input signal **u** and the history of desired output signal **y** itself, thus in each particular task, E shares certain properties with the desired output and/or given input. How strongly the internal activation $\mathbf{x}(n+1)$ is influenced by the activations $\mathbf{u}(n+1)$, $\mathbf{y}(n)$ and $\mathbf{x}(n)$ (which recursively consist of previous

input/output activations) is controlled by the size of the weights in matrices $\mathbf{W}^{in}, \mathbf{W}^{back}$ and $\mathbf{W}$ respectively. Algebraic properties of the matrix $\mathbf{W}$ are particularly important for the short-term memory property of an ESN [5].

Besides using the activations of internal units, sometimes it is advantageous to use also the activations of input and output units directly. Although the activation vector $\mathbf{x}(n)$ reflects the history of the desired output and/or given input, the activation vectors $\mathbf{u}(n+1), \mathbf{y}(n)$ in Eq. 3 are used merely as another form of input. This usage corresponds to connecting the input units to the output units and output units to output units themselves directly. Direct connection of input units to output units is often used whereas direct connection of output units to output units is rare. It is the connecting of output units to each other what makes the output layer recurrent.

ROL implies a substantial influence of the previously generated output $\mathbf{y}(n)$ on the successive output $\mathbf{y}(n+1)$. The activation $\mathbf{y}(n)$ is only an approximation of a system output at the step n and, thus, it is always generated with a certain error. This error is included in the computation of the successive output activation $\mathbf{y}(n+1)$ and can easily accumulate with each update step. It is for this reason that computation using ROL has been rare.

3. RNN with a Recurrent Output Layer for Naturalness Learning

In this section, we will demonstrate the principles of naturalness learning by showing how to express and model the unique quality of hand-written letters. We also explain why ROL works well with the naturalness learning.

The style of writing of any given person is very much individual and can be distinguished easily from mechanically or electronically printed text. Moreover, we can distinguish between the writing of different people. Everybody learns their alphabet in a school, and while writing in one's own individual way, a person is trying to approximate the shapes of the letters as they learned them in school. We can therefore understand handwriting as turning the basic shape of a letter as learned in a school into the writer's particular, unique, handwritten form. A human then can be seen as a 'filter' which adds his or her own characteristics to the shapes of those basic letters.

To explain the term of *naturalness*, first we must define our terminology. Let us refer to the letters used in textbooks (either printed or cursive) as the *font letters* (*fontL*). We view handwriting as the process of turning a *fontL* into its handwritten form. The unique quality of the handwritten letter (*handL*) can be then understood as the difference between the *handL* and the *fontL* and expressed as a 2-D displacement vector field of evenly spaced points along the strokes of the font and its respective handwritten version (see Fig. 2) [1]. We refer to this difference as *naturalness*. In other words, adding the *naturalness* to the *fontL* will result in a *handL* of a unique form. We can thus, assume a relation between the *fontL* and the *naturalness*.

handwritten letter = *font letter* + *naturalness*

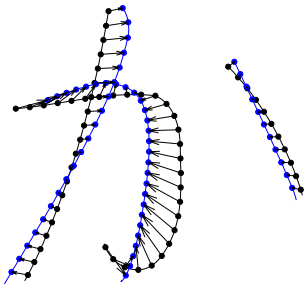

Figure 2: Hiragana letter /ka/. Naturalness expressed by 2-D displacement vector field. Font letter strokes shown in black, handwritten strokes shown in blue.

This assumption enables us to model *naturalness* by a system which employs certain characteristics of the *fontL* as its input.

The task of naturalness learning is to find and learn the relation between font letters and naturalness. Speaking in the terms of naturalness learning, the target system (handwritten letters) resembles the basic system (font letters) with its behavior (shape of handwritten letters) deviating from the basic system to a certain extent.

[1] A displacement vector is not the only mean of expressing naturalness, it can be expressed using an arbitrary mechanism.

Learning and modeling naturalness using a RNN with ROL produced interesting results. As mentioned in section 2., computing with ROL is prone to the accumulation of error from previous steps. This phenomena has been found harmful in many tasks. With naturalness learning, however, ROL performs well.

In the handwriting task, we found that introducing an activation $\mathbf{y}(n)$ into play via update Eq. (3) always helped network to generate $\mathbf{y}(n+1)$ with a greater accuracy. An intuitive explanation is as follows. The way a person writes the first half of a handwritten stroke influences, to a certain extent, how the second part is going to look, i.e. distortion of a certain part of a stroke usually implies some other distortion to a successive part of the stroke. It is therefore reasonable to assume that a short sequence of points on a handwritten stroke influences where the next point will appear, with the very last point of a sequence having the greatest influence. The same holds for the naturalness extracted as a difference between *handL* and *fontL*. On the other hand, it is not only the very last point which influences the position of the next point of a stroke, but a short sequence of the previous points. Thus, a backprojection matrix $\mathbf{W}^{back}$ was used so as to ensure recent short sequence of generated output $..., \mathbf{y}(n-1), \mathbf{y}(n)$ is also reflected in $\mathbf{y}(n+1)$ via the activation $\mathbf{x}(n+1)$ (see Eq. (2) and Eq. (3)).

The same principle holds for the input activations $\mathbf{u}(n+1)$ extracted from *fontL* strokes. Update Eq. (2) ensures that a short sequence $..., \mathbf{u}(n), \mathbf{u}(n+1)$ is reflected in the activation $\mathbf{x}(n+1)$ which is in turn used to compute $\mathbf{y}(n+1)$ via update Eq. (3). The activation $\mathbf{u}(n+1)$ is also used directly in Eq. (3) to ensure that the very last point of the input sequence has significant influence in the computation of $\mathbf{y}(n+1)$.

The fact that naturalness is being modeled, instead of the target system, allows us to control the amount of the naturalness being added to the font letters. A weight of value 1 will render generated letters as close to a person's handwriting as possible. The value of, say, 0.6 will reduce the naturalness to 60%, providing us with a neat handwritten letters. Values close to 0 will render generated letters very close to the font letters. It is also possible to combine several individual's naturalness (e.g. 40% of person A's naturalness with 60% of person B's naturalness).

The naturalness learning approach is not limited to the handwriting task only. We believe, one can generate natural looking movements for parts of the human body with the inverse kinematics algorithm being employed as the basic system. Generating emotional human speech with synthesized speech being used as the basic system might be another interesting application of the naturalness learning approach.

4. Experiments

In this chapter, we demonstrate how the naturalness learning approach copes with the handwriting task. The letters used in the experiments are symbols of Japanese syllabary - hiragana.

The *fontL* were extracted from the Bitstream Vera Sans font onto 250×250 pixel canvas. Every stroke of a given letter was turned into a set of bezier curves - a path. Every path was then evenly spaced into a set of points. Let each $\mathbf{P}_{ij}^{fl}$ be the set of points represented by $n_{ij} \times 2$ matrix where i denotes the index of the letter, j the index of the stroke within the letter and n_{ij} is the number of points. $\mathbf{P}_{ij}^{fl}(k) = (x_k, y_k)$ therefore represents the k-th point of the j-th stroke of the i-th letter (k-th row of the matrix $\mathbf{P}_{ij}^{fl}$).

The *handL* were first scanned and then appropriately scaled so as to ensure they fit the canvas. Every *handL* stroke was then aligned to its *fontL* stroke counterpart. This alignment ensures that the displacement vector field expresses only a shape transformation between a pair of *fontL* and *handL* strokes. In order to split strokes of *handL*, the same procedure was applied as with the *fontL*, saving the points for each stroke into $\mathbf{P}_{ij}^{hl}$. The spacing interval along each stroke (path) of a *handL* was, however, also adjusted so that number of points matched those in the corresponding *fontL* stroke.

4.1 Data specification

The input signals were extracted from the points of *fontL*'s strokes as follows. Let $\mathbf{D}_{ij}^{fl}(k)$ be the difference vector $\mathbf{P}_{ij}^{fl}(k+1) - \mathbf{P}_{ij}^{fl}(k)$. Then the inertia for the k-th point of the j-th stroke of the i-th letter is given by

$$\mathbf{inertia}_{ij}(k) = \mathbf{D}_{ij}^{fl}(k) \tag{5}$$

with each $\mathbf{inertia}_{ij}(k)$ being the k-th row of the $(n_{ij}-1) \times 2$ matrix $\mathbf{inertia}_{ij}$. The inertia can be thought of as a representation of the movement of an imaginary pen which 'wrote' the font letter. The curvature for the k-th point

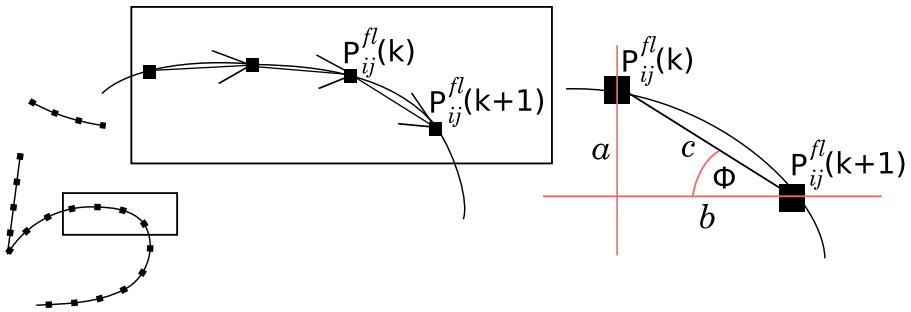

Figure 3: Geometrical meaning of the input data. $inertia_{ij}(k)$ represents difference vector between two successive points $P_{ij}^{fl}(k+1), P_{ij}^{fl}(k)$. $curv_{ij}(k)$ is the sine of angle ϕ.

of the j-th stroke of the i-th letter is given by

$$\mathbf{curv}_{ij}(k) = \frac{\mathbf{D}_{ij}^{fl}(k) \begin{pmatrix} 0 \\ 1 \end{pmatrix}}{\sqrt{\mathbf{D}_{ij}^{fl}(k)\mathbf{D}_{ij}^{fl}(k)^t}} \tag{6}$$

with each $\mathbf{curv}_{ij}(k)$ being the k-th row of the $(n_{ij}-1)\times 1$ matrix $\mathbf{curv}_{ij}$. Fig. 3 illustrates the geometrical meaning of Eq. (5) and (6). Each matrix $\mathbf{curv}_{ij}$ and $\mathbf{inertia}_{ij}$ was merged into a single $(n_{ij}-1)\times 3$ matrix $\mathbf{U}_{ij}$ with each column being normalized into the interval $(-1,1)$. In order to partially erase the transient dynamics from the previous stroke, a zero sequence of size $n_{gap}\times 3$ was inserted before every $\mathbf{U}_{ij}$, resulting in the final input matrix $\mathbf{U}$.

Naturalness, which serves as the (2 dimensional) output signal, is represented by 2-D displacement vector field. The 2-D displacement vector field for the j-th stroke of the i-th letter is given by

$$\mathbf{Y}_{ij} = \mathbf{P}_{ij}^{hl} - \mathbf{P}_{ij}^{fl} \tag{7}$$

with $\mathbf{Y}_{ij}$, $\mathbf{P}_{ij}^{hl}$ and $\mathbf{P}_{ij}^{fl}$ each being $n_{ij}\times 2$ matrices. The last row of every $\mathbf{Y}_{ij}$ was dropped to ensure each pair of $\mathbf{Y}_{ij} and \mathbf{U}_{ij}$ have the same length of $n_{ij}-1$. Each column of $\mathbf{Y}_{ij}$ was scaled down to the interval $(-1,1)$. A zero sequence of size $n_{gap}\times 2$ was inserted before every $\mathbf{Y}_{ij}$, resulting in the final output matrix $\mathbf{Y}$.

4.2 Network parameters

A 300 unit network was used with activation function $\mathbf{f} = RBF$. Internal weights in the matrix $\mathbf{W}$ were randomly assigned values of 0, 0.2073, -0.2073 with probabilities 0.95, 0.025, 0.025. For a 300×300 matrix $\mathbf{W}$, this implies a spectral radius of ≈ 0.85, providing for a relatively long short-term memory [6]. 3 input and 2 output units were attached. Input weights were randomly drawn from a uniform distribution over $(-1,1)$ with probability 0.9 or set to 0. With such an input matrix, the network is strongly driven by the input activations because many elements of the matrix $\mathbf{W}^{in}$ are non-zero values. The network had output feedback connections, which were randomly set to one of the three values of 0, 0.1, -0.1 with probabilities 0.9, 0.05, 0.05. Such a setup of feedback connections makes the network excited only marginally by previous output activations. The activation function for the output units was identity $\mathbf{f}^{out}(x) = x$.

4.3 Training and testing

The training data was made from the letters shown in Fig. 4, resulting in a 3027×3 input matrix for $\mathbf{U}_{train}$ and a 3027×2 output matrix for $\mathbf{Y}_{train}$, which were prepared according to the data specification with $n_{gap} = 16$ being used. Eq. (2) was used for updating with $\mathbf{u}(n), \mathbf{y}(n)$ being the transposed n-th rows of matrices $\mathbf{U}_{train}$ and $\mathbf{Y}_{train}$ respectively. The first 300 steps were discarded and the network internal states with input unit states $\mathbf{x}(n), \mathbf{u}(n)$ were collected from $n = 301$ through $n = 3355$. The output weights $\mathbf{W}^{out}$ were computed using the internal

states and input unit states only. The training errors of first and second output unit were $mse_{train,1} \approx 9.5 \times 10^{-4}$ and $mse_{train,2} \approx 2.8 \times 10^{-3}$ respectively. Making the output layer recurrent (ROL) and computing $\mathbf{W}^{out}$ using not only internal/input states $\mathbf{x}(n), \mathbf{u}(n)$ but also output activation values $\mathbf{y}(n-1)$ reduced the training errors $mse_{train,1}$ and $mse_{train,2}$ down to $\approx 6 \times 10^{-4}$ and $\approx 2.0 \times 10^{-3}$ respectively. A visual comparison is shown in Fig. (4).

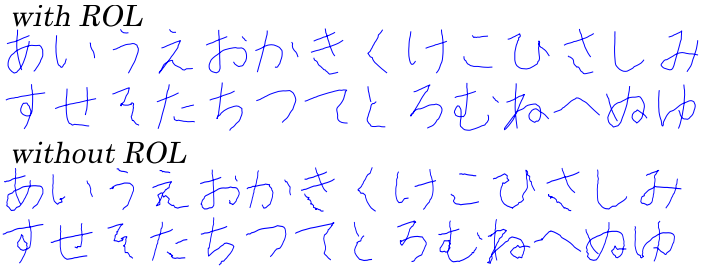

Figure 4: Testing with the training data: Letters produced by RNN with/out ROL with teacher-forcing switched off from $n = 301$.

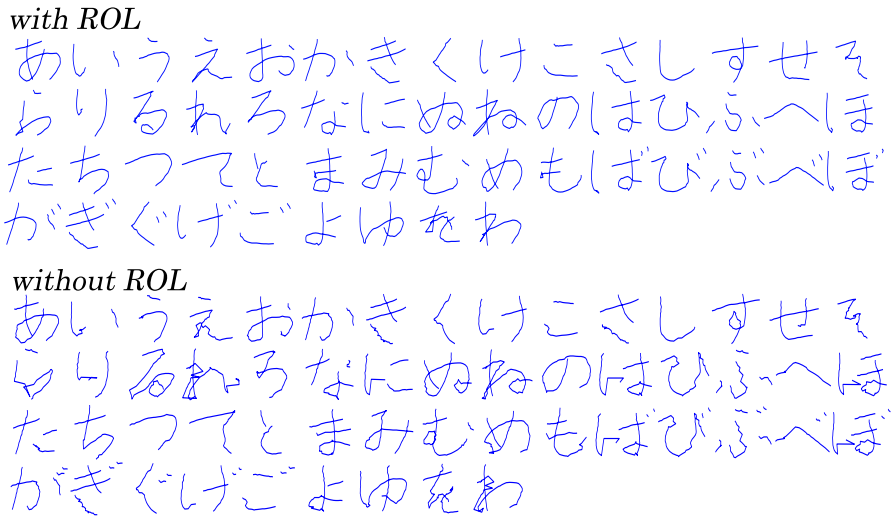

Figure 5: Testing with the testing data: Letters produced by RNN with/out ROL

The testing data was made from letters shown in Fig. 5, resulting in a 6045×3 input matrix $\mathbf{U}_{test}$ and a 6045×2 output matrix $\mathbf{Y}_{test}$ with $n_{gap} = 16$ being used. For non-ROL topology the test errors were found to be $mse_{test,1} \approx 1.2 \times 10^{-2}, mse_{test,2} \approx 3.5 \times 10^{-2}$. Using ROL reduced the test errors to $mse_{test,1} \approx 0.9 \times 10^{-3}, mse_{test,2} \approx 3.0 \times 10^{-2}$. The errors $mse_{test,i}$ provide only a rough indication of network performance. A visual comparison between these two trials is shown in Fig. 5. We can observe that the network also produces appropriate naturalness for letters on which it had not been trained.

5. Analysis of ROL Dynamics

5.1 Robustness

In the previous chapter, we showed that ROL can enable better performance. The improved performance was demonstrated numerically and visually. Fig. 5 shows that the performance boost provided by ROL is essential for the handwriting task. In this section, we analyze the dynamics of RNN with ROL in more detail to determine whether ROL also performs better with different network configurations. The weight matrices of the network

are produced in a pseudo-random fashion and thus, even the same configuration (i.e. the configuration from section 4), if reinitialized, produces slightly different results. We therefore test each configuration several times with reinitialized weights. All experiments were carried out using the same data as in section 4, with the exception of the configuration which uses no internal units[2].

In order to better understand the impact of ROL on RNN performance, a setup with no internal units was tested first. This configuration reveals the importance of the output from previous time step (=ROL) more clearly due to the lack of involvement from any internal states. Further, this setup also demonstrates the importance of the internal states themselves. For the non-ROL setup, the output weights $\mathbf{W}^{out}$ were computed using the input unit states $\mathbf{u}(n)$ only. This boils down to a regression problem with three predictor variables (input activations) and two response variables (output activations). For the ROL setup, the output weights $\mathbf{W}^{out}$ were computed using the input states $\mathbf{u}(n)$ and output activations $\mathbf{y}(n-1)$. This is equivalent to a regression problem with five predictor variables (input activations and output activations from the previous step) and two response variables (output activations). To better evaluate the performance of the models, we also computed the correlations between the teacher signals and the produced output signals. The results are given in the Table 1. Poor training performance of the non-ROL setup shows that the input activations $\mathbf{u}(n)$ alone are not sufficient for computing the output $\mathbf{y}(n)$ and suggests that nonlinear expansion of the input activations into a space of higher dimension might be necessary. Using the output activations $\mathbf{y}(n-1)$ from the previous step (ROL configuration) decreased the training error and increased the correlations significantly. This result indicates that the activation $\mathbf{y}(n-1)$ is vital for computing the successive output $\mathbf{y}(n)$. The non-ROL performed roughly the same on the testing data as it did on the training data. In contrast, the ROL configuration performed significantly worse on the testing data than it did on the training data. This drop in performance can be attributed to the fact that during the testing phase the ROL model could not generate an appropriate output which, in turn, could be used to compute the next output. Relatively high testing errors and low correlations for both setups suggest that nonlinear expansion (=internal state, Eq. (2) of both input activations $\mathbf{u}(n)$ and output activations $\mathbf{y}(n-1)$ is necessary to achieve better performance. Comparisons of performance with testing data for both configurations shows that even in this trivial case ROL performs slightly better[3].

Table 1: Training and testing errors and correlations for configuration with no internal units. See text for details.

	train				test			
	$mse_{1,2}$		$corr_{1,2}$		$mse_{1,2}$		$corr_{1,2}$	
non-rol	0.0278	0.0247	0.1761	-0.0525	0.0272	0.0216	0.0989	0.0004
rol	0.0029	0.0025	0.9477	0.9466	0.0307	0.0212	0.2480	0.0770

In the ESN approach, every initialization of a network produces slightly different weight matrices, and thus, it is common practice to run the same setup several times and keep the network which performed the best. In order to find out whether ROL is superior to non-ROL in general or only for a particular configuration with particular weights, four different setups were tested. Each configuration was tested 25 times with and without ROL, and the network was reinitialized for each new trial. In total 100 trials were carried out. Within each trial, the same network was used without reinitialization of any weights so as to ensure that having or not having ROL was the only difference between the compared networks.

Configuration 1 is identical to the one introduced in section 4. For configuration 2, the spectral radius of matrix $\mathbf{W}$ was set to 0.95. All remaining parameters were identical to configuration 1. For configuration 3, the spectral radius of matrix $\mathbf{W}$ was set to 0.75 with all remaining parameters being identical to configuration 1. Finally, for configuration 4, the update function for the internal units was set to $tanh$. All remaining parameters were identical to configuration 1. The results are given in the Table 2. As we can see, for each configuration, ROL performed better. Fig. 6, 7 and 8 illustrate visually the results of randomly selected trials of configurations 2, 3 and 4, respectively. Visual comparison of the results obtained using configuration 1 is possible using Fig. 4 and 5. The superior performance of ROL is clearly visible for configurations 1, 2 and 3 (Fig. 5, 6 and 7). With configuration 4, the difference between the performance of the configuration with and without ROL might not be obvious at

[2]The configuration with no internal units is equivalent to a regression problem with no short-term memory, and thus, the data were prepared according to the data specifications with $n_{gap} = 0$ being used.

[3]Comparable testing errors $mse_{test,i}$ but higher correlations $corr_{test,i}$ for ROL configurations (i = 1,2).

first look. Configuration 4 without ROL was often found, however, to produce misshapen letters and consequently higher $mse_{test,i}$.

Throughout the 100 trials, some trials where ROL $mse_{test,i}$ was very similar or slightly higher than nonROL $mse_{test,i}$ were also found to occur. Visual comparison of these few cases nevertheless confirmed superiority of ROL; ROL configurations always produced outputs that were visually more appealing.

Stability of configurations that use ROL is another important issue. Here, the spectral radius of the internal weights matrix $\mathbf{W}$ was found to be of special significance. Configurations with spectral radius higher than 0.8 were found to very often generate stable solutions. Changing the spectral radius to lower values was found to increase the number of non-stable solutions. Networks which exhibited non-stable behavior with ROL usually produced numerically stable solutions without ROL, but the testing errors of such networks was rather high and their visually evaluated performance was rather poor. On the other hand, cases where networks exhibited stable behavior with ROL but non-stable without were also found to occur. Neither of the above cases were included in the averaged results shown in the Table 2 because the high testing errors of those trials would significantly bias the results towards ROL or nonROL configurations.

In general, ROL setups were found to perform better numerically (lower mse) and visually (they produced visually more appealing characters). Concerning stability, the spectral radius of internal weights matrix $\mathbf{W}$ was found to be of significance, and the existence/non-existence of ROL has not found to have any impact on stability.

Table 2: Errors and correlations for training and testing averaged over 25 trials for configurations 1,2,3 and 4. See text for details.

configuration		train				test			
		$mse_{1,2}$		$corr_{1,2}$		$mse_{1,2}$		$corr_{1,2}$	
1	*non-rol*	7.90×10^{-4}	2.48×10^{-3}	0.979	0.978	2.22×10^{-2}	6.08×10^{-2}	0.450	0.410
	rol	5.79×10^{-4}	1.97×10^{-3}	0.985	0.983	1.15×10^{-2}	4.18×10^{-2}	0.574	0.446
2	*non-rol*	1.02×10^{-3}	2.89×10^{-3}	0.973	0.975	1.66×10^{-2}	4.18×10^{-2}	0.504	0.469
	rol	6.03×10^{-4}	2.18×10^{-3}	0.984	0.981	1.18×10^{-2}	3.87×10^{-2}	0.568	0.476
3	*non-rol*	6.85×10^{-4}	2.10×10^{-3}	0.982	0.982	1.86×10^{-2}	6.33×10^{-2}	0.510	0.418
	rol	5.48×10^{-4}	1.75×10^{-3}	0.985	0.985	1.19×10^{-2}	4.23×10^{-2}	0.568	0.450
4	*non-rol*	1.18×10^{-3}	2.34×10^{-3}	0.968	0.980	2.25×10^{-2}	4.39×10^{-2}	0.371	0.407
	rol	1.11×10^{-3}	2.05×10^{-3}	0.970	0.982	1.25×10^{-2}	3.91×10^{-2}	0.483	0.421

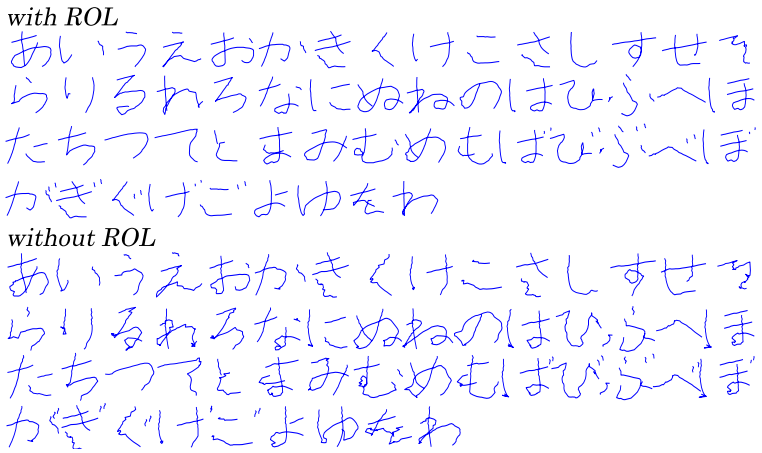

Figure 6: Setup 2

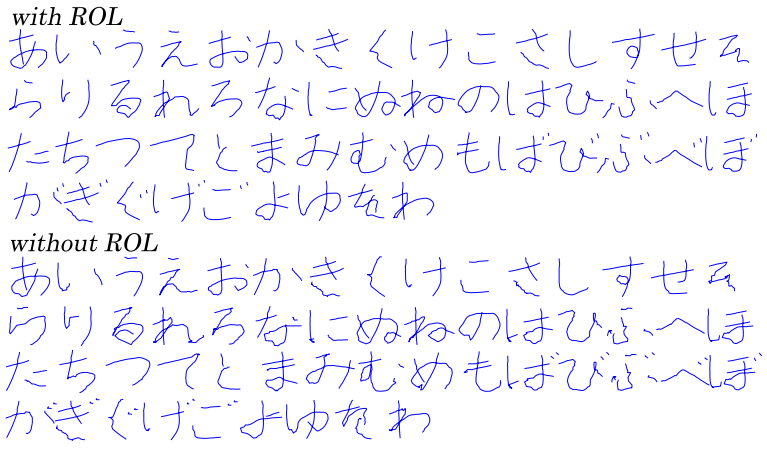

Figure 7: Setup 3

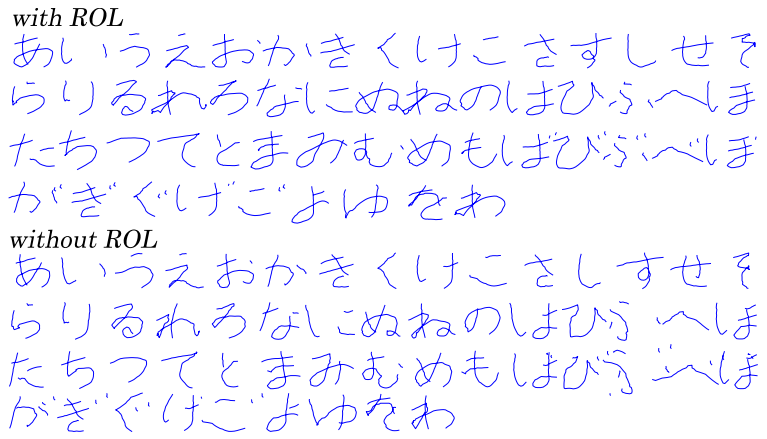

Figure 8: Setup 4

5.2 ROL Weights

In order to determine why ROL performs better, we observed the sizes of ROL weights and analyzed which ROL weights contribute the most to the final output. ROL weights were extracted from the output weights of the one hundred networks generated in the previous section. Two output units were used in all the experiments, yielding four ROL weights $w_{1,1}$, $w_{1,2}$, $w_{2,1}$ and $w_{2,2}$ (Fig. 9). To see how the weights vary between each trial, the multivariate probability density function (PDF) of the four variables $w_{1,1}$, $w_{1,2}$, $w_{2,1}$, $w_{2,2}$ was estimated. The mean vector and covariance matrix is given by Eq. 8. The diagonal elements of Σ contain the variances for each variable. Low variances for the weights $w_{1,1}$, $w_{1,2}$, $w_{2,1}$ indicate that their values in each trial were very close to their mean values. The higher variance value for weight $w_{2,2}$ was caused by some samples from configuration 4. In all one hundred trials, a pattern could be observed wherein there were low $w_{1,2}$, $w_{2,1}$ and significantly higher $w_{1,1}$, $w_{2,2}$. This finding indicates that the activation values $\mathbf{y}_1(n)$ and $\mathbf{y}_2(n)$ are strongly related to their respective successors $\mathbf{y}_1(n+1)$ and $\mathbf{y}_2(n+1)$. In contrast, the relationship between $\mathbf{y}_1(n)$ and $\mathbf{y}_2(n+1)$ (or vice-versa) was found to be of little importance. A plausible explanation is that x component of the naturalness in step n is strongly related to the x component in the successive step $n+1$. Similarly, y component of the naturalness in step n is strongly related to the y component in the successive step $n+1$. This strong relationship is the reason why ROL enables better numerical performance. Moreover, ROL also works as a smoother, and thus, even when the mse of ROL and nonROL configurations are very similar, ROL always produces results that are visually more appealing.

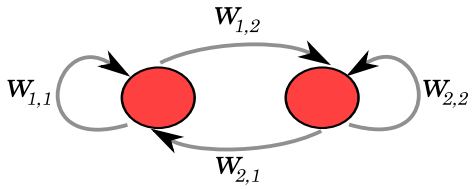

Figure 9: ROL weights

$$\mu = \begin{pmatrix} 1.1591 \\ -0.0160 \\ -0.0702 \\ 1.64830 \end{pmatrix} \quad \mathbf{\Sigma} = \begin{pmatrix} 0.0238 & -0.0063 & -0.0017 & 0.0317 \\ -0.0063 & 0.0674 & -0.0023 & 0.0070 \\ -0.0017 & -0.0023 & 0.0099 & -0.0159 \\ 0.0317 & 0.0070 & -0.0159 & 0.1966 \end{pmatrix} \tag{8}$$

6. Discussion

6.1 Network

Here we try to provide an insight into why the configuration from section 4. works the best. The network has information concerning the shape of the strokes in *fontL* and *handL* via its history of input and output activations. The matrices $\mathbf{W}^{in}$ and $\mathbf{W}$ from the configuration in section 4. make the network strongly driven by the (finite) history of the input. Moreover, using the most recent input activation $\mathbf{u}(n+1)$ in Eq. (3) directly makes the impact even stronger. It is most likely the case that the last several points of the path from where a *fontL* stroke is coming have a substantial influence on the naturalness and thus also on where the *handL* stroke is going to continue. An intuitive explanation is that a human, while writing in his or her own individual way, is trying to 'approximate' the shape of the letter of a font as memorized at school. This finding is in line with the basic idea underpinning naturalness learning: that a target system may be modeled (*handL*) by means of a basic system (*fontL*) and its difference (*naturalness*) with the target system.

The feedback weights $\mathbf{W}^{back}$ from the configuration in section 4. only slightly excite the network with the output history. Surprisingly, using the most recent output activation $\mathbf{y}(n)$ in Eq. (3) directly (= ROL) always improved the performance. A plausible explanation is that the *already written part* of a *handL* stroke influences how the *yet-to-be-written* part will look like, with the naturalness of the the previous step having highest relevance to the naturalness generated in the next step (i.e. a distortion in a certain part of a stroke usually implies some other distortion in a successive part of the stroke). Analysis of the ROL weights in section 5 confirm the above theory and show that the x component of the naturalness in the current step is strongly related to the x component of the naturalness generated in the next step. The same holds true for the y component.

Feedback connections with larger weights (up to +1) and no ROL were also tested. With this setup the network training error was about the same as in sec. 4, but driving the network with the testing data yielded worse performance or made the network unstable.

To verify the robustness of the advantage provided by ROL, we tested ROL with several different configurations (section 5). In all cases, ROL enabled better performance.

6.2 Data structure

The naturalness in this paper is represented as a 2-D displacement vector field (Fig. 2). We are, however, not strictly bound to this representation. The naturalness can be also represented as a set of parameters in a system which represents a difference between the target (*handL*) and the basic (*fontL*) system. The input data characterizing the basic system (*handL*) was made position independent so as to ensure the same stroke will be represented by the same data regardless of its starting position. This reduces significantly the complexity of the handwriting task because while separate strokes are substantially different in shape, some of their small parts are often similar. The short-term memory of a RNN makes distinction between a identical/similar short stroke sequences possible because the RNN accounts also for points before the identical/similar stroke part.

7. Conclusion

In the handwriting task we showed that by modeling the target system by means of a basic system and its difference from the target system, a substantial relevance is revealed in the difference produced in step n and step $n+1$. In such a case the usage of a ROL turned out to be advantageous.

We would like to confirm these findings by applying naturalness learning to other tasks as well. Modeling the unique individualistic quality of human motion is the next step in confirming the feasibility of both naturalness learning and the usability of a ROL for tasks formulated in terms of naturalness learning.

Acknowledgment: This study was supported in part by Grant-in-Aid for the 21st Century COE program.

References

[1] Patrick van der Smagt Ben Krose. Wiley-Interscience, November 1996.

[2] Jordan M. I. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 531–546. Hillsdale, NJ: Erlbaum, 1986.

[3] Jordan M. I. Serial order: A parallel distributed processing approach. Tech. Report 8604, La Jolla, CA: Institute for Cognitive Science, University of California, May 1986.

[4] Herbert Jaeger. The 'echo-state' approach to analysing and training recurrent neural networks. GDM Report 148, Fraunhofer Institute for Autonomous Intelligent Systems, December 2001.

[5] Herbert Jaeger. Short term memory in echo state networks. GDM Report 152, Fraunhofer Institute for Autonomous Intelligent Systems, March 2002.

[6] Herbert Jaeger. Supervised training of recurrent neural networks, especially with esn approach. GDM Report 159, Fraunhofer Institute for Autonomous Intelligent Systems, October 2002.

[7] Elman J. L. Finding structure in time. *Cognitive Science*, (14):179–211, 1990.

[8] Ying-Chung Wang; Chiang-Ju Chien; Ching-Cheng Teng. Direct adaptive iterative learning control of nonlinear systems using an output-recurrent fuzzy neural network. In *IEEE Transactions on Systems, Man and Cybernetics-Part B : Cybernetics, volume 34, number 3*, pages 1348–1359. IEEE, June 2004.

Ján Dolinský received the Master degree from the Technical University of Kosice, Slovakia in 2004. From 2004 to 2005, he was a system engineer at the Oracle Corp., Prague, Czech Republic. He was granted honorary scholarship of the MEXT: Ministry of Education, Culture, Sports, Science and Technology, Japan in 2005 and is currently working towards the PhD degree at the Kyushu University. His research interests include complex and hyper-complex numbers in intelligent systems (NN, RNN), Clifford algebra, curse of dimensionality, computational biology, complex network theory and nonlinear estimation.

Hideyuki TAKAGI received the degrees of Bachelor, Master, and Doctor of Engineering from Kyushu Institute of Design (KID) in 1979, KID in 1981, and Toyohashi University of Technology in 1991. His professional career includes a researcher at the Central Research Labs of Matsushita Electric Industrial Co., Ltd. in 1981 - 1995, a Visiting Industrial Fellow at the UC Berkeley in 1991 - 1993, and an Associate Professor at KID in 1995 -2003 and at Kyushu University since 2003. Dr. Takagi is interested in the cooperative technology of neural networks, fuzzy systems, and evolutionary computation and is one of the most active researchers in interactive evolutionary computation area. He made significant contribution to the start of neuro-fuzzy system since 1988. Dr. Takagi received the Shinohara Memorial Young Engineer Award from Institute IEICE in 1989,

the Best Paper Awards from KES'97, IIZUKA'98 conferences in 1997 and 1998, the PC Best Paper Awards from ICOIN-15 conference in 2001, and Paper Award from Japan Society for Fuzzy Theory and Intelligent Informatics (SOFT) in 2004. He also received Distinguished Service Award, Outstanding Contribution Award, and Best Associate Editor Award from Slovak Artificial Intelligence Society in 2002, and IEEE SMC Society in 2003 and 2005, respectively.