RNN With a Recurrent Output Layer for Learning of Naturalness

Dolinsky, Jan
Graduate School of Design, Kyushu University

Takagi, Hideyuki
Faculty of Design, Kyushu University

RNN With A Recurrent Output Layer For Learning Of Naturalness

Jan Dolinsky and Hideyuki Takagi

Kyushu University, 4-9-1 Shiobaru, Minami-ku, Fukuoka, 815-8540 JAPAN

Abstract. The behaviour 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.

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 [4]. Recurrent connections have not been found to increase the approximational capabalities of the network [7]. 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 throught 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 [1][6]. 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 theorethical advances in RNN research such as Echo State Networks (ESN) afford the modelling 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. Althought connections from the input units are often used, connections from the output layer are rare. In the following chapters, we explain what behaviour 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 modelling 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 [6][5]. 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 modelling 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 $\mathbf{G}$ 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 $\mathbf{G}$ directly, without the necessity to convert a time series into static input patterns by the sliding window technique [2].

Consider a discrete-time ESN [4] 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 $\mathbf{f}$ 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

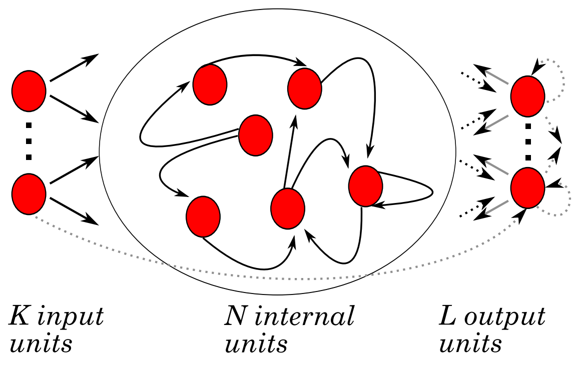

Fig. 1. Echo-state network: the dotted lines plot connections which can be trained, the gray lines plot connections which are optional.

consits only of input and internal activations or internal activation only. Fig. 1 shows the architecture of an ESN. See [4] for further details concerning the training of ESN.

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 $\mathbf{u}$ and the history of desired output signal $\mathbf{y}$ 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 particulary important for the short-term memory property of an ESN [3].

Besides using the activations of internal units, sometimes it is advantageous to use also the activations of input and output units directly. Althought 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*. 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 behaviour (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

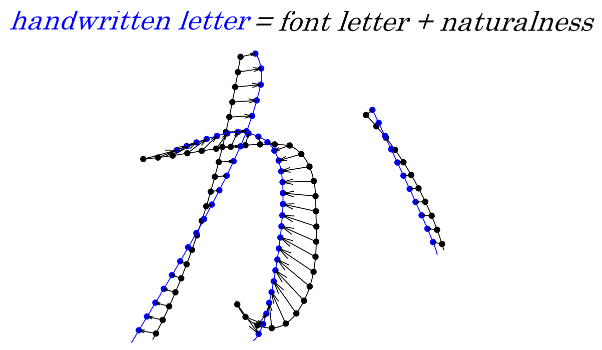

Fig. 2. Hiragana letter /ka/. Naturalness expressed by 2-D displacement vector field. Font letter strokes shown in black, handwritten strokes shown in blue.

Learning and modelling 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 compututation 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 250x250 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 aligment 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 of the j-th stroke of the i-th letter is given by

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

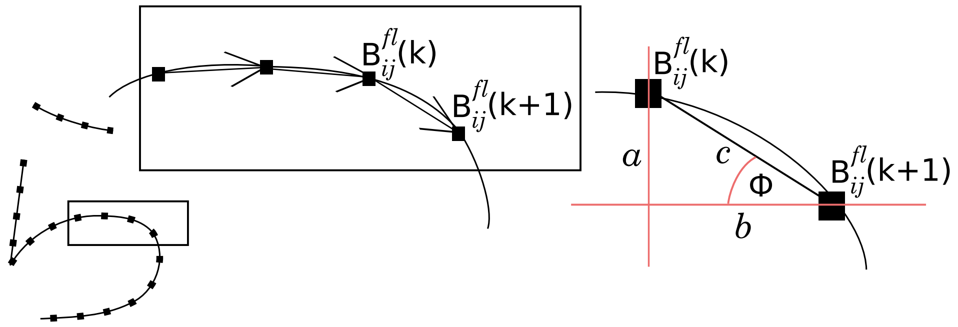

Fig. 3. Geometrical meaning of the input data. $inertia_{ij}(k)$ represents difference vector between two successive points $P_{ij}^{fl}(k+1), P_{ij}^{fl}(k)$. $curv_{ij}(k)$ is the sine of angle ϕ.

with each $\mathbf{curv}_{ij}(k)$ being the k-th row of the $(n_{ij}-1)\times 1$ matrix $\mathbf{curv}_{ij}$. Figure 3 illustrates the geometrical meaning of (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 connection weights in the matrix $\mathbf{W}$ were randomly assigned values of 0, 0.31, -0.31 with probabilities 0.98, 0.01, 0.01. For a 300×300 matrix $\mathbf{W}$, this implies a spectral radius of ≈ 0.8, providing for a relatively long short-term memory [4]. 3 input and 2 output units were attached. Input connection weights were randomly drawn from a uniform distribution over $(-1,1)$. 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 units 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 $mse_{test,2}$ down to $\approx 6 \times 10^{-4}$ and $\approx 2.0 \times 10^{-3}$ respectively. A visual comparison is shown in Fig. (4).

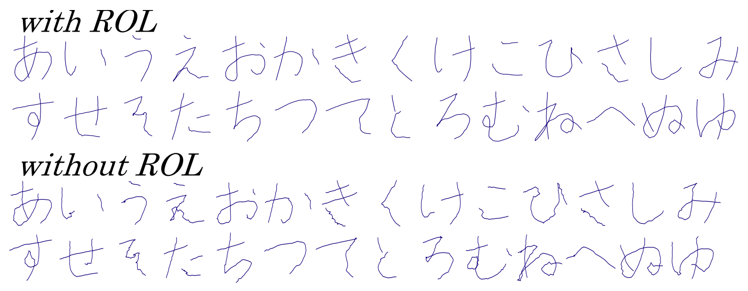

Fig. 4. Testing with the training data: Letters produced by RNN with/out ROL with teacher-forcing switched off from $n = 301$.

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 Discussion

5.1 Network

Here we try to provide an insight into why the setup 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

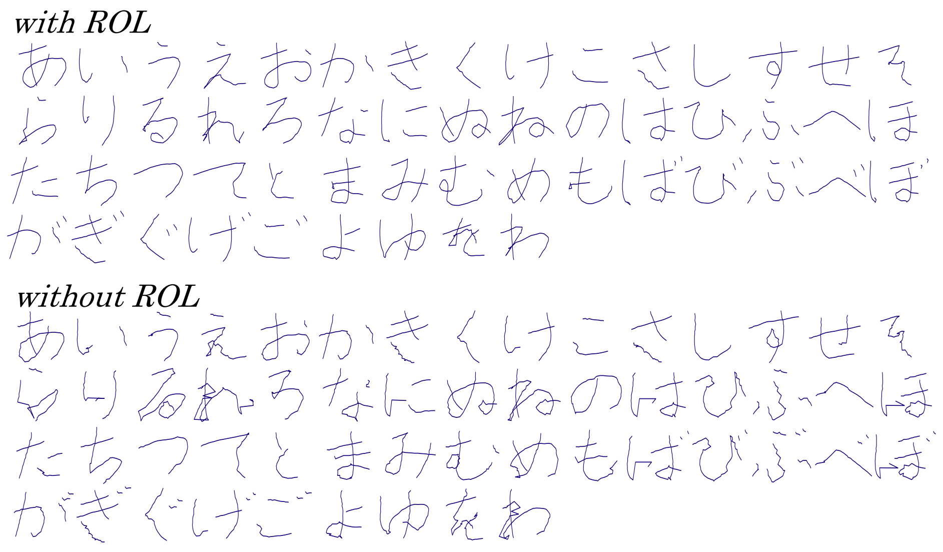

Fig. 5. Testing with the testing data: Letters produced by RNN with/out ROL

$\mathbf{W}^{in}$ and $\mathbf{W}$ from the setup in sec. 4. make the network strongly driven by the (finite) history of 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 has a substantial impact 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 one's own individual way, is trying to 'approximate' a font letter shape as memorized in a school. This finding is in line with the basic idea of the naturalness learning: to model a target system (*handL*) by means of a basic system (*fontL*) and its difference (*naturalness*) with the target system.

The feedback weights $\mathbf{W}^{back}$ from the setup in section 4. only slightly excite the network with the output history. Surpisingly, use of 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 *going 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. distortion of a certain part of a stroke usually implies some other distortion to a successive part of the stroke).

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 rendered worse performance or made the network unstable.

5.2 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 substantialy 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.

6 Conclusion

In the handwriting task we showed that by modelling 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. Modelling 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.

References

1. Elman J. L., “Finding structure in time,” Cognitive Science: A Multidisciplinary Journal, vol. 14, no.2, pp.179–211, 1990.
2. Jaeger, H., “The “echo state” approach to analysing and training recurrent neural networks,” Sankt Augustin: GMD-Forschungszentrum Informationstechnik, GMD-Report 148, Dec. 2001.
3. Jaeger, H., “Short term memory in echo state networks,” Sankt Augustin: GMD-Forschungszentrum Informationstechnik, GMD-Report 152, March 2002.
4. Jaeger, H., “Supervised training of recurrent neural networks, especially with ESN approach,” Sankt Augustin: GMD-Forschungszentrum Informationstechnik, GMD-Report 159, Oct. 2002.
5. Jordan, M. I., “Serial Order: a parallel distributed processing approach,” Tech. Rep. 8604, Univ. of California at San Diego, Inst. for Cognitive Science, May 1986.
6. Jordan, M. I., “Attractor dynamics and parallelism in a connectionist sequential machine,” Eighth Annual Conf. of Cognitive Science Society, Amherst, MA, USA, pp.531–546, August 1986.
7. Krose, B. and van der Smagt, P., “Recurrent networks,” Ch.5 in *An introduction to neural networks*, Eighth Edition, Univ. of Amsterdam, Nov. 1996.
8. Wang, Y-C, Chien, C-J, and Teng, C-C, “Direct adaptive iterative learning control of nonlinear systems using an output-recurrent fuzzy neural network,” IEEE Trans. on SMC-B, vol. 34, no.3 pp.1348–1359, June 2004.