(Accepted in Neural Networks - Elsevier, 2017)

How Can a Recurrent Neurodynamic Predictive Coding Model Cope with Fluctuation in Temporal Patterns? Robotic Experiments on Imitative Interaction

Ahmadreza Ahmadi[1] and Jun Tani*[1,2]

[1]Dept. of Electrical Engineering, KAIST, Daejeon, 305-701, Korea

[2]Cognitive Neurorobotics Research Unit, Okinawa Institute of Science and Technology Graduate University, 1919-1 Tancha, Onna-son, Kunigami-gun, Okinawa, Japan 904-0495

Abstract

The current paper examines how a recurrent neural network (RNN) model using a dynamic predictive coding scheme can cope with fluctuations in temporal patterns through generalization in learning. The conjecture driving this present inquiry is that a RNN model with multiple timescales (MTRNN) learns by extracting patterns of change from observed temporal patterns, developing an internal dynamic structure such that variance in initial internal states account for modulations in corresponding observed patterns. We trained a MTRNN with low-dimensional temporal patterns, and assessed performance on an imitation task employing these patterns. Analysis reveals that imitating fluctuated patterns consists in inferring optimal internal states by error regression. The model was then tested through humanoid robotic experiments requiring imitative interaction with human subjects. Results show that spontaneous and lively interaction can be achieved as the model successfully copes with fluctuations naturally occurring in human movement patterns.

*Corresponding author.

Keywords— neuro-robotics, predictive coding, recurrent neural networks, synchronized imitation, time-warping, error regression

1 Introduction

The principle of predictive coding suggests that organisms become able to predict perceptual outcomes due to current intentions for acting on the external environment, and to infer intentions behind perceptions themselves, via accumulated learning of perceptual experience through an agents own actions, the actions of other agents, and the consequences of these actions over time [1, 2, 3]. Predictive coding can be implemented in particular types of recurrent neural networks (RNNs) including the RNN with parametric biases (RNNPB) [4] and multiple timescale RNN (MTRNN) [5] models. By optimizing synaptic weights for minimizing error, these types of RNN learn to predict future perceptual input sequences based on current intention states, represented by either the parametric bias (PB) in the RNNPB or the initial state values of the context units in the MTRNN. After learning, they are able to predict perceptual sequences corresponding to given intention states, for example proprioception and visual input sequences anticipated in the successful exercise of an intention. In the other direction, they can infer optimal intention states for given target perceptual sequences. These capacities, including the abilities to reconstruct action sequences and so to look for intentions behind perceptual outcomes, have interesting implications for social robotics and philosophy of mind. Ito and Tani [6] showed that a RNNPB driven humanoid robot can learn to imitate multiple movement patterns as demonstrated by human experimenters. The robot was able to imitate and to synchronize its own actions with demonstrated patterns by inferring corresponding intentional states as represented by PB units. And, a RNNPB inspired by the deterministic predictive coding principle [1] has also been shown to account for mirror neural functions [7] pairing generation and recognition of movement patterns with their intention and reconstruction.

That aside, these RNN-based deterministic predictive coding models have demonstrated difficulties in dealing with fluctuated sequential patterns. Especially, time-warping[8] (temporal expansion and contraction) causes severe problems in learning and recognizing sequence patterns because it tends to generate large amounts of accumulated error as processing of sequences proceeds. Although conventional schemes such as Dynamic Programming and

Hidden Markov Models (HMM) can deal with this problem efficiently, they are incapable of fully autonomous learning because they require predefined labels and graph structures. Murata and colleagues [9, 10] have attempted to overcome this limitation with the stochastic RNN (S-RNN), inspired by the Bayesian predictive coding scheme proposed by Friston [2, 11, 12]. The essential characteristic of this model is that the output predicts the estimated mean and variance of the target value statistically instead of predicting its exact value deterministically. This characteristic allows s-RNN to deal with observational noise in the outputs. However, it cannot deal with noise in the internal states, which is why this model is limited in coping with the time-warping problem. The current paper investigates the possibility that a dynamic predictive coding scheme implemented in a MTRNN model can deal with the problem effectively, and how it develops this potential through learning.

The MTRNN model is composed of multiple levels, each containing internal neural units (context units). The context units in the lower sub-network are constrained by fast, and those in the higher level by slow time constants. Current intention states for generating future sequence patterns are represented by the current dynamic states of the context units in all sub-networks. Our main conjecture is that a MTRNN model based on deterministic dynamics can learn to extract dimensions of modulation latent in observed temporal patterns with generalization, and so manage time warping type problems. We test this conjecture with varying internal state trajectories, especially in the slower dynamics sub-network. These variant trajectories account for possible modulations in particular dimensions (such as speed or amplitude in target patterns) by specifying dynamic structures sensitive to given inputs. And, all learned fluctuated temporal patterns can be reconstructed by inferring the set of the initial state values for the corresponding sequence of internal states. So, if dynamic structures accounting for fluctuations are adequately developed via learning, test patterns belonging to the same dimensions of fluctuation should be recognized and regenerated in the same ways. At the same time, variations in the higher level of the deterministic MTRNN should be able to account not only for switching among a set of learned prototypical patterns, but also for their fluctuation in particular dimensions. The present work investigates both whether and how this happens through a series of simulations and humanoid robotics experiments.

In the MTRNN, memory of a set of prototypical patterns develops in terms of invariant sets of corresponding local attractors, with fluctuations in

these prototypical patterns evident in the vector flow in the transient region surrounding the invariant sets. First, the following section sets out simulation experiments using low dimensional simple patterns in order to examine how particular dimensions of fluctuation can be extracted from training exemplars and how the acquired internal structure can be utilized effectively in a test of synchronized imitation of given target patterns characterized by the same dimension of fluctuation. After that, the paper details a second simulation experiment using naturally fluctuating human movement patterns in order to investigate structures of attractor dynamics developed through the course of learning multiple categories of movement patterns with certain degrees of fluctuation. Analysis shows that the developed attractor structure contributes to increasing robustness in tests of synchronized imitation performed by using error regression. Finally, the model was tested on a task of imitative interaction between a humanoid robot and human subjects in order to examine how well it copes with fluctuations in perceived temporal patterns. Results here show that the model affords spontaneous and lively interactions with human partners.

2 MTRNN model

2.1 Overview

A MTRNN is a type of RNN that consists of input units, output units, and multiple levels of sub-networks containing context units operating on the leaky integrator neuron model [13] with specified time constants. Leaky integrator units within each sub-network employ time constants specific and unique to that sub-network, i.e. neural activation dynamics in the higher level are slower with a larger time constant, and the lower level is faster with smaller time constant. Input units receive current percepts, and prediction of the next steps perceptual state is generated in the output units. This prediction is significantly affected by the current internal state of the MTRNN, represented by the activation state of all context units in all levels in the model network. Basically, the prediction is made based on the current intention, which is dynamically represented by the internal states of the model network.

Prediction outputs can be generated in three different ways. The first called sensory entrainment or open loop generation is a scheme whereby the

feeding of perceptual inputs to the network model entrains its internal neural dynamics, enabling predictive outputs. The second way is called closed-loop generation in which the output prediction sequence is generated by copying the current step perceptual prediction in the output units into the next step perceptual inputs. This operation can be used to generate mental simulation [14, 15] of possible perceptual sequences based on the current intention without using the real perceptual inputs from the environment. Schematics of open-loop and closed-loop operations are shown in Figures 1(A) and 1(B). Parameter l in Figure 1 represents the number of predicted steps, i.e. given current input, the MTRNN generates l look-ahead prediction steps as output. Parameter l was set as 5 in our simulation experiments and 7 in the robot experiments.

The third way is called closed-loop output generation while inferring the internal states by error regression. This is the main scheme employed in the current paper, enabling on-line action generation via predictive coding. Action is generated along with the top-down prediction of the perceptual sequence in the closed-loop way based on the current internal state representing the intention of the model network. In the other direction, prediction error is generated from the perceived action outcome and the internal state is modified in the direction of minimizing prediction error via error regression in the bottom-up manner. Further actions are generated based on the modified intention while the top-down and the bottom-up processes iterate in an on-line manner. A schematic of closed-loop output generation while inferring the internal state by error regression can be seen in Figure 1(C).

The learning process of a MTRNN involves the optimization of the initial context states corresponding to all exemplar sequences for the purpose of minimizing prediction/reconstruction error. Variations in initial context states account for different classes of patterns to be learned and for fluctuations within each class.

2.2 Generation and training methods

Neural unit activation dynamics are those of the leaky integrator neuron as shown in Equation 1.

$$u_i^{t+1} = \left(1 - \frac{1}{\tau_i}\right) u_i^t + \frac{1}{\tau_i} \left(\sum_j w_{ij} c_j^t + \sum_k w_{ik} x_k^t + b_i \right) \tag{1}$$

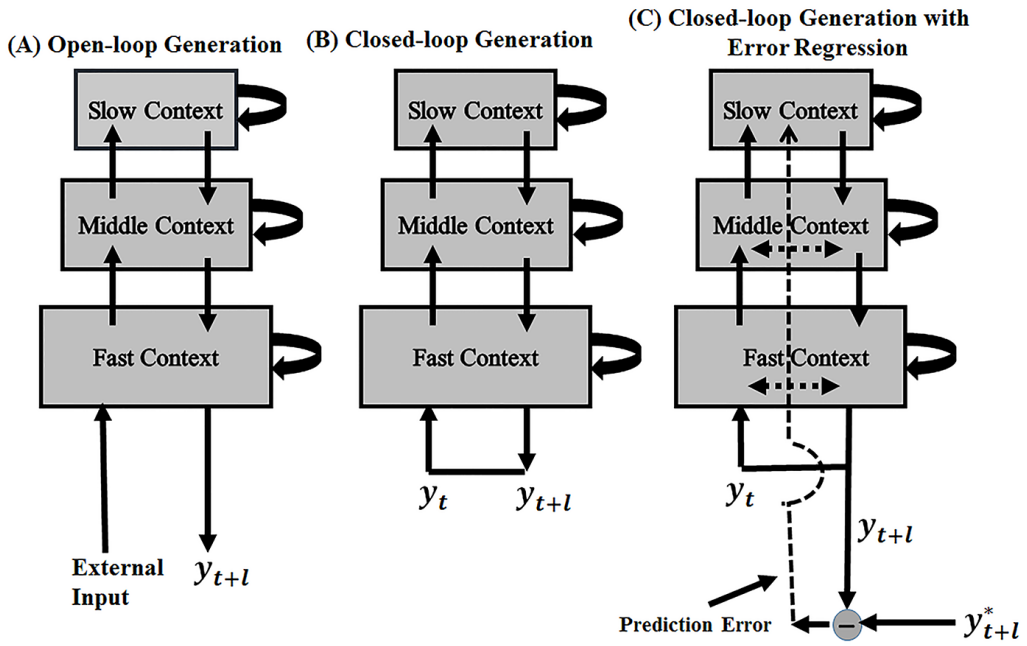

Figure 1: Schemes for (A) open-loop output generation (B) closed-loop output generation (C) closed-loop output generation while inferring the internal states by the error regression. Parameter l represents the number of predicted steps. In (A), by giving current external input, MTRNN generates l look-ahead prediction steps as outputs. In (B) and (C), by giving the current output prediction (y_t), MTRNN generates l look-ahead prediction steps as outputs (y_{t+l}).

where u_i^t is the internal state value of the i_{th} neural unit at time t, w_{ij} is the connectivity weight from the i_{th} context unit to the j_{th} context unit, w_{ik} is the connectivity weight from the i_{th} neural unit to the k_{th} input unit, c_j^t is the context unit activation value of the j_{th} neural unit at time t, x_k^t is the external input of the k_{th} input unit at time t, b_i is the bias of the i_{th} neural unit, and τ_i is the time constant of the i_{th} neural unit. If the neural unit does not belong to the lowest level sub-network, the second summation term does not exist, since there are no connections to the input units. The context units inside the lowest level sub-network are referred to as Fast Context (FC) units while the ones inside the highest level sub-network are referred to as Slow Context (SC) units. If there are sub-networks between the highest and lowest sub-networks, context units inside them are referred as Middle Context (MC) units. Similar to [5], input and output units are only connected to FC neurons. All output units have a time constant of 1 in this paper.

The context unit output values are found with the following activation function as recommended by [16, 17] for faster convergence.

$$c_i^t = (1.7159)\tanh(\frac{2}{3}u_i^t) \tag{2}$$

The connection weights between context units (w_{ij}) are bidirectional ($w_{ij} \neq w_{ji}$). If a MTRNN consists of only two sub-networks, all neural units are connected to each other. If a MTRNN consists of three sub-networks, no connections between FC and SC units exist (see Figure 1). A softmax transformation is used to remap each input x_i^t into a higher dimensional space x_{ij}^t according to receptive fields of adjacent intervals of equal length, as follows [18]:

$$x_{ij}^t = \frac{\exp\left(\frac{-||k_{ij}-x_i^t||^2}{\sigma}\right)}{\sum_{j\in Z}\exp\left(\frac{-||k_{ij}-x_i^t||^2}{\sigma}\right)} \tag{3}$$

where k_{ij} represents the j_{th} dimension of the reference vector for the i_{th} dimension of the real input value before transformation at time t, x_i^t is the real input value at time t, σ is a constant value that specifies the shape of the distribution (set as 0.05 in our all experiments), Z is the dimension of the reference vector and x_{ij}^t is the transformed vector. Z is set 11 in all experiments and it is found heuristically. Sparse encoding of input data is achieved by using the softmax transformation and it can reduce the overlaps of input sequences[5]. In this paper, variables including inputs and outputs

before mapping by the softmax transformation or after mapping by inverse softmax transformation are called the real variables and the variables that are mapped by the softmax transformation are called the softmax variables (softmax inputs or outputs). The reference vectors are computed by the equation below:

$$k_{ij} = B_i^{Min} + \frac{B_i^{Max} - B_i^{Min}}{Z - 1}(j - 1) \tag{4}$$

where B_i^{Min}, B_i^{Max}, and Z represent the minimum value for the i_{th} dimension of the real input data, the maximum value for i_{th} dimension of the real input data, and the dimension of the reference vector, respectively.

The activations of the output units are computed as follows:

$$u_{ij}^t = \sum_l w_{ijl} c_l^t + b_{ij} \tag{5}$$

$$y_{ij}^t = \frac{\exp\left(u_{ij}^t\right)}{\sum_{k \in Z} \exp(u_{ik}^t)} \tag{6}$$

where u_{ij}^t is the internal state of the j_{th} softmax output unit corresponding to the i_{th} real output unit, w_{ijl} is the connectivity weight from the l_{th} neuron in the FC units to the j_{th} softmax output unit corresponding to the i_{th} real output unit, c_l^t is the context output of the l_{th} neuron in the FC units, b_{ij} is the bias of the j_{th} softmax output unit corresponding to the i_{th} real output unit, and y_{ij}^t is the j_{th} softmax output unit corresponding to the i_{th} real output unit at time t. The inverse of the softmax transformation described in Eq. (7) calculates the i_{th} dimension of the real output units (y_i^t) at time t. In other words, Eq. (7) maps the softmax outputs to their original dimensions (real outputs).

$$y_i^t = \sum_{j \in Z} y_{ij}^t k_{ij} \tag{7}$$

A conventional back-propagation through time (BPTT) scheme is used for the network training [19, 20]. The learnable parameters are optimized in the direction of minimization of the Kullback-Leibler divergence (noted as E) between desired (target) and real activation values of softmax output units

($\bar{y}_i^t$ and y_i^t, respectively), according to Equation 8 [5]:

$$E = \sum_{t=1}^{T} \sum_{i \in O} y_i^t \log(\frac{\bar{y}_i^t}{y_i^t}) \tag{8}$$

Where T is the length of a sequence. All learnable parameters, shown as θ, are weights and biases and initial states that approach their optimal values in the opposite direction of the gradient $\frac{\partial E}{\partial \theta}$, and they are updated as follows:

$$\nabla\theta(n+1) = \mu\nabla\theta(n) - \alpha\frac{\partial E}{\partial \theta} \tag{9}$$

$$\theta(n+1) = \theta(n) + \nabla\theta(n+1) \tag{10}$$

where α is the learning rate, set as 0.00003 in all of our experiments, and μ is the momentum term which is set as 0.9 in all experiments. These values were found heuristically. The weight and bias gradients for all training sequences (s) can be obtained as follows:

$$\frac{\partial E}{\partial w_{ij}} = \begin{cases} \frac{1}{\tau_i} \sum_s \sum_t \frac{\partial E}{\partial u_i^t} x_j^t, & (i \in C \ \wedge \ j \in I) \\ \frac{1}{\tau_i} \sum_s \sum_t \frac{\partial E}{\partial u_i^t} c_j^t, & (i \in C \ or \ O \ \wedge \ j \in C) \end{cases} \tag{11}$$

$$\frac{\partial E}{\partial b_i} = \frac{1}{\tau_i} \sum_s \sum_t \frac{\partial E}{\partial u_i^t} \tag{12}$$

where C is context units, I is input units, and O is the output units. $\frac{\partial E}{\partial u_i^t}$ can be computed using Equation 13:

$$\frac{\partial E}{\partial u_i^t} = \begin{cases} y_i^t - \bar{y}_i^t, & (t \geq 1 \ \wedge \ i \in O) \\ \sum_{j \in C} \frac{\partial E}{\partial u_j^{t+1}} [\delta_{ij}(1 - \frac{1}{\tau_i}) + \frac{1}{\tau_i} w_{ji} c'(u_i^t)] + & \\ \sum_{k \in O} \frac{\partial E}{\partial u_k^t} [w_{ki} c'(u_i^t)], & (t \geq 0 \ \wedge \ i \in C) \end{cases} \tag{13}$$

where $c'(u_i^t)$ is the derivative of c_i^t at time t, and δ is Kronecker delta function.

As learning begins, synaptic weights (w_{ij}) are set randomly from a uniform distribution on the interval $[\frac{-1}{N_I}, \frac{1}{N_I}]$ (if $j \in I$) and $[\frac{-1}{N_C}, \frac{1}{N_C}]$ (otherwise), where N_I and N_C are the number of input and context units, respectively, and biases and initial states are set to 0. The open-loop approach was used in the learning phase in all experiments, i.e. the MTRNN receives current

inputs and generates one or multiple look-ahead prediction steps as outputs. Training ran for 50000 epochs in all experiments, and average mean square error (MSE) of closed-loop generation was computed in each training epoch. The learnable parameters obtained from the training epoch with minimum average MSE of closed-loop generation were used in test phases. It should be noted that the closed-loop generation did not affect updating of the learnable parameters.

2.3 Inferring internal states by error regression during synchronized imitation

After the learning process, the MTRNN model was tested on a synchronized imitation task. Given target temporal patterns were structurally the same as learned ones, but with modulations in speed and amplitude. During synchronized imitation, the model has to imitate the target pattern without delay by predicting next step input by inferring the prototypical patterns to be generated as well as their modulations. The MTRNN does this by using error regression to infer the intention. Closed-loop prediction of target patterns is accomplished by backpropagating prediction error through time (within a temporal window (W) of the immediate past) from the output through the internal states of the MTRNN in a bottom-up manner, thereby modifying internal states in order to minimize the current prediction error. In greater detail, by means of the BPTT scheme the prediction errors from the t-W time step to the current t time step can be used to update the internal states of the t-W time step, which result in changes to all context units and prediction outputs inside the temporal window. It should be noted that the connectivity weights and biases obtained during the learning phase are fixed in the error regression during synchronized imitation testing.

Equations used in updating the internal states (activation states of all context units in the whole network) at the onset of a temporal window are same as equations 9 and 10. However, we use a different term for the learning rate (α) in the error regression scheme. This is called the error regression adaptation rate (α_{ER}), and was set as 0.001 in all of the present experiments except for the last robotic experiment. Also, we do not use the momentum here (we set the value to zero). In all present experiments, the temporal window length was set to 15 and error regression was performed for 100 regression steps. All former internal states were overwritten from the *t-15*

time step to the current t time step at each time step within the temporal window. By inferring the internal states inside the temporal window, the prediction of internal states and output values after this window (future plans) can be also modified (see Figure 10 for details).

3 Simulation experiments

Two simulation experiments investigated the basic mechanism of the model network in coping with temporally fluctuating patterns. The first experiment examined how the proposed model learns to extract essential dimensions of fluctuations latent in exemplar patterns, and how the error regression can utilize the learned structures in successfully performing synchronized imitation. The second experiment examined what sorts of dynamic structures develop in the model during learning naturally fluctuated low dimensional patterns generated by human subjects using a drawing tablet. Analysis suggests that error regression with fluctuated patterns employs both the transient and steady state regions. Note that these simulation experiments employed relatively simple patterns without compositionality or hierarchy. Our focus was testing the expectation that the slow dynamics of the higher level of the MTRNN would play a crucial role in coping with fluctuations in exemplar patterns as well as in the test patterns, so more complex patterns were not necessary.

3.1 A sine-curve pattern with two dimensions of fluctuation

This section reviews a simple experiment using a one-dimensional sine curve as shown in the following equation

$$y = Asin(B\frac{\pi}{p}t), \quad (t \geq 0) \tag{14}$$

where A, B are variables that modulate the amplitude and period of the sine curves, and p is set to 30. Training sequences were made in two forms. In the first case, periodicity fluctuation, 5 sine-curve patterns were generated in which each signal had 5 cycles. Only the periodicity of sine-curve patterns was fluctuated (time warped signals). Values of A were set to 1 for all sequences but B values were chosen randomly each cycle from a normal

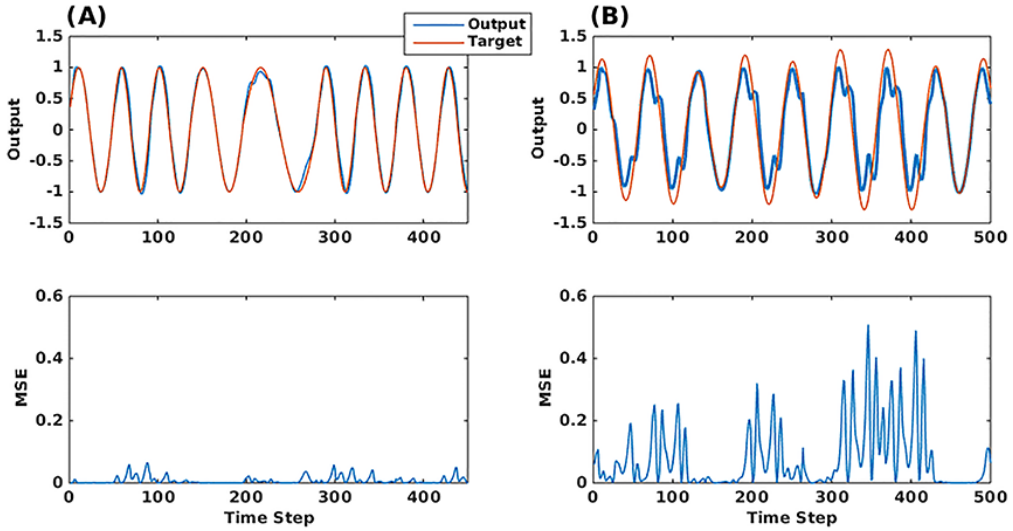

Figure 2: Closed-loop output generations with the error regression and MSE of MTRNN-P that was trained using time warped teaching sequences when (A) A test pattern with only fluctuations in its periods (time warped) was used in the test phase and (B) A test pattern with only fluctuations in its amplitudes was used in the test phase.

distribution with a mean value (μ) of 1 and standard deviations (σ) of 0.2. This means that 25 B values were chosen randomly (each cycle had a different value). In the second case known as amplitude fluctuation, we generated 5 different sine-curve patterns by setting B as 1 and choosing A values randomly, meaning that only the amplitude of each cycle was fluctuated. One MTRNN referred to as MTRNN-P was trained in order to reconstruct the 5 training patterns with fluctuation in their periodicity whereas MTRNN-A was trained in order to reconstruct the 5 training patterns with amplitude fluctuation. Both MTRNNs consisted of 20 FC, 10 SC, 11 softmax input and 11 softmax output units. The time constants of FC and SC units were set to 2 and 50, respectively. Training ran for 50000 epochs in each case.

After training, both MTRNN-P and MTRNN-A successfully regenerated all training patterns in a closed-loop manner by using corresponding initial states, with average mean square error (MSE) of 0.0456 for MTRNN-P and 0.0108 for MTRNN-A. In the test phase, we generated two test patterns using Equation 14 in the same way that we generated training patterns, but

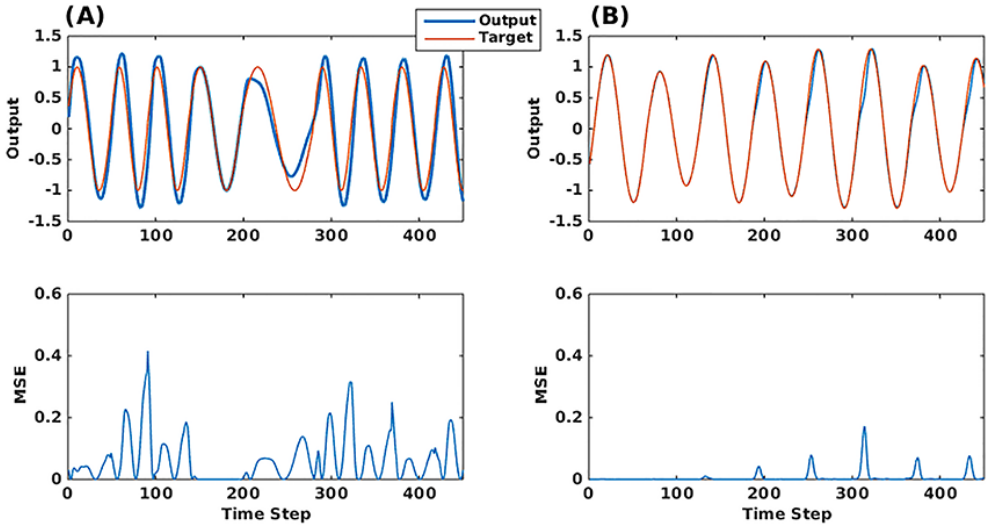

Figure 3: Closed-loop output generations with the error regressions and MSE of MTRNN-A trained using teaching patterns with only fluctuations in amplitude when: (A) A test pattern with only fluctuations in its periods (time warped) was used in the test phase, and (B) A test pattern with only fluctuations in its amplitudes was used in the test phase.

with the standard deviations (σ) of these two test patterns set at 0.3. The first test pattern had fluctuations only in its periods while the other had fluctuations only in its amplitudes. The error regression results of MTRNN-P and A are given for the two test patterns in Figures 2 and 3. The first and second rows in both figures show prediction outputs and Mean Square Errors (MSEs), respectively.

Looking at these figures, we can see that both networks show good performance when they are tested with same dimension of fluctuations as that with which they were trained, while they both generate large errors when they are tested with dimensions of fluctuation different from their training patterns. In other words, the MTRNN can perform synchronized imitation by minimizing error only with target patterns sharing the same fluctuation structures as learned ones. And thus, error regression enables the network to infer internal states corresponding to currently perceived fluctuations in patterns by way of these prior learned structures.

3.2 Naturally fluctuated pattern

The next simulation experiment involves naturally fluctuating patterns instead of the computer-generated patterns of the previous experiment. Here, model performance in learning and in synchronized imitation by error regression is demonstrated using low-dimensional temporal patterns, one made by a computer without perturbation, and the other made by a human attempting to mimic the computer generated patterns using a tablet input device. The first type of training pattern was generated using equations for a circle and for Lissajous curves as follows:

$$\begin{cases} y_1 = rcos(t) + a_1, & (0 \leq t \leq 2\pi) \\ y_2 = rsin(t) + b_1, & (0 \leq t \leq 2\pi) \end{cases} \tag{15}$$

$$\begin{cases} y_1 = A_1 sin(a_2 t + \delta_1), & (0 \leq t \leq 2\pi) \\ y_2 = B_1 sin(b_2 t), & (0 \leq t \leq 2\pi) \end{cases} \tag{16}$$

where, in Equation 15, r, a_1, and b_1 were set as 0.4, 0, and -0.2, and in Equation 16, A_1, B_1, a_2, b_2 and δ_1 were set as 0.5, 0.5, 1, 2, and $\frac{\pi}{2}$, respectively. Each training pattern consisted of 5 full cycles (t goes from 0 to 2π in one full cycle) of circle and Lissajous figures. It should be noted that these two training patterns exhibited no fluctuation. They were perfectly regular.

The second type of training pattern was generated by humans attempting to draw the same circle and Lissajous-curve patterns on a drawing tablet. As in the first type, patterns were generated for 5 full cycles for each figure. Two dimensions of fluctuation were naturally generated, both time-warping and amplitude shifting, as illustrated in tablet generated training pattern 1 and 2 in Figure 4(A) and Figure 4(B), respectively. The first and second rows show the first dimension (y_1) and second dimension (y_2) of training patterns over time. Their phase plots appear in the third row. The 5 Lissajous-curve patterns shown in Figure 4(A) differ in periodicity and amplitude. The same is evident in the circle-curve patterns in Figure 4(B). See Figure in B for computer generated patterns.

One MTRNN referred to as MTRNN-C (using computer generated patterns) was trained to reconstruct the 2 computer-generated training patterns (as in the first experiment) and another MTRNN referred to as MTRNN-H (using human generated patterns) was trained to reconstruct 10 naturally generated training patterns. Both MTRNNs consisted of 30 FC, 15 SC, 22

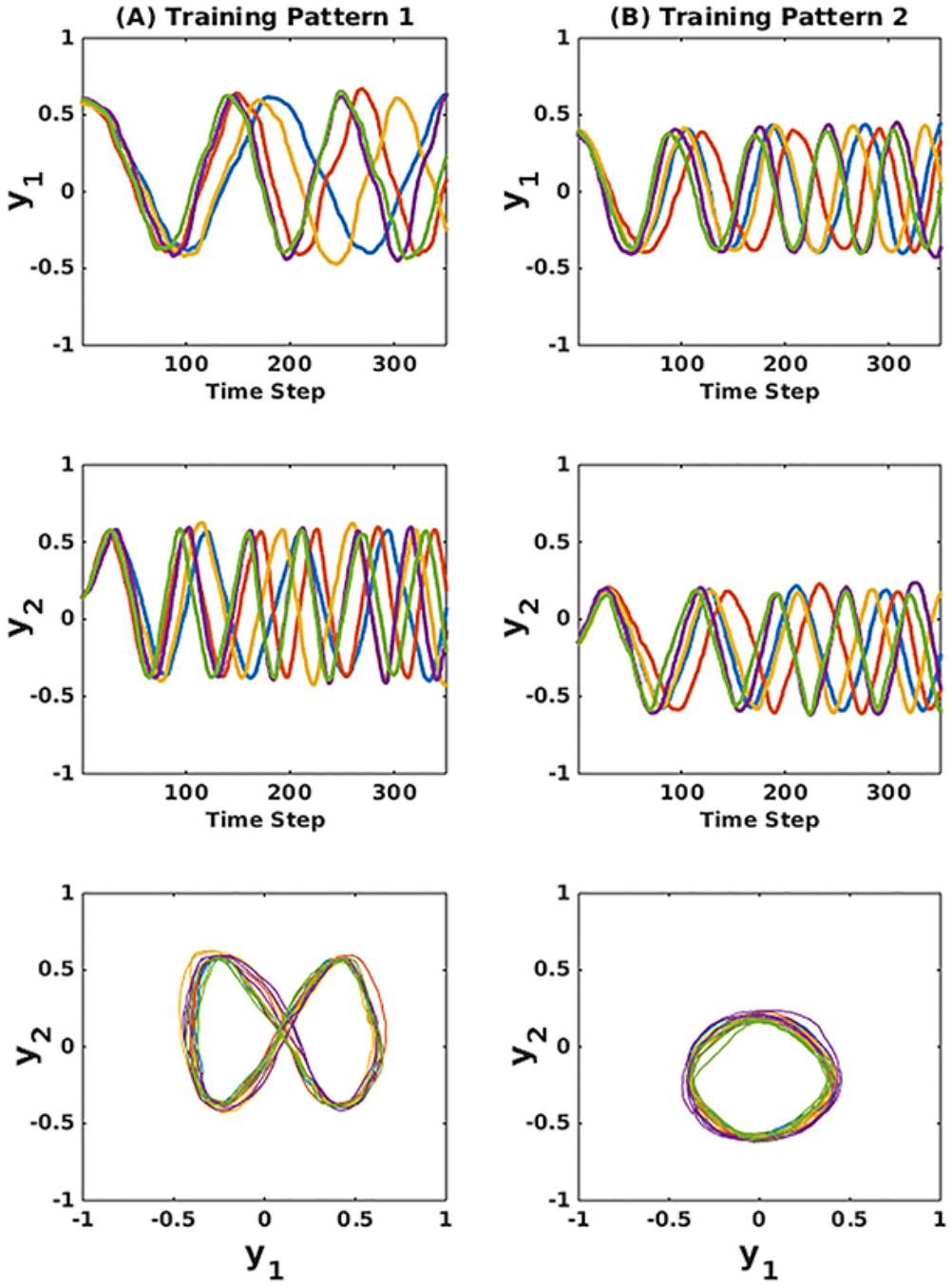

Figure 4: Training patterns generated by a human using a drawing tablet when (A) 5 Lissajous-curve patterns were produced and (B) 5 circle-curve patterns were produced. The first and second rows illustrate the first (y_1) and second (y_2) dimensions of the patterns, respectively. The last row shows the phase plots of training patterns 1 and 2.

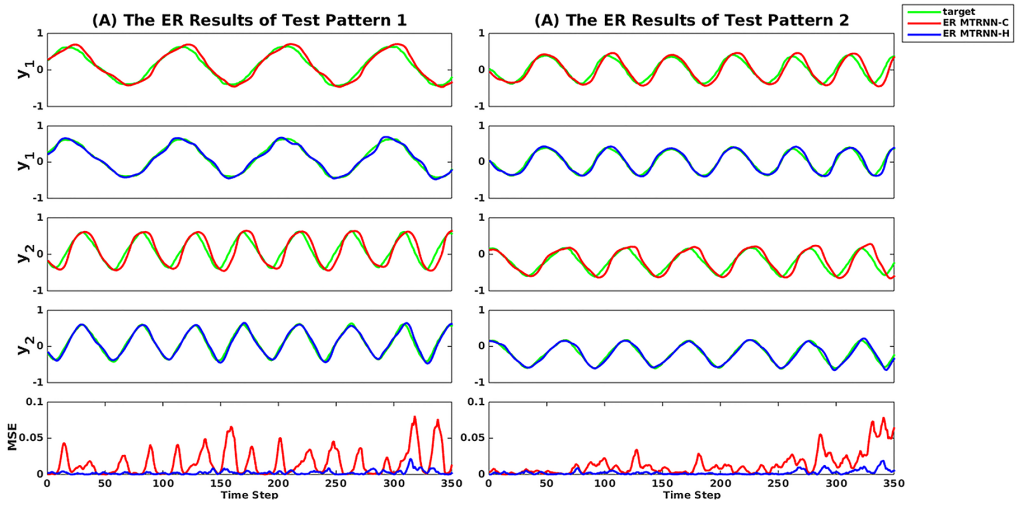

Figure 5: Comparison of closed-loop output generations with the error regression and MSEs for MTRNN-C that was trained using computer generated patterns with no fluctuations and for MTRNN-H that was trained using naturally fluctuated patterns given (A) Lissajous-curve pattern (test pattern 1) and (B) Circle-curve pattern (test pattern 2). ER MTRNN-C and ER MTRNN-H are abbreviations for the error regression outputs of MTRNN-C and the error regression outputs of MTRNN-H, respectively. The first and second rows show the first dimension of ER outputs and target patterns for MTRNN-C and MTRNN-H, respectively, while the third and fourth rows illustrate the second dimension of ER outputs and target patterns for MTRNN-C and MTRNN-H, respectively. The green, red, and blue lines correspond to test target, ER MTRNN-C and ER MTRNN-H, respectively.

softmax input and 22 softmax output units. The time constants of FC and SC units were set to 2 and 50, respectively. The training was successfully done for 50000 epochs for both cases.

In the test phase, the two test patterns, one Lissajous-curve pattern (test pattern 1) and one circle-curve pattern (test pattern 2), were generated by drawing tablet. Then, we looked at how the error regression exploits the learned structures in performing synchronized imitation with these two target patterns. Error regression results are depicted in Figure 5. The first and second rows show the first (y_1) dimension of the error regression outputs and target patterns for MTRNN-C and MTRNN-H, respectively, and the third and fourth rows illustrate the second (y_2) dimensions of the error regression outputs and target patterns for MTRNN-C and MTRNN-H. The red and blue lines in the last row represent MSEs for MTRNN-C and MTRNN-H, respectively. This figure shows that MTRNN-H outperforms MTRNN-C. It can be also seen that the proposed model can deal with time-warping and amplitude shifting which were naturally generated inside the test patterns.

As explained earlier, for each training pattern, the MTRNN learned initial context states during the leaning phase. By using the initial states corresponding to 2 training patterns of MTRNN-C, closed-loop output generations were computed for 50000 time steps. Likewise, by using initial states corresponding to 10 training patterns, MTRNN-H closed-loop output generations were computed for 50000 time steps. The phase plots for the first and last 4-cycle of the closed-loop output generations of MTRNN-C for the first pattern (Lissajous-curve pattern) are shown in Figure 6(A) and Figure 6(C), respectively. Similarly, the phase plots for the first and last 4-cycle of the closed-loop output generation of MTRNN-H for the first patterns (Lissajous-curve patterns) are depicted in Figure 6(B) and Figure 6(D), respectively. The first and last 4-cycle responses are referred to as transient and steady-state responses, respectively. It should be noted that in both Figure 6(A) and Figure 6(C), the transient and steady-state closed-loop output for only one pattern are shown because only one Lissajous-curve pattern was included in the training phase of MTRNN-C. Figure 6(B) and Figure 6(D) illustrate the transient and steady-state closed-loop output generated for 5 patterns because there were 5 Lissajous curve patterns in the training phase of MTRNN-H. It is also worth noting that the same results were obtained for circle-curve patterns that are not shown here.

In Figure 6, it can be seen that MTRNN-C learns a given target pattern as a global attractor, and the transition from the adapted initial states is

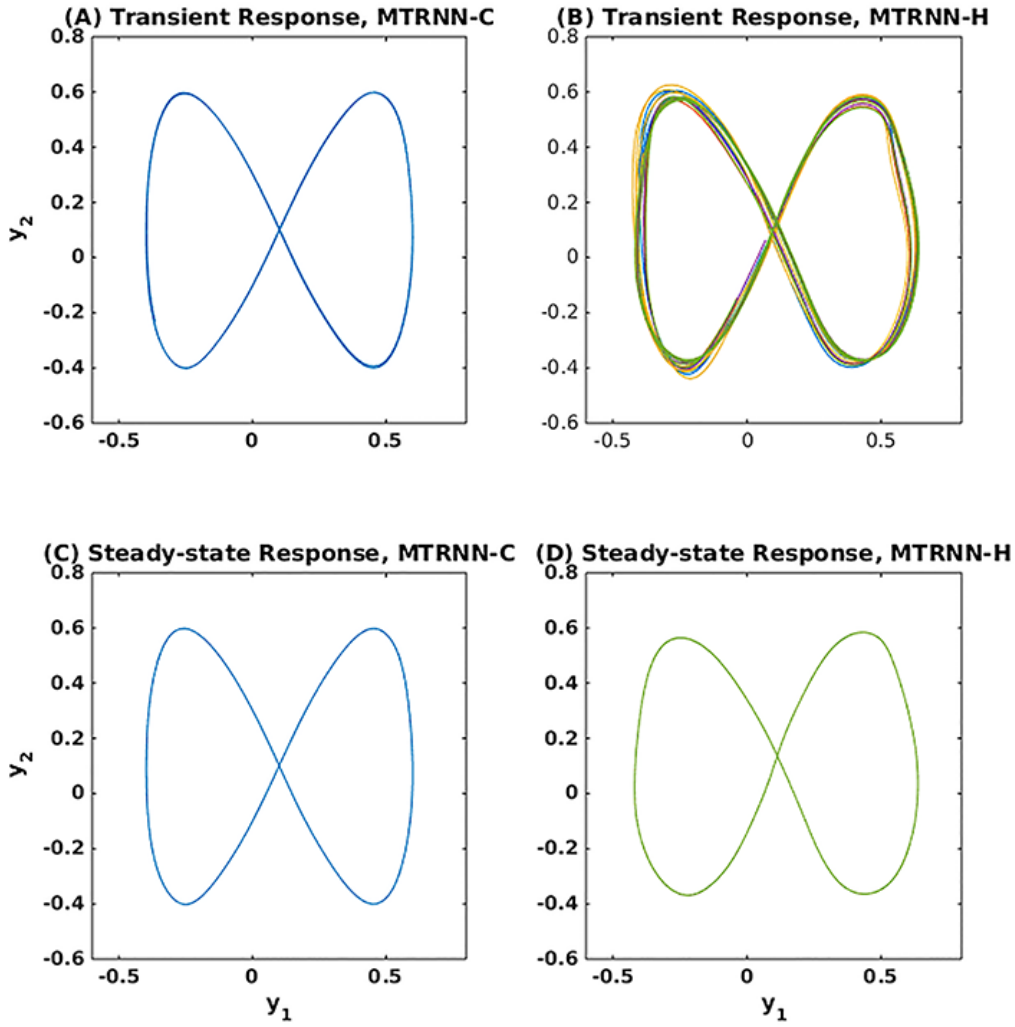

Figure 6: This figure illustrates the transient closed-loop output generations of (A) MTRNN-C and (B) MTRNN-H and also the steady-state closed-loop output generations of (C) MTRNN-C and (D) MTRNN-H for all Lissajous-curve training patterns. The closed-loop output generations were computed for 50000 time steps by using the initial states of all training patterns that were obtained in the learning phase. The first row (transient response) show the phase plot of the closed-loop output generation in the first 4 cycles, which are almost the same as corresponding training patterns shown in the last row of Figure 4. The second row shows the phase plot of the closed-loop output generations in the last 4 cycles (from time step 49500 to 50000) referred as the steady-state response.

quite short when compared to that of MTRNN-H. MTRNN-H encodes the trajectory of an averaged target pattern among multiple fluctuated patterns as a global attractor, and fluctuated patterns outside of the averaged pattern are reconstructed in the larger transient region.

Principle Component Analysis (PCA) was used to visualize the slow context activities of MTRNN-H in both training and test phases. PCA was applied on the corresponding slow context activities of the closed-loop output generations shown in Figures 6(B), and 6(D), and the first and second principle components are shown in Figure 7(A). The 5 patterns have different slow context activities in the transient region, but they all have the same activities in the steady-state region, consistent with results shown in Figures 6(B) and 6(D). PCA was also used to visualize the slow context activities of the error regression with test pattern 1 (corresponding to ER MTRNN-H in Figure 5(A)), and the first and second principle components are shown in Figure 7(B). The same steady-state figure shown in Figure 7(A) is repeated in Figure 7(B) to facilitate the visualization.

Figure 7(B) shows that the error regression moves between the steady state and transient regions, with most activity within the transient region. Comparing Figures 7(B), 6(B) and 6(D), it can be seen that fluctuated patterns are learned in the transient region, and that the error regression can infer both this transient region and steady-state regions through modulation of internal states.

4 Robot experiments

Imitation learning, which is called also learning by observation, has been widely studied in different fields of research. In robotics, imitation learning has been approached in terms of symbolic reasoning [21, 22, 23] and non-symbolic learning tools such as fuzzy logic [24], the Active Intermodal Matching (AIM) mechanism [25], Dynamical Recurrent Associative Memory Architecture (DRAMA) [26], and the MTRNN [27]. These models have allowed robots to follow and to learn multiple skills and actions from their tutors. For example, in an imitation game between a human and robot using a RNNPB model without hierarchy [6], the model was first trained to memorize multiple cyclic movement patterns of a human subject and was then used to control a robot in the regeneration of the memorized patterns. However, this study did not examine the problems introduced with pattern fluctua-

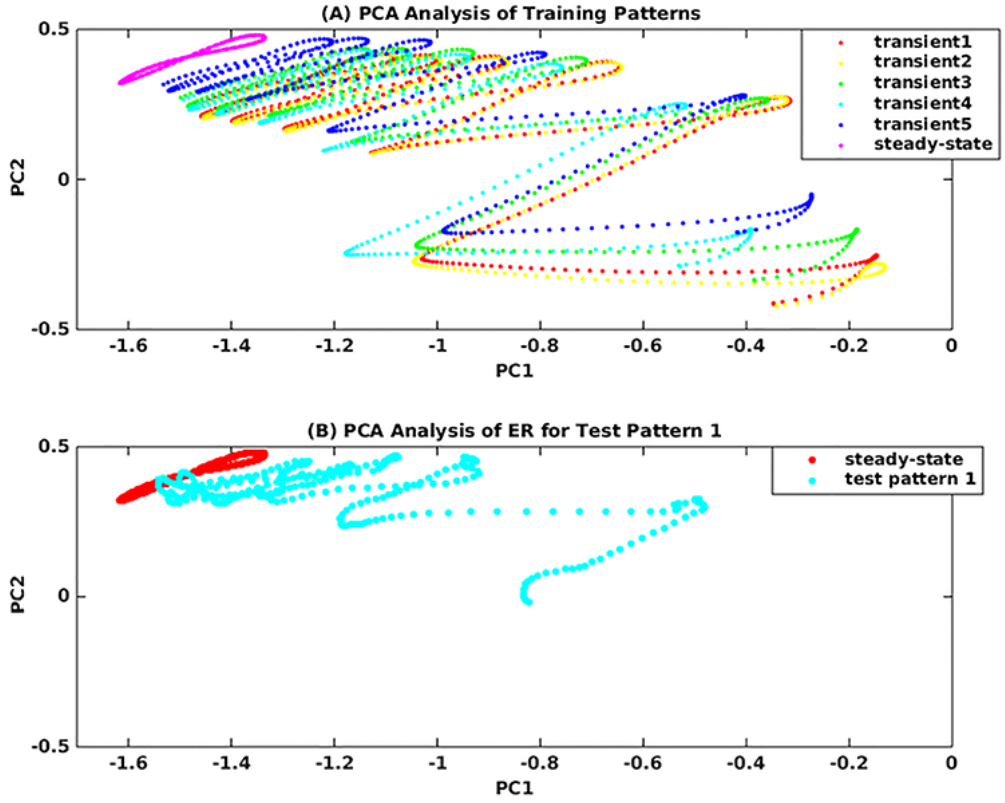

Figure 7: Slow context activities in a 2 dimensional space based on the results of PCA analysis. (A) Shows the slow context activities of the transient and steady-state closed-loop output generations of 5 training patterns corresponding to Figures 6(B), and 6(D). (B) shows the slow context activities of the error regression for test pattern 1 corresponding to ER MTRNN-H in Figure 5(A). Transient responses of slow context activities (the first 4 cycles) of the training patterns 1 to 5 are indicated by transient1 to 5 (dotted lines) in (A). The steady-state dotted line (pink dotted line) in (A) shows the steady-state responses of slow context activities (the last 4 cycles) of training patterns 1 to 5. Only one line appears because all 5 steady-state responses are the same and so perfectly overlap. In (B), test pattern 1 shows the slow context activities of the error regression with test pattern 1 and the same steady-state results shown in (A) is repeated here again in order to aid visualization.

tion. Many graphical probabilistic models such as hidden Markov models have been used to overcome the problem of fluctuation in temporal patterns such as the time-warping problem [28, 29, 30]. The present study attempts to show that the proposed RNN models can also deal with this problem by using its dynamical systems characteristics. In fact, RNNs work better for storing and accessing information over long periods of time than conventional sequential pattern learning algorithms such as hidden Markov models [31].

As explained in previous sections, the MTRNN is able to cope with fluctuations such as time warping and amplitude shifting during simple tasks. To test our model in more realistic situations, we conducted two robotic experiments involving imitative interaction between a humanoid robot and human subjects. These experiments examined how the proposed model can support spontaneous interaction between robots and human subjects while dealing with possible fluctuations in perceived temporal patterns. The first experiment follows human subjects in acquiring a set of cyclic patterns also learned by the robot during an imitation game. Synchronized imitation can be achieved on the robot side by using the on-line error regression of the internal states within the window of the immediate past. Natural interaction forces the robot to deal with more naturally fluctuated patterns. The second experiment examined how the balance between the top-down intentional process from the higher level and the bottom-up recognition processes from the lower level can affect spontaneous interaction between the robot and the human subjects by changing the error regression adaptation rate (α_{ER}). These robotic experiments employed a larger MTRNN than those used in the previous experiments consisting of three sub-networks, one with fast dynamics, one with intermediate dynamics and one with slow dynamics for the purpose of coping with more complex patterns (such as the hierarchically organized movement patterns used in the second robotic experiment).

4.1 Robot experiment design

We employed a NAO humanoid robot (developed by Aldebaran Robotics) and a Kinect sensor (developed by Microsoft) for imitative interaction tasks. The Kinect SDK and OpenNI framework were used to track the 3-D (X, Y, Z) coordinates of a humans arm joints. The 3-D positions of human-user arms were mapped to the 3-D positions of the NAO's arms with respect to the robot coordinate system. Next, the 3-D positions of the NAO were mapped to its joint angles (shoulder roll, and pitch and elbow roll, and yaw)

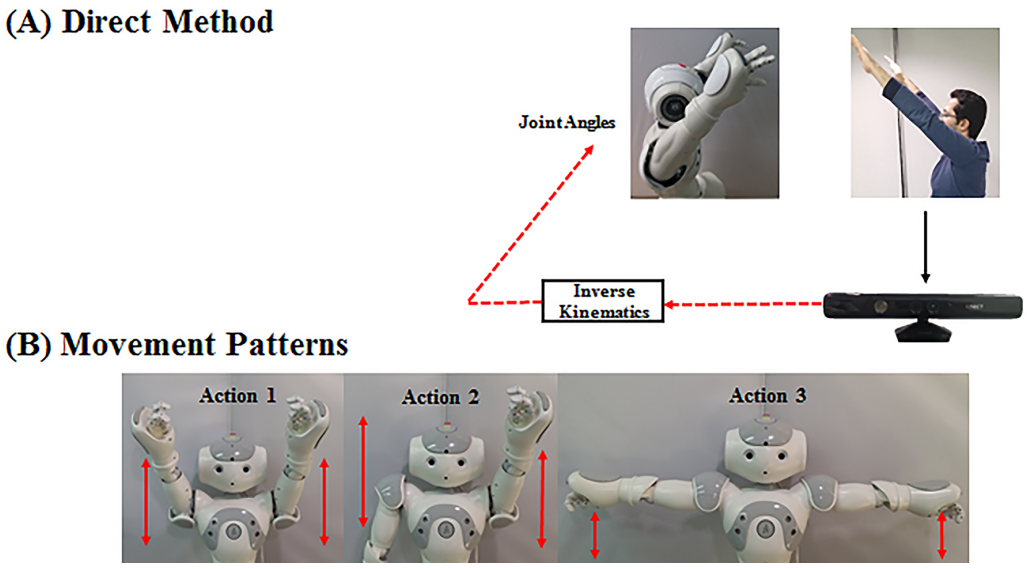

Figure 8: (A) Schematics of the direct method and (B) three cyclic movement patterns. MP is the abbreviation for movement pattern. The direct method is used during training and the MTRNN is not used for controlling the NAO. With the direct method, three movement patterns shown in (B) were generated.

by applying inverse kinematics.

4.2 Imitative interaction game

We designed an imitative interaction game between a robot and human subjects in order to investigate how well the model could cope with naturally fluctuated patterns generated by human subjects.

First, training data were collected by the experimenter who interacted with the NAO using the direct method. The direct method refers to the situation where a MTRNN is not used to control the NAO as shown in Figure 8(A). The three cyclic movement patterns shown in Figure 8(B) were generated using only the shoulder roll and pitch of both arms. Other joint angles remained fixed. Similar to Section 3.2, 5 sequences of training data were collected for each cyclic movement pattern. The final 15 training patterns had average time lengths of 232 steps with each time step length lasting

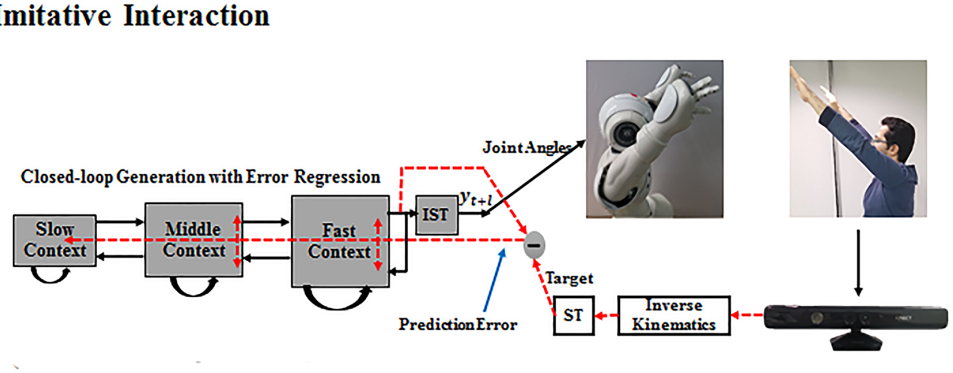

Figure 9: Schematic of the imitative interaction game. ST and IST are the abbreviation for softmax transformation and inverse softmax transformation, respectively. The 4 joint angles that are obtained after the inverse kinematic are mapped by softmax transform to be used as targets in the error regression process. The softmax outputs are then mapped to real outputs (joint angles) through inverse softmax transformation.

75 ms. There was around 520 ms delay between actual human movement patterns and perceiving them by the Kinect sensor. To overcome this delay, the prediction step l was set to 7 in all robotic experiments (7*75=525 ms).

A MTRNN consisting of 30 FC, 20 MC, 10 SC, 44 softmax input and 44 softmax output units was trained to reconstruct these 15 cyclic movement patterns. The time constants of FC, MC, and SC units were set to 5, 25, and 150, respectively. The training was successfully performed for 50000 epochs. The trained MTRNN was used in an imitative interaction game using the error regression approach as shown in Figure 9. 10 university students participated in the experiment, in which they interacted with the NAO without any prior knowledge about the cyclic movement patterns and experiments. Their first task was to figure out all of the movement patterns memorized by the MTRNN while achieving synchronized imitation with the NAO. This was a challenging task because human subjects tried so many different patterns with different periodicities and amplitudes in order to figure out the memorized patterns. They were given 10 minutes for the first trial, but if they failed to figure out all 3 movement patterns, they could have another trial with duration of 5 minutes. If again the human subject failed, he/she was given another 5 minutes. It turned out that all participants were

able to interact with the NAO successfully. However, two of them could not figure out one of the movement patterns.

The second task for the human subjects was to synchronize with the robot. First, they had to repeat movement pattern 1 until they felt that they were well synchronized with the NAO, then they could switch to movement pattern 2 and do the same, and finally they could synchronize movement pattern 3. All subjects synchronized successfully with the NAO. Previous and current task results show that the network deals with time warping and amplitude shifting even in such a noisy natural environment.

Figure 10 displays the detailed dynamic of the error regression approach for one of the participants during the synchronization stage. In Figure 10(A), transitions from movement patterns 1 to 2 ($MP1 \rightarrow MP2$) and from 2 to 3 ($MP2 \rightarrow MP3$) are shown at the top of the sensory prediction (sensory output) panel. The temporal windows of the error regression at a particular moment of switching from movement pattern 2 to 3 are depicted by gray areas in all panels, with the current time (labeled as now in Figure 10(B)) indicated in the right line of the windows. Internal states cannot be modified before the temporal window, but they can be revised within it to minimize prediction error by means of BPTT. Predicted states after this temporal window (the plan) are obtained by continuing closed loop generation to the end for each time step.

The error regression procedure can be seen more clearly in Figure 10(B). Sensory inputs, their prediction and the internal states change from the time step 300 to 400. The states for the current time steps (now) of 347, 355, and 363 are so-called pre-modification, modification, and post-modification phases, respectively. In the pre-modification phase depicted in the left panels, the human subject starts to change the movement patterns from 2 to 3 and MSE increases. The plan is still the movement pattern 2 (MP2) at this point. During the modification phase shown in the center panels, neural internal states in both SC and FC units are revised by error regression in order to reduce prediction error, and change the intention state. The immediate past of states is changed inside the temporal window and the plan is changed to movement pattern 3 (MP3), and remains movement pattern 3 in the post-modification phase displayed on the right panels. The MSE decreases significantly in the post-modification phase and the sensory prediction follows the target. Pre-modification, modification, and post-modification phases demonstrate that both future predicted states and the record of past states differ from each other during the same time steps. This is because during each time

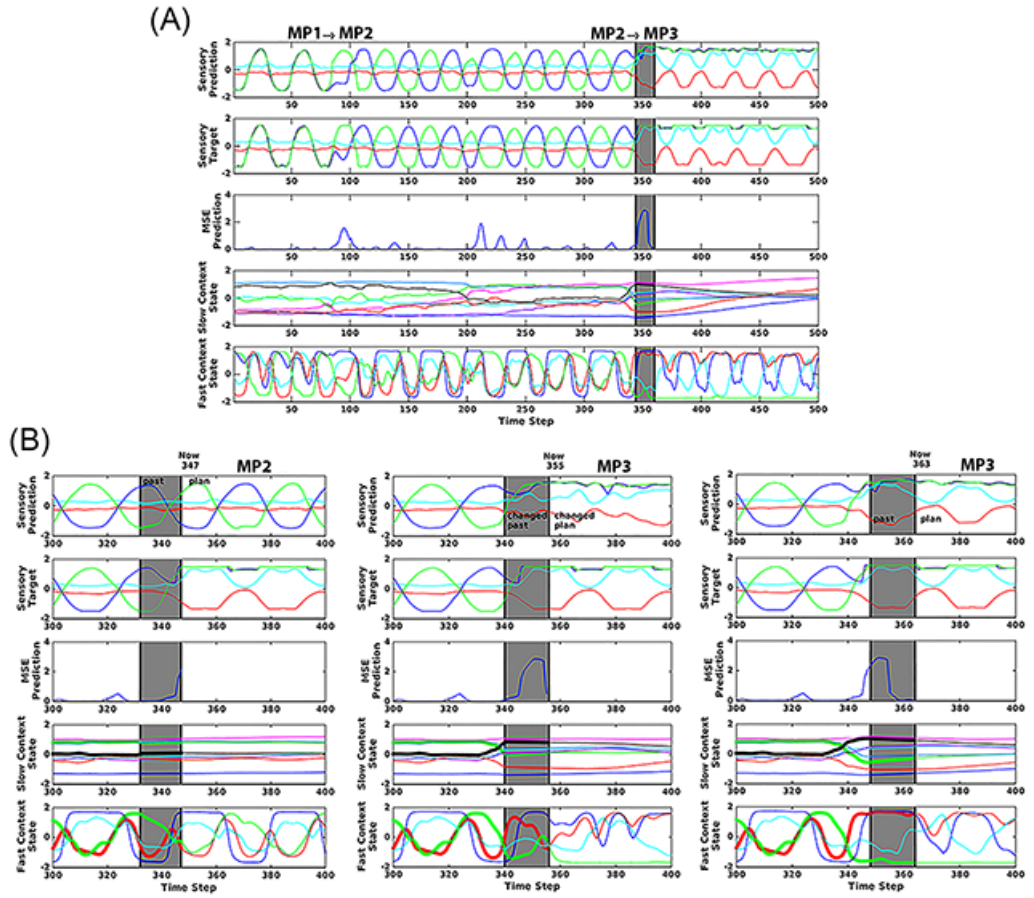

Figure 10: Detailed dynamic of the error regression scheme for one of the participants during the synchronization stage. Plot (A) illustrates error regression results during a large period of time (from 0 to 500). (B) shows error regression results during a shorter period of time (from 300 to 400) and the regression dynamic is shown for three different current (Now) time steps; 347, 355, and 363 referred to as the pre-modification, modification and post-modification phases, respectively. The gray areas are the temporal windows of the immediate past in which the internal neural states are modified to minimize the error. Changing states inside the temporal window can also affect future states, but states before the temporal window cannot be changed. As shown in left panels of (B), the plan is different from the sensory target, which is why a large MSE is generated in the pre-modification phase. The slow and fast context states are modified by error regression, and the plan is changed in the modification phase (MP3). As a result, MSE decreases significantly and sensory predictions follow the target in the post-modification phase. The blue, red, green, and cyan lines correspond to right shoulder pitch, right shoulder roll, left shoulder pitch, and left shoulder roll, respectively. A video corresponding to this figure can be seen in A

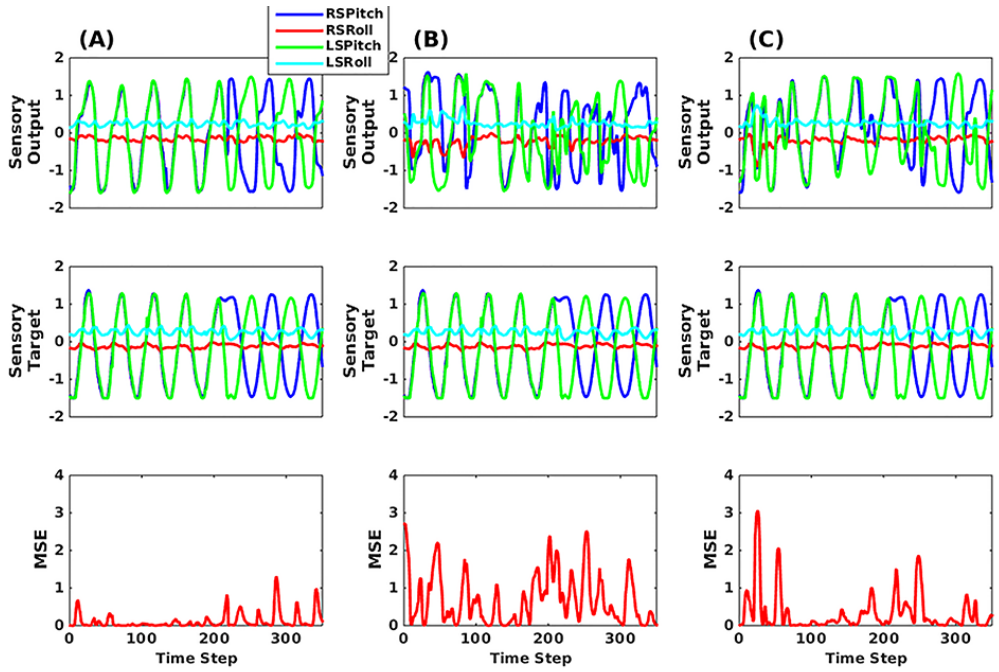

Figure 11: The sensory output, sensory target, and MSE results of (A) The error regression scheme, (B) The error regression scheme when only SC internal neural units are modified, and (C) Sensory entrainment scheme. The RSPitch, RSRoll, LSPitch, and LSRoll abbreviate right shoulder pitch, right shoulder roll, left shoulder pitch, and left shoulder roll, respectively. Joint angles are measured in radians.

step, immediate past states within the temporal window can be overwritten by means of error regression, thereby modulating internal neural states. Past states outside of the temporal window are constant and cannot be changed by error regression, however.

We also compared the performance of the error regression scheme with that of the conventional sensory entrainment scheme (open-loop output generation) using the same MTRNN. In other words, we did not use the error regression scheme, instead inputting directly to the sensory inputs of the MTRNN from the Kinect. The test experiment showed that it was difficult for the robot to synchronize with human movement patterns using the entrainment scheme. The sensory entrainment scheme is not powerful enough to immediately modify the internal states in order to minimize prediction

Table 1: MSE for three different schemes of generating prediction outputs

Different schemes		
ER	ER-H	SE
0.1329	0.6775	0.3234

error. Also, we tested a case in which error regression was applied to SC units but not to FC units, analogous to the on-line adaptation of PB units in previous work [6]. It was again found that synchronization between the human subjects and the robot was not easily achieved.

In order to show quantitative differences between these three schemes, we conducted an off-line synchronized imitation test using a test sensory pattern containing sequential switching between two trained prototypical patterns. The test pattern was collected by using the direct method. The simulation results are shown in Figure 11. The left panel shows the results of the proposed model. The middle and right panels show results of the closed-loop output generation with error regression applied only to the SC units but not the FC units, and the sensory entrainment scheme (open-loop output generation). In order to compare three schemes quantitatively, their MSEs over the whole time steps are shown in Table 1. ER, ER-H, SE are abbreviations for the error regression in all layer, error regression in the higher layer (SC units), and sensory entrainment schemes, respectively. In summary, the proposed model performs best in minimizing on-line prediction error.

4.3 Effect of balance between the top-down and the bottom-up processes

We investigated how the balance between the top-down intentional prediction and the bottom-up error regression for modifying the intentional state can affect natural interaction between the human subjects and the robot in terms of synchronized imitation by manipulating the error regression adaptation rate (α_{ER}). This experiment was conducted with a MTRNN which was trained with a concatenated sequence consisting of the prior-learned prototypical patterns. One sequence of the training data concatenated 3 cycles each of movement patterns 1, 2, and 3 via the direct method. This series was repeated 5 times. This one sequence of data was given to the MTRNN model (with the same parameter settings as the network in Section 4.2) during the

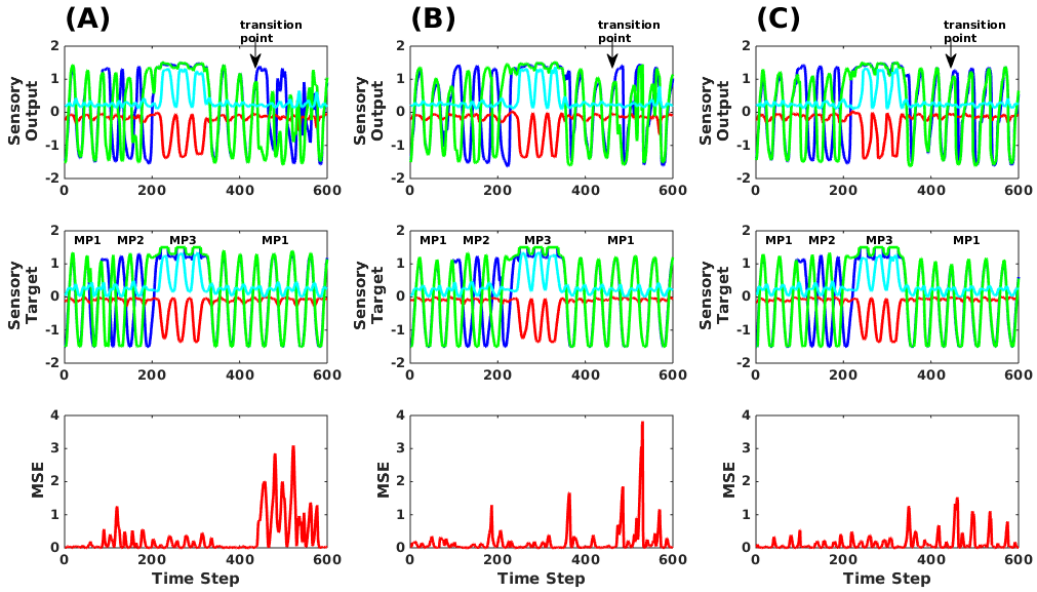

Figure 12: The sensory output generations with the error regression with three α_{ER} values of (A) 0.00001, (B) 0.0001, and (C) 0.001. Sensory targets and MSEs are shown in the second and third rows. In the sensory output panels, the transition points indicating instances that the human subject tried to repeat MP1 instead of shifting to MP2 are shown. The movement patterns (MPs) generated by the human subject are shown in the sensory target panels. The blue, red, green, and cyan lines correspond to right shoulder pitch, right shoulder roll, left shoulder pitch, and left shoulder roll, respectively

training phase and all learnable parameters were initialized with the values computed in Section 4.2. Only the weights and biases of SC units were allowed to change while other learnable parameters were fixed, according to the expectation that new learning involving the concatenation of prior-learned prototypical patterns is conducted mainly by the higher level sub-network.

After training, one human subject synchronized with the NAO by first generating movement patterns 1, 2, and 3 (in a series, 3 cycles for each movement pattern) repeated several times. Then, the subject changed the order of movement patterns by continuing to generate the same movement pattern 1 (MP1) without shifting to MP2 at the crucial time step at which switching was learned (indicated as the transition point in Figure 12). We examined how the robot reacts to the change of order using three different

Table 2: MSE for experiments with three different error regression adaptation rates

ER Adaptation Rate		
Small	Middle	Large
0.3107	0.1824	0.1445

error regression adaptation rates. Results for α_{ER} set with 0.00001, 0.0001, and 0.001 are depicted in Figure 12(A), (B), and (C), respectively. As can be observed in Figure (C), the robot could follow human movement best, generating the minimum prediction error against its learned memory, with the error regression α_{ER} set with the largest value. On the other hand, with of α_{ER} set with the smallest value, the robot attempted to shift to its own prior learned pattern at the transition point, trying to move to MP2 against the human generated pattern of MP1 and thereby generating the largest error as seen in Figure (A). Subtle balance between the top-down proactive intention from the learned memory and the bottom-up error regression of the sensory reality perceived in human subject movement can be observed in Figure (B) wherein α_{ER} was set with an intermediate value. After repeated experiments, the human subject reported that he felt that with a large α_{ER} value he could drive the robot at will. On the other hand, he felt as if the robot possessed its own will to generate sequential movement patterns with a small α_{ER} value. It was reported that the most vivid and lively interaction emerged with an intermediate α_{ER} value. And interestingly, the subject reported that the robot responded to his movement patterns by generating unexpected modulations in movement patterns from time to time. The MSEs over the whole time steps of three adaptation rates are shown in Table 2. Small, Middle and Large correspond to α_{ER} values of 0.00001. 0.0001, 0.001, respectively. These results conform to the results illustrated in Figure 12.

5 Discussion and conclusion

The preceding demonstrates how a RNN-based dynamic predictive coding model can cope with fluctuations such as time warping and amplitude shifting. We conducted a set of simulation and robot experiments under different conditions to examine the performance of a deterministic MTRNN model and also to investigate the mechanisms enabling the model to cope with fluctua-

tions latent in perceived temporal patterns. Periodicity or amplitude of a set of sine-curve patterns was fluctuated in the first experiment. One MTRNN was trained to reconstruct time-warped patterns and another MTRNN was trained to reconstruct amplitude-shifted patterns. Results show that these models are able to perform synchronized imitation of given test patterns with minimal error if the test patterns contain fluctuations in the same dimensions as those in which the models had been trained.

Computer-generated patterns without perturbations, and patterns generated by a human mimicking these computer-generated patterns using a drawing tablet, were used in the second experiment. Human generated patterns include natural fluctuations in amplitude and periodicity. One MTRNN, MTRNN-H, was trained to reconstruct the naturally fluctuating patterns and another, MTRNN-C, was trained to reconstruct the perfectly regular computer-generated patterns. During testing, the performance of the error regression during synchronized imitation with fluctuated patterns showed that MTRNN-H outperforms MTRNN-C in natural, noisy environments. Analysis of the dynamic structures developed during learning revealed that two sets of trained prototypical patterns emerge as different invariant sets in both MTRNN-H and MTRNN-C. The MTRNN-H was able to regenerate fluctuated training patterns in the transient region outside of the invariant set. Also, error regression could infer both this transient region and the local attractor for each prototypical pattern during the process of synchronized imitation in MTRNN-H. In conclusion, particular dimensions of fluctuation can be extracted via learning by developing an adequate dynamical map between the distribution of the initial states of the context units in all levels and the fluctuated temporal patterns of the exemplar. Each of the fluctuated training patterns can be reconstructed along with the transient trajectory of internal state dynamics which develop over time from the corresponding initial state. On the other hand, MTRNN-C was trained only with clean computer generated patterns and was unable to perform synchronized imitation with fluctuated patterns because the model cannot develop the transient region which can cope with fluctuated input patterns.

Although it had been established that the MTRNN is capable of learning the structures underlying naturally fluctuating exemplar temporal patterns [32], the current study clarifies the underlying mechanism in robotic studies for the first time. In the first experiment using the NAO humanoid robot, the MTRNN was trained to regenerate three prototypical cyclic movement patterns. The robot was prepared with exemplar patterns exhibiting natural

fluctuation through the direct teaching method. An imitative interaction game was designed wherein 10 human subjects without any prior knowledge about the patterns and experiments were asked to figure out the three given movement patterns by actively synchronizing with the robot. 8 out of 10 subjects could figure out the patterns, and synchronize by the end of the imitation game. We speculate that successful imitation interaction could be achieved between the robot and the human subjects for two reasons. One, the structure of the feedback during interaction with the robot makes sense to human beings due to a similar structure enabling predictive coding around anticipations due to trained sensitivities to patterns present in the immediacy and backpropagated through time in reflection. And two, the MTRNN was able to cope with fluctuated patterns naturally developed in the course of the imitative interaction by extracting this fluctuations in this way. Finally, analyses shows that the error regression scheme applied to context units in all levels outperformed models which applied error regression only to the slow context units, as well as models using the open-loop sensory entrainment scheme.

The second robot experiment manipulated the error regression adaptation rate α_{ER} in order to examine how balance between the top-down proactive intention and the bottom-up error regression of the sensory inputs can affect natural imitative interaction between robots and human subjects. The MTRNN model was trained to generate a cyclic sequence concatenating three different prototypical movement patterns. With the larger α_{ER} value, the human subject tends to drive the interaction. Set with smaller α_{ER} value however, the robot tends to drive the interaction based on its memory, instead. More natural and spontaneous interaction took place with α_{ER} tuned to an intermediate value, with balance achieved between the top-down intentional process and the bottom-up error regression process.

Compare the imitative interaction of the RNNPB model in [6] and that of the MTRNN used in the current study. In the former study, the RNNPB learned four different naturally fluctuated prototypical movement patterns. While learning, the PB vector for each prototypical pattern and the connectivity weights of the RNNPB were determined using the BPTT scheme. This learning scheme makes it almost impossible for the RNNPB to extract structures of fluctuations latent in training patterns because there are no means to account for perceived fluctuations - the PB vector is fixed for all steps during each sequence. Although with the PB vector determined, the RNNPB can regenerate the mean trajectory of each prototypical cyclic pat-

tern, it cannot regenerate fluctuations apparent in each cycle. On the other hand, the currently proposed MTRNN model can learn to regenerate fluctuations associated with prototypical exemplar patterns by determining the initial states for the context units in all levels of sub-networks. By developing particular dynamic structures that can embed the mean trajectories of the prototypical patterns in local attractors and their possible fluctuations in adjacent transient regions, the proposed MTRNN model becomes able to imitate test patterns in so far as test patterns exemplify the same class of fluctuation as did trained ones. It does this by means of adequately modulating internal states in the transient region by way of the error regression scheme.

For future study, it should be interesting to compare the deterministic predictive coding scheme presented here and the Bayesian predictive coding scheme proposed by Friston [2]. For this purpose, the aforementioned stochastic MTRNN [10] may be enabled to manipulate stochasticity not only in the outputs but also in the context units in all levels of sub-networks. So developed, it should be worthwhile examining how structures of fluctuation latent in exemplar temporal patterns can be extracted through the development of an adequate stochastic neurodynamics structure. Then, we can compare its performance against that of the deterministic neurodynamic structure which has been the subject of the current paper.

A Demonstration video of detailed dynamic of the error regression

This video is corresponding to Figure 10 and it shows the whole time steps of detailed dynamic of the error regression.

B Figure of training patterns generated by a computer without perturbation

Training patterns generated by a computer when (A) A Lissajous-curve patterns was produced and (B) a circle-curve patterns was produced. The first and second rows illustrate the first (y_1) and second (y_2) dimensions of the patterns, respectively. The last row shows the phase plots of training patterns

1 and 2.

Acknowledgement

This work was suppoeted by the ICT R&D program of MSIP/IITP. [2016(R7117-16-0163), Intelligent Processor Architectures and Application Software for CNN-RNN].

References

References

[1] J. Tani, M. Ito, Y. Sugita, Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using rnnpb, Neural Networks 17 (8) (2004) 1273–1289.

[2] K. Friston, The free-energy principle: a unified brain theory?, Nature Reviews Neuroscience 11 (2) (2010) 127–138.

[3] R. P. Rao, D. H. Ballard, Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects, Nature neuroscience 2 (1) (1999) 79–87.

[4] J. Tani, M. Ito, Self-organization of behavioral primitives as multiple attractor dynamics: A robot experiment, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 33 (4) (2003) 481–488.

[5] Y. Yamashita, J. Tani, Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment, PLoS Comput Biol 4 (11) (2008) e1000220.

[6] M. Ito, J. Tani, On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system, Adaptive Behavior 12 (2) (2004) 93–115.

[7] G. Rizzolatti, L. Fadiga, V. Gallese, L. Fogassi, Premotor cortex and the recognition of motor actions, Cognitive brain research 3 (2) (1996) 131–141.

[8] J. J. Hopfield, C. D. Brody, What is a moment? transient synchrony as a collective mechanism for spatiotemporal integration, Proceedings of the National Academy of Sciences 98 (3) (2001) 1282–1287.

[9] S. Murata, J. Namikawa, H. Arie, S. Sugano, J. Tani, Learning to reproduce fluctuating time series by inferring their time-dependent stochastic properties: Application in robot learning via tutoring, IEEE Transactions on Autonomous Mental Development 5 (4) (2013) 298–310.

[10] S. Murata, Y. Yamashita, H. Arie, T. Ogata, S. Sugano, J. Tani, Learning to perceive the world as probabilistic or deterministic via interaction with others: A neuro-robotics experiment, IEEE Transactions on Neural Networks and Learning Systems (2015) 1–18.

[11] K. Friston, J. Mattout, J. Kilner, Action understanding and active inference, Biological cybernetics 104 (1-2) (2011) 137–160.

[12] K. J. Friston, J. Daunizeau, J. Kilner, S. J. Kiebel, Action and behavior: a free-energy formulation, Biological cybernetics 102 (3) (2010) 227–260.

[13] W. Gerstner, W. Kistler, Spiking Neuron Models: An Introduction, Cambridge University Press, New York, NY, USA, 2002.

[14] M. Jeannerod, The representing brain: Neural correlates of motor intention and imagery, Behavioral and Brain sciences 17 (02) (1994) 187–202.

[15] J. Tani, Model-based learning for mobile robot navigation from the dynamical systems perspective, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (3) (1996) 421–436.

[16] Y. A. LeCun, L. Bottou, G. B. Orr, K.-R. Müller, Efficient backprop, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 9–48.

[17] Y. LeCun, et al., Generalization and network design strategies, Connectionism in perspective (1989) 143–155.

[18] C. M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

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

[20] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, in: D. E. Rumelhart, J. L. McClelland, C. PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.

[21] B. Dufay, J.-C. Latombe, An approach to automatic robot programming based on inductive learning, The International journal of robotics research 3 (4) (1984) 3–20.

[22] A. Levas, M. Selfridge, A user-friendly high-level robot teaching system, in: Robotics and Automation. Proceedings. 1984 IEEE International Conference on, Vol. 1, IEEE, 1984, pp. 413–416.

[23] A. Segre, G. DeJong, Explanation-based manipulator learning: Acquisition of planning ability through observation, in: Robotics and Automation. Proceedings. 1985 IEEE International Conference on, Vol. 2, IEEE, 1985, pp. 555–560.

[24] R. Dillmann, M. Kaiser, A. Ude, Acquisition of elementary robot skills from human demonstration, in: International symposium on intelligent robotics systems, Citeseer, 1995, pp. 185–192.

[25] J. Demiris, G. M. Hayes, Imitation as a dual-route process featuring predictive and learning components: A biologically plausible computational model, in: K. Dautenhahn, C. L. Nehaniv (Eds.), Imitation in Animals and Artifacts, MIT Press, Cambridge, MA, USA, 2002, pp. 327–361.

[26] A. Billard, Imitation: A means to enhance learning of a synthetic protolanguage in autonomous robots, in: K. Dautenhahn, C. L. Nehaniv (Eds.), Imitation in Animals and Artifacts, MIT Press, Cambridge, MA, USA, 2002, pp. 281–310.

[27] H. Arie, T. Arakaki, S. Sugano, J. Tani, Imitating others by composition of primitive actions: A neuro-dynamic model, Robotics and Autonomous Systems 60 (5) (2012) 729–741.

[28] A. Vakanski, I. Mantegh, A. Irish, F. Janabi-Sharifi, Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42 (4) (2012) 1039–1052.

[29] T. Oates, L. Firoiu, P. R. Cohen, Clustering time series with hidden markov models and dynamic time warping, in: Proceedings of the IJCAI-99 workshop on neural, symbolic and reinforcement learning methods for sequence learning, Citeseer, 1999, pp. 17–21.

[30] O. Deshmukh, C. Y. Espy-Wilson, A. Juneja, Acoustic-phonetic speech parameters for speaker-independent speech recognition, in: Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on, Vol. 1, IEEE, 2002, pp. I–593.

[31] A. Graves, Sequence transduction with recurrent neural networks, arXiv preprint arXiv:1211.3711.

[32] R. Nishimoto, J. Tani, Development of hierarchical structures for actions and motor imagery: a constructivist view from synthetic neuro-robotics study, Psychological Research PRPF 73 (4) (2009) 545–558.

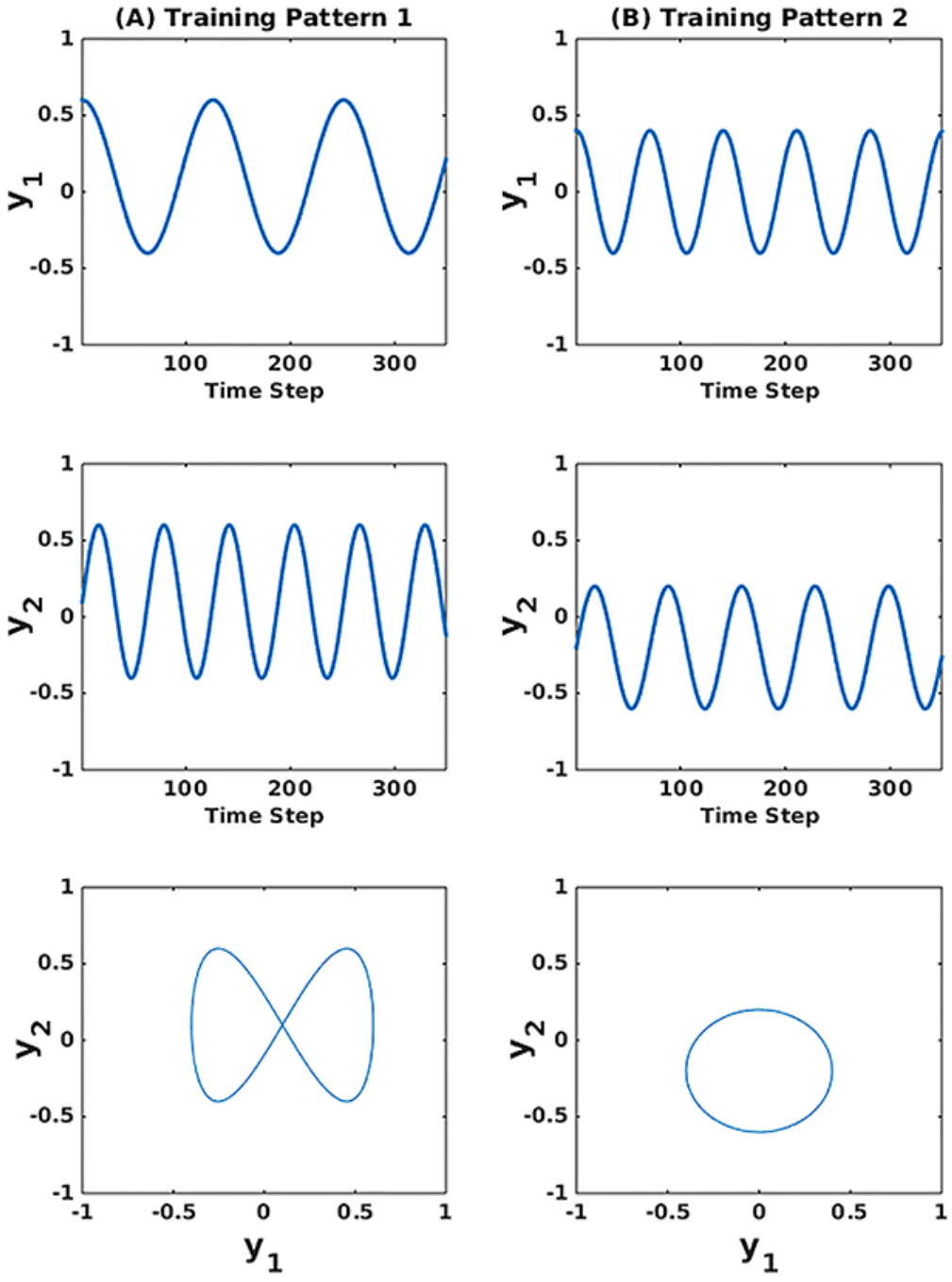

Figure B.1: Training patterns generated by a computer when (A) A Lissajous-curve patterns was produced and (B) a circle-curve patterns was produced. The first and second rows illustrate the first (y_1) and second (y_2) dimensions of the patterns, respectively. The last row shows the phase plots of training patterns 1 and 2.