Multiple Timescale and Gated Mechanisms for Action and Language Learning in Robotics

Wenjie Huang
School of Engineering
University of Manchester
Oxford Road, Manchester, UK
huangwj96@gmail.com

Junpei Zhong
School of Science and Technology,
Nottingham Trent University
Clifton, NG11 8NS, Nottingham, UK
zhong@junpei.eu

Angelo Cangelosi
School of Engineering
University of Manchester; AIST-AIRC
UK; Japan
angelo.cangelosi@manchester.ac.uk

***Abstract*—Recurrent Neural Network (RNN) have been used for sequence-related learning tasks, such as language and action, in the field of cognitive robotics. Gated mechanisms used in LSTM and GRU perform well in remembering long-term dependency. But to better mimic the neural dynamics in cognitive processes, the Multiple Time-scales (MT) RNN uses a hierarchical organization of memory updates which is similar to human cognition. Since the MT feature is typically used with a vanilla RNN or different gated mechanisms, its effect on the updates and training is still not fully uncovered. Therefore, we conduct a comparative experiment on two MT recurrent neural network models, i.e. the Multiple Time-Scale Recurrent Neural Network (MTRNN) and the Multiple Time-Scale Gated Recurrent Unit (MTGRU), for action sequence learning in robotics. The experiment shows that the MTRNN model can be used in learning tasks with low requirements for learning of long-term dependency due to its low computation. On the other hand, the MTGRU model is appropriate for learning the long-term dependency. Furthermore, because of the duplicated feature of the MT and the GRU feature, we also propose a simplified MTGRU model, named Multiple Time-scale Single-Gate Recurrent Unit (MTSRU) which could reduce computational cost while it achieves the similar performance as the original version.**

Keywords—cognitive robotics, Neural Network, Multiple Time-scale, MTSRU

I. INTRODUCTION

Time is important in both generating behaviour and action patterns [1-2] and in the neuro-dynamics to represent different levels of cognitive functions for intelligent agents [3]. Recurrent Neural Networks (RNNs) have been widely used to learn temporal sequences of data. The performance of vanilla RNN can be degraded dramatically when the learning tasks involve long-term dependency [4]. This is attributed to the optimisation process of back-propagation through time (BPTT). The BPTT process is based on the calculation of gradients of the errors, which, however, involves consecutive multiplication through time resulting in the gradient to be too small to affect the updates.

To solve this problem, the gated mechanism within the recurrent units have been proposed. For instance, the Long Short-Term Memory (LSTM) [5] and the Gated Recurrent Unit (GRU) [6] are two of the most common RNN architectures adopting the gated-mechanism. In the tasks of language modelling, motor control etc, they have proven to be able to alleviate the vanishing gradient problem of RNNs [7-9].

Nevertheless, the development of RNNs without vanishing gradients is not sufficient to build biologically- and cognitively-inspired neural systems. To build a bio-inspired intelligent agent, the critical mechanisms of the neuro-dynamics of the brain should be also modelled. For example, when we use humanoid robots to model cognitive agents, one of key goals and standards for testing neural mechanism modelling (neuro-robotics) is to simulate not only the human behaviours, but also the neural dynamics arisen from the stimuli perceived by the robotic sensors. Specifically, in the field of neuro-robotics models of language, one of the key research questions is how the neural dynamics facilitate the emergence of language in the forms of pre-symbolic representation while the agent is interacting with the world.

To study this, the Multiple Timescale (MT) recurrent neural network (MTRNN) [10] model was proposed as a functionally hierarchical architecture to mimic the hierarchical information processing in neural systems [11-14]. Having different time-scales in different neurons allows the robots to learn the primitive units from both perception and action and to compose these for higher-level pre-symbolic representation, such as the language learning. The timescale feature is realised by the updating of neural activities, using different time-scale constants representing different dynamic properties. Using a similar concept, combined with the gating mechanism of GRUs, the Multiple Timescale Gated Recurrent Unit (MTGRU) model was recently proposed [15].

In [16], a language learning model employing MTGRU was used in a robot language grounding experiment. This study reports that MTGRUs can slightly outperform MTRNNs, although the computation cost was almost double. However, the learning task in [16] does not have a real long-term dependency, as it consists of a one-step-ahead prediction. To better examine the MT feature in a neuro-robotic experiment with long-term dependencies, in this paper we will introduce challenging learning tasks with long-term dependency to examine the performances of the MT feature in both MTGRU and MTRNN units. Moreover, as in the MTGRU the functions for the time constants and updating gates are similar, we also propose a simplified version of MTGRU, which we call Multiple Time-Scale Single-Gate

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Recurrent Unit (MTSRU). This simplified version is examined in comparison to MTRNN and the original MTGRU in the robot learning setting.

To sum up, this paper has the following contributions:

1) The performance of MTRNN and MTGRU are analysed in both simple and challenging long-term dependency tasks, with which we can further investigate the role of the multiple time-scale feature in these hierarchical architectures.

2) Based on the finding of (1), a simplified MTGRU architecture, MTSRU is presented, resulting in a lower computational cost without dramatic degradation in the learning performance.

II. MODELS

In this section, the two existing Multiple Time-scale neural networks are introduced, with which we will investigate the property and performance of the MT feature. The new MTSRU model is then presented.

A. Multiple Time-Scale Recurrent Neural Network (MTRNN)

MTRNN stores the temporal information of the learned sequences in the recurrent connections on different layers by introducing the Multiple Time-scale concept to RNNs, and organizing the neurons with the same time-scale on the same layer in a hierarchical architecture. Specifically, in an MTRNN, the hidden layer is composed of several context layers with different time-scale constants. The constants determine the fast and the slow context layers: the fast context layer has a smaller time constant, causing the cells' states in this context layer to change fast, while the larger time constant of the slow context layer allows the corresponding cell's state to update slowly. This way the network can make use of previous temporal information and control the decay of neural states [17]. Usually, the fast context layer is located between the input and the slow context layers, and it is fully connected with these. Via these connections, the computation of the fast context layer at t time step is determined by the input signal x_t, the previous neural activity $h_f\ (t-1)$ and the inputs from the connecting slow context layer h_s. The computation for the fast context layer is as below.

$$\tilde{h}_{f,t} = \sigma\left(W_f x_t + U_f h_{f,t-1} + U_{sf} h_{s,t-1} + b_f\right) \quad (1)$$

$$h_{f,t} = \frac{1}{\tau}\tilde{h}_{f,t} + \left(1 - \frac{1}{\tau}\right) h_{f,t-1} \quad (2)$$

where W_f, U_f and U_{sf} are weight matrices, b_f is the bias parameter, and $h_{f,t-1}$ and $h_{s,t-1}$ are the hidden state of both context layers at $t-1$ time step. The non-linear function σ is *tanh* here and τ is the time constant for this layer.

As for the slow context layer, this is usually located on top of the fast context layer, and it only has connections with the fast layer. The computation process is the same as for the common RNN, except for the use of the time constant. The input signal for the slow context layer, if it is on the topmost layer, is the output of the fast layer, $h_{f,t}$. The calculation is as below:

$$\tilde{h}_{s,t} = \sigma\left(W_s h_{f,t} + U_s h_{s,t-1} + b_s\right) \quad (3)$$

$$h_{s,t} = \frac{1}{\tau}\tilde{h}_{s,t} + \left(1 - \frac{1}{\tau}\right) h_{s,t-1} \quad (4)$$

The overall architecture of MTRNN is as below:

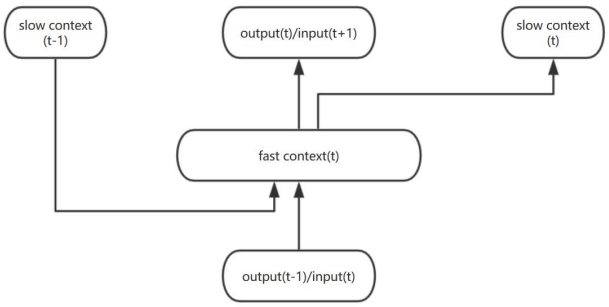

Fig. 1. The Multiple Time-Scale RNN models

During training, the output state of the fast context layer is passed to the slow context layer, as well as the output layer. The two context layers have different time constants: With the large time constant, it indicates that the neural states on the slow context layer are largely determined by the previous states as well as the states of the neurons on the lower level; and vice versa. The time constants also control how the previous signal influences the current state. Without the time-constants, the vanilla RNN can only store a short range of temporal sequences. On the other hand, with the differences of time constants, the MTRNN model is able to extract the temporal sequences in terms of their temporal features and store those with the proportional summation of the historical trace determined by the time constants. Furthermore, this long-term information store on the slow context layer is the primitive information of the input sequences, e.g. the sensorimotor status.

B. Multiple Time-Scale Gated Recurrent Unit (MTGRU)

Using a similar concept of multiple time-scale as with the MTRNN, the MTGRU model shares a similar structure which contains context layers with different time constants [15]. Same as the spreading of information in MTRNNs, the state information of slow context layers is used for the computation of the cell state of the fast context layer. The states of the context neurons decay along time, as determined by the time constants.

At the same time, the MTGRN inherits the gated mechanism from the GRU model. For the output, the internal memory of the neurons is composed of the calculations of the gated mechanism, reset gate, update gate and current cell state. The reset and update gates are determined only by the current input signal x_t and the previous hidden state $h_{f,t}$ of the fast context layer.

The reset and update gates are computed by Eq. (5) and (6) respectively:

$$z_t = \sigma\left(W_z x_t + U_z h_{f,t-1} + b_z\right) \quad (5)$$

$$r_t = \sigma\left(W_r x_t + U_r h_{f,t-1} + b_r\right) \quad (6)$$

The result of the reset gate r_t will then be used to calculate the current information representation along with $h_{f,t-1}$ and $h_{s,t-1}$, the hidden cell states of both layers at last time step:

$$u_{f,t} = \sigma\left(W_u x_t + U_u(r_t \otimes h_{f,t-1}) + U_{sf} h_{s,t-1} + b_u\right) \quad (7)$$

Below is the updating equation for hidden state involving the time constant [15]:

$$\tilde{h}_{f,t} = (1 - z_t) \otimes h_{f,t-1} + z_t \otimes u_{f,t} \quad (8)$$

$$h_{f,t} = \frac{1}{\tau}\tilde{h}_{f,t} + \left(1 - \frac{1}{\tau}\right) h_{f,t-1} \quad (9)$$

where $\otimes$ indicates the element-wise computation, other symbols are the same as the ones of the MTRNN model. As for the computation of the slow context layer, the definition of the fast context layer is the same as in MTRNN, in which the fast context layer is determined by the three input signals including the one from the slow context layer, while the slow one only uses the output of last layer and the previous state of itself.

C. Multiple Time-Scale Single-Gate Recurrent Unit (MTSRU)

We can reconsider the internal structures and training of the MTGRU model as the similar linear computation between the Multiple Time-Scale and gate mechanism.

Firstly, we analyse the functionality of the gate mechanism and how it avoids the gradient vanishing. Similar as [5], a kind of derivative of the calculation is shown below for clarification, on how the gradient of the network parameters in the LSTM model can be computed:

$$\frac{\partial E}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t^*}\frac{\partial y_t^*}{\partial h_t}\frac{\partial h_t}{\partial \tilde{h}_t}\frac{\partial \tilde{h}_t}{\partial W} \quad (10)$$

where the E_t is the error at time step t, y_t^* is the prediction, h_t is the hidden state while $\tilde{h}_t$ is the cell state, w is a parameter of the network, i.e. the weight matrix of the ***forget gate***. The complex here leading to the gradient vanishing is $\frac{\partial \tilde{h}_t}{\partial W}$ if explicitly write it as:

$$\frac{\partial \tilde{h}_t}{\partial w} = \frac{\partial \tilde{h}_t}{\partial f_t}\cdot\frac{\partial f_t}{\partial w} + \frac{\partial \tilde{h}_t}{\partial \tilde{h}_{t-1}}\cdot\frac{\partial \tilde{h}_{t-1}}{\partial f_{t-1}}\cdot\frac{\partial f_{t-1}}{\partial w} + \frac{\partial \tilde{h}_t}{\partial \tilde{h}_{t-1}}\cdot\frac{\partial \tilde{h}_{t-1}}{\partial \tilde{h}_{t-2}}\cdot\frac{\partial \tilde{h}_{t-2}}{\partial f_{t-2}}\cdot\frac{\partial f_{t-2}}{\partial w} + \cdots + \prod_{j=1}^{t}\frac{\partial \tilde{h}_j}{\partial \tilde{h}_{j-1}} \quad (11)$$

More explicitly, we can further unfold the term $\frac{\partial \tilde{h}_t}{\partial \tilde{h}_{t-1}}$ as:

$$\frac{\partial \tilde{h}_t}{\partial \tilde{h}_{t-1}} = \tilde{h}_{t-1}\sigma'(\cdot)W_{f^*}o_{t-1}(\tilde{h}_{t-1}) + a_t\sigma'(\cdot)W_{i^*}o_{t-1}(\tilde{h}_{t-1}) + i_t\sigma'(\cdot)W_{a^*}o_{t-1}(\tilde{h}_{t-1}) + f_t \quad (12)$$

where a_t, i_t, f_t are the gate signals of LSTM and they have three weight matrices at each gate. The value o_{t-1} is the network output at time step $t-1$. When the time steps are large, the gradients of weight matrices do not result in vanishing gradient, since the network can easily set the value of the forget gate f_t to bring the gradient closer to 1. What must be noted is that the recursive derivative for the forget gate of LSTM cannot totally avoid small values between 0 and 1 but is highly avoided. Thus, by the forget gate, the vanishing gradient problem can be significantly alleviated.

Furthermore, noticing the update equation for the hidden state for the MTGRU model, the functions of the update signal and the time constant are similar (both of them are linear mappings). Hence, (8) and (9) can be combined into a linear function.

Given the above-mentioned reasons, we propose to remove the update gate. Thus, the calculation of hidden state can be written as below (take the fast context as an example):

$$h_{f,t} = \frac{1}{\tau}u_{f,t} + \left(1 - \frac{1}{\tau}\right) h_{f,t-1} \quad (13)$$

Although the update signal provides more flexible updating for each neuron, the computation of optimising the weight matrix for this signal is expensive when the number of hidden units is large. Moreover, the target for gated architectures to eliminate the gradient vanishing problems depends on the ***forget gate***. Therefore, the modification to the ***update gate*** would not destroy the learning ability for long-term dependencies in such models. Therefore, we claim that MTSRU has the similar functionality of the original version of MTGRU.

III. EXPERIMENT

This section will first introduce the experiment dataset. Also, the experimental plan including the prospective hypotheses and the evaluation standards for the model performance will be presented. Finally, the experiment results are discussed.

A. Experiment Dataset

The dataset for this comparative experiment was recorded by Zhong et al. [18] from the object manipulation experiment on the iCub robot [19]. The iCub is a robot with the size and appearance to a 2.5-year old human child and essential open access hardware and software to develop cognitive modules [20]. It has a total of 53 degrees of freedom. In the work of Zhong et al. [18], the iCub was taught to learn the meaning of natural language commands with the sensorimotor demonstrations. The dataset includes the angles of the upper torso joints (35) and of the neck and eyes (6), as described in [18].

Table 1 is the lexicon for human command translation which was produced by word embedding techniques.

TABLE 1. Lexicon of 9 verbs (for actions) and 9 nouns (for objects) used in the commands

Actions	Slide Left	Slide Right	Touch	Reach	Push
Verb Value	0.0	0.1	0.2	0.3	0.4
Actions	Pull	Point	Grasp	Lift	
Verb Values	0.5	0.6	0.7	0.8	
Objects	Tractor	Hammer	Ball	Bus	Modi
Noun Value	0.0	0.1	0.2	0.3	0.4
Objects	Car	Cup	Cubes	Spiky	
Noun Values	0.5	0.6	0.7	0.8	

The aim of this experiment is to associate a language command which contains one verb and one noun with the motor action and the visual feature, respectively. For the

robotic experiment, the spoken language command is the first input for the iCub robot. The object referred to (noun) in the command is located by the visual tracking module embedded in the robot's open-access software toolbox. The sequence of the joint angles is used and recorded to produce the corresponding motor action (verb). The language commands are composed of two discrete words: The 9 actions and 9 objects were combined to produce 81 possible commands pairs. The first input, about the language command, uses two words. The encoding of each word corresponds to a real number in the range of 0 and 0.9 as shown in Table 1. As the second input on about the object, the neck and eyes angles were used to describe the location of the object in the visual system of this robot, encoding it as 6 real numbers. The third input about the action of the robot uses a total number of 35 joints, consisting of 3 torso joints, 16 left and 16 right arm joints. Hence, the total number of dimensions of the input data is 43, as shown in Table 2. Moreover, the recording of a complete movement sequence was taken every 5 seconds with an interval of 50ms. Thus, the whole sequence contains 100 temporal status vectors.

TABLE 2. The structure of data from Zhong et al, 2019

Description	Semantic Commands	Object Location (Neck and Eyes)	Torso Joints
Dimension	2	6	3
Description	Left Arm Joints	Right Arm Joints	
Dimension	16	16	

B. Network Training Experiments

We conducted two experiments to examine our hypotheses of the MT features. We will firstly introduce the experimental settings, and then the evaluation of these models will be given.

1) Experiment Outline & Hypotheses The training process in the work of Zhong et al. [16] is based on the encoder-decoder architecture, where the encoder has inputs from the current time-step and the decoder has a one-step-ahead prediction. In our study, although using the same dataset, a different training task called *trajectory generation* is defined. In this training task, only the first sequence vector, which indicates the language command, object location and the initial state or the information of physical positions of the robot, is needed. By processing this starting state information, the network attempts to generate the whole movement trajectory for fulfilling the requested action. Compared to the encoder-decoder training task, the prediction of each step becomes the next input to the network. As such, this significantly increases the difficulty of the learning task.

Firstly, based on the two learning situations discussed above, the performances of MTRNN and MTGRU in both contexts are compared. Based on the experiments done in [16], the self-adaptive optimisation algorithm is introduced to improve the training process. The convergence performance will be compared. Also, the results of the prediction mode, which require a higher learning ability to retain long-term dependency, is presented.

Moreover, the performance of our proposed MTSRU model is examined in the context of the trajectory generation task. As a modified version of MTGRU, the emphasis of this sub-experiment is on the two questions: 1) Would the modification to the model degrade the learning ability? And 2) How much reduction in the overall computation cost is achieved? Our hypothesis is that the performance of the simplified version will be slightly degraded, although it will remain above an acceptable level. The computation saving will be analysed in Section 5.

2) Evaluation Method To quantify the performances and computation complexities of the three models, three evaluation terms are set.

Firstly, we will examine the accuracy of the model, which is the most important benchmark for a learning model. This is evaluated by the root mean square error (RMSE) [21] of the predictions, which is defined as below:

$$RMSE = \sqrt{\frac{1}{S*N}\sum_{t=0}^{S} \quad \sum_{k=0}^{N} \quad (y_k^* - y_k)^2} \tag{8}$$

where S is the number of time steps which is 99 in the sequence data. N is 43 which is the total number of dimensions of the data. y_k^* indicates the predicted vector and y_k is the real value for the predictions.

The second evaluation term is the time cost of the models. The elapsed time of each model for each epoch is recorded and compared as this property is critical for researchers to select learning algorithms for their tasks when the time cost should be limited.

Next, although the elapsed time reflects the computation cost of the models, a more straightforward way to quantify the number of parameters is needed in the experiments. More precisely, this evaluation metric should also consider the dimensionalities of different parameters. Hence, the total computation complexity of a model is calculated by (9) as below:

$$C = \sum_{i=1}^{N} \quad \prod_{j=1}^{D} \quad d_{ij} \tag{9}$$

Where d_{ij} represents the length of i th parameter on the dimension j. N is the number of parameters. D indicates the dimensionality of this parameter.

Finally, other issues such as the convergence rate and the property of stability for models should be also considered when choosing an appropriate model for machine learning tasks. These evaluation methods will also be used in the analyses and discussion of the experiment results and in Section 5.

C. Experiment Results

***Case 1**: Comparison of the performances of MTRNN and MTGRU for the training task in the encoder-decoder model.*

The performances of these models are shown in Table 3, and their time costs are shown in Table 4, and the common

training curves of these three models in this task is illustrated in Figure 2.

TABLE 3. Performances of three models in the encoder-decoder task with three repetitive experiments within the same context

	experiment 1	experiment 2	experiment 3	average
MTRNN	2.91E-05	2.80E-05	2.94E-05	2.88E-05
MTGRU	1.06E-05	1.15E-05	1.12E-05	1.11E-05

TABLE 4. The time costs of three models in the encoder-decoder task (Run by Tensorflow with GPU-NVIDIA GeForce MX150 2GB)

	Time Cost (s) / epoch
MTRNN	15.62
MTGRU	38.92

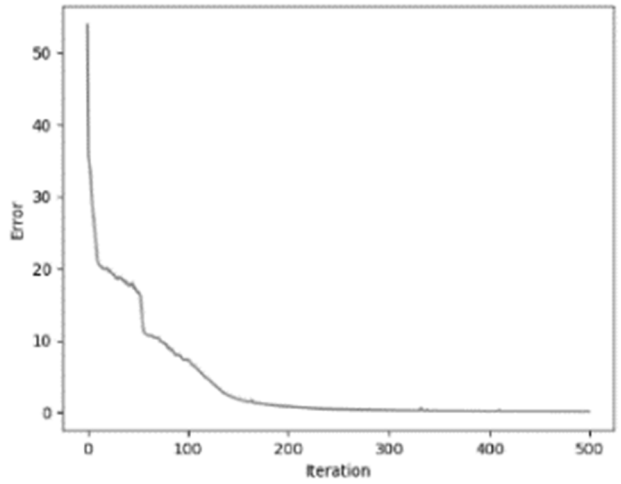

Fig. 2a. MTRNN curve

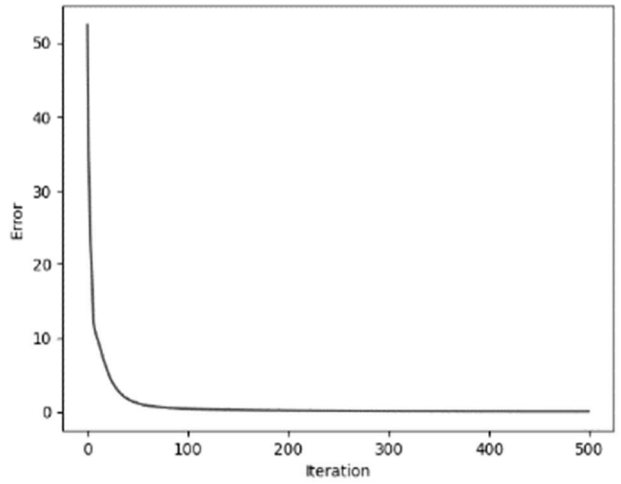

Fig. 2b. MTGRU curve

Fig. 2. The training curves of two models in the encoder-decoder task

Fig. 2a shows the training curve of the MTRNN model and Fig. 2b shows that of the MTGRU model. In the above experiment on the encoder-decoder task, the performance of the MTGRU is better, while both models generate a low error. Although the accuracy of the MTRNN model is lower, the time cost of the MTGRU model is significantly higher, more than double the time cost of the MTRNN model. Furthermore, the convergence curve of the MTGRU model is very smooth, while that of the MTRNN model has sudden spikes. Meanwhile, the convergence rate shown by the training curve of the MTGRU is higher than the MTRNN model. The MTGRU model converges at around the 100th epoch, while the convergence point for the MTRNN model is after 200 epochs. In addition, compared to the training performance in [16], the adaptive optimisation generates much more smooth curves and better convergence trends.

***Case 2**: Comparison of MTRNN, MTGRU and MTSRU for the trajectory generation task.*

The performances of these three models for trajectory generation task are shown in Table 5, the time costs in Table 6, and the training curves of the three models is illustrated in Figure 3.

TABLE 5. Performances of three models in the trajectory generation task

	experiment 1	experiment 2	experiment 3	average
MTRNN	6.39E-03	6.46E-03	6.48E-03	6.44E-03
MTGRU	1.14E-05	1.11E-05	1.15E-05	1.13E-05
MTSRU	1.61E-05	1.78E-05	1.69E-05	1.69E-05

TABLE 6. The time costs of three models in the trajectory generation task

	Time Cost (s) / epoch
MTRNN	15.15
MTGRU	39.05
MTSRU	37.36

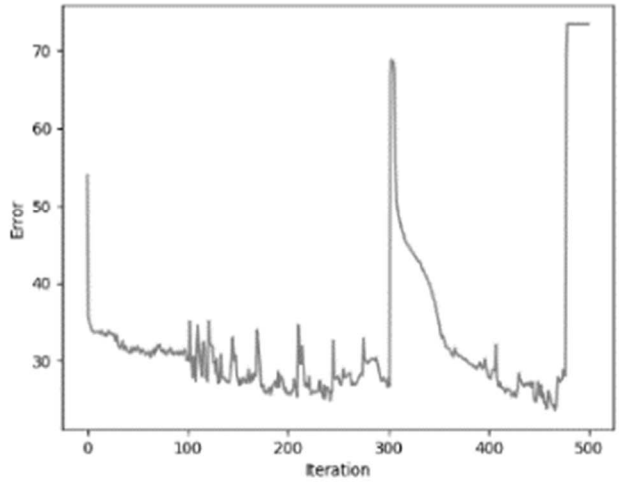

Fig. 3a. MTRNN curve

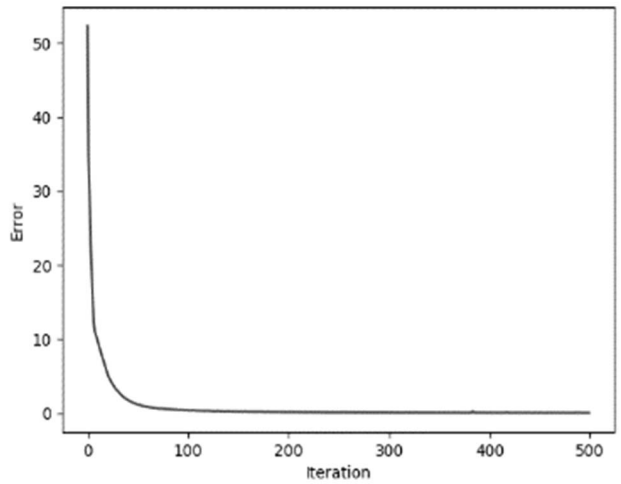

Fig. 3b. MTGRU curve

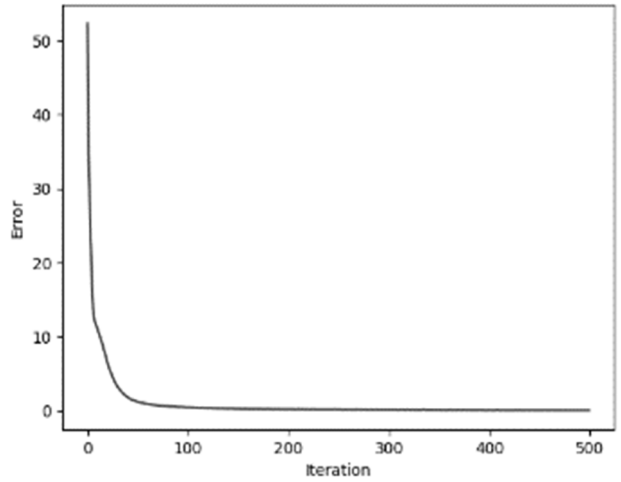

Fig. 3c. MTSRU curve

Fig. 3. The training curves of three models in the trajectory generation task

The second curve, for the MTGRU model, keeps its good learning ability as in the encoder-decoder task, while the performance of the MTRNN model (the first curve) dramatically degrades. As for the time cost, there is no obvious difference amongst them. However, the training curves show that the fluctuation of the MTRNN model is remarkable. On the contrary, the training curve of the MTGRU model remains stable and smooth as in the first case experiment. When it comes to the last curve of the MTSRU model, this shows almost all the distinctions of its original version. Noticeably, the slightly degraded accuracy is acceptable and stays at a satisfactory level. The time cost for each Gradient Descent epoch is also reduced.

The computational complexity of the models is also compared, especially to quantify the computation saving of the MTSRU model. This term is calculated as introduced in (9) and the comparison is shown in Table 7.

TABLE 7. The computational complexity of three models

MTRNN	MTGRU	MTSRU
212649	583849	398249

For clarification, while calculating the computational complexity of these models, the biases are ignored since their computations are trivial to the training process. As shown, the MTRNN model is the simplest one while the MTGRU model contains the most parameters expected to be well trained. The MTSRU model reduces the computation workload much lower than the original model.

IV. CONCLUSION AND FUTURE WORK

In this study, in addition to the encoder prediction task in [16], the new generation task is conducted to compare the performances of the MTRNN and MTGRU models. A simplified version of the MTGRU model named MTSRU is also proposed. Through the experiment, the following conclusions can be drawn:

1. In the context without a high requirement for learning long-term dependency, the MTRNN model should be preferred for its low requirements of the computational effort.
2. While doing challenging work involving prediction, the MTGRU model is preferred as it can retain its learning accuracy and stability of its convergence. although it requires higher computation effort.
3. The MTSRU model also has its advantage over the original model in terms of its comparable accuracy and lower computation requirement in the robot manipulation setting.

The idea of Multiple Time-Scale is appreciable as its ability to model the information with time effect. However, there are no rules in stone helping us assert a method to be the best. The situation in development is usually of a trade-off between accuracy and cost. From this point, this work of MTSRU should shed light on the adaptation of cutting-edge approaches for the performance requirements in specific tasks.

REFERENCES

[1] Breazeal, C. L. (2004). *Designing sociable robots*. MIT press.

[2] Dautenhahn, K. (2007). Socially intelligent robots: dimensions of human–robot interaction. *Philosophical transactions of the royal society B: Biological sciences*, *362*(1480), 679-704.

[3] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179-211.

[4] Mesnil, G., He, X., Deng, L., & Bengio, Y. (2013, August). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech* (pp. 3771-3775).

[5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735-1780.

[6] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations

using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[7] Wu, Z., & King, S. (2016, March). Investigating gated recurrent networks for speech synthesis. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5140-5144). IEEE.

[8] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[9] Tang, Y., Huang, Y., Wu, Z., Meng, H., Xu, M., & Cai, L. (2016, March). Question detection from acoustic features using recurrent neural network with gated recurrent unit. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 6125-6129). IEEE.

[10] Yamashita, Y., & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS computational biology*, *4*(11), e1000220.

[11] Felleman, D. J., & Van, D. E. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex (New York, NY: 1991)*, *1*(1), 1-47.

[12] Hilgetag, C. C., O'Neill, M. A., & Young, M. P. (2000). Hierarchical organization of macaque and cat cortical sensory systems explored with a novel network processor. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, *355*(1393), 71-89.

[13] Fuster, J. M. (2001). The prefrontal cortex—an update: time is of the essence. *Neuron*, *30*(2), 319-333.

[14] Boemio, A., Fromm, S., Braun, A., & Poeppel, D. (2005). Hierarchical and asymmetric temporal sensitivity in human auditory cortices. *Nature neuroscience*, *8*(3), 389.

[15] Kim, M., Singh, M. D., & Lee, M. (2016). Towards abstraction from extraction: Multiple timescale gated recurrent unit for summarization. *arXiv preprint arXiv:1607.00718*.

[16] Zhong, J., Cangelosi, A., & Ogata, T. (2017, May). Toward abstraction from multi-modal data: empirical studies on multiple time-scale recurrent models. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 3625-3632). IEEE.

[17] Zhong, J., Ogata, T., Cangelosi, A., & Yang, C. (2017, September). Understanding natural language sentences with word embedding and multi-modal interaction. In 2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob) (pp. 184-189). IEEE.

[18] Zhong, J., Peniak, M., Tani, J., Ogata, T., & Cangelosi, A. (2019). Sensorimotor input as a language generalisation tool: a neurorobotics model for generation and generalisation of noun-verb combinations with sensorimotor inputs. Autonomous Robots, 43(5), 1271-1290.

[19] Metta, G., Sandini, G., Vernon, D., Natale, L., & Nori, F. (2008, August). The iCub humanoid robot: an open platform for research in embodied cognition. In *Proceedings of the 8th workshop on performance metrics for intelligent systems* (pp. 50-56). ACM.

[20] Tsagarakis, N. G., Metta, G., Sandini, G., Vernon, D., Beira, R., Becchi, F., ... & Caldwell, D. G. (2007). iCub: the design and realization of an open humanoid platform for cognitive and neuroscience research. *Advanced Robotics*, *21*(10), 1151-1175.

[21] Levinson, N. (1946). The Wiener (root mean square) error criterion in filter design and prediction. *Journal of Mathematics and Physics*, *25*(1-4), 261-278.