Continuous Timescale Long-Short Term Memory Neural Network for Human Intent Understanding

Zhibin Yu¹, Dennis S. Moirangthem² and Minho Lee²*

¹ Department of Electrical Engineering, College of Information Science and Engineering, Ocean University of China, Qingdao, China, ² School of Electronics Engineering, Kyungpook National University, Daegu, South Korea

Understanding of human intention by observing a series of human actions has been a challenging task. In order to do so, we need to analyze longer sequences of human actions related with intentions and extract the context from the dynamic features. The multiple timescales recurrent neural network (MTRNN) model, which is believed to be a kind of solution, is a useful tool for recording and regenerating a continuous signal for dynamic tasks. However, the conventional MTRNN suffers from the vanishing gradient problem which renders it impossible to be used for longer sequence understanding. To address this problem, we propose a new model named Continuous Timescale Long-Short Term Memory (CTLSTM) in which we inherit the multiple timescales concept into the Long-Short Term Memory (LSTM) recurrent neural network (RNN) that addresses the vanishing gradient problem. We design an additional recurrent connection in the LSTM cell outputs to produce a time-delay in order to capture the slow context. Our experiments show that the proposed model exhibits better context modeling ability and captures the dynamic features on multiple large dataset classification tasks. The results illustrate that the multiple timescales concept enhances the ability of our model to handle longer sequences related with human intentions and hence proving to be more suitable for complex tasks, such as intention recognition.

Keywords: continuous timescale, recurrent neural network, LSTM, classification, dynamic sequence

INTRODUCTION

In machine learning, dynamic sequence modeling is a burning research topic, which includes intention understanding, action recognition, language understanding, semantic understanding (Peniak et al., 2011; Wasser and Lincoln, 2012; Wonmin et al., 2015; Kim et al., 2017) etc. Unlike popular static models, such as Convolutional Neural Network (CNN) (LeCun et al., 1998) and Deep Belief Network (DBN) (Hinton and Salakhutdinov, 2006) that focus on the feature of the data without considering any time dependency, the dynamic models try to find the relationships between data following the time axis. Context, which is generally mentioned in language understanding (Ghadessy, 1999; Givón, 2005), also plays an important role in dynamic sequence classification. Context contains several physical and abstract aspects such as time, symbols, location, names, etc. to describe the background of dynamic signal. Same words may have different meaning under different contexts. In short, context plays the role of surroundings, which contains some inconspicuous but important descriptions of the current phenomenon. Context can be deemed as the key of the dynamic sequence learning.
Multiple Timescales Neural Network (MTRNN), developed by Tani et al. (2008), is believed to be efficient to hold the context of dynamic trajectories. MTRNN is a successive extension of Recurrent Neural Network (RNN). All biological neural networks are recurrent (Jaeger, 2002), which is one of the reasons to choose RNN for dynamic sequence modeling. MTRNN, in turn, consists of multiple Continuous Recurrent Neural Network (CTRNN) layers. Each CTRNN layer is allowed to have one or more different timescale constants. Different time constants imply different activation speeds. That is why this network is called “multiple timescales.” Inspired by the structure of human brain, MTRNN has been proved to be useful on goal-planning problems (Arie et al., 2009; Jeong et al., 2012).

There are several extensions of RNN such as Elman networks, Jordan network, etc. These extensions aim to improve the memory ability and the performance of RNN (Cruse, 2006) but suffer from the vanishing gradient problem (Hochreiter et al., 2001). Long Short Term Memory network (LSTM), developed by Hochreiter and Schmidhuber (1997), and promises to overcome this problem. Similar to most RNNs, LSTM also uses derivative based methods to evolve itself. LSTM uses several gates with different functions to control the neurons and store the information. LSTM cell has the ability to keep important information for a longer period it is used. This property of holding information allows LSTM to perform well on classifying, processing or predicting a complex dynamic sequence. Research has shown that LSTM can achieve better performance than Hidden Markov Model (HMM) along with other RNNs on several real-world problems, such as handwriting recognition (Graves and Schmidhuber, 2005; Baccouche et al., 2011; Graves et al., 2013). It has also been proved that RNN performs well in human action modeling (Schrodt and Butz, 2016; Bütepage et al., 2017a). Moreover, deep RNN structures are able to represent human motion and natural language (Bütepage et al., 2017b; Plappert et al., 2017). Thus, deep RNN is a good candidate to handle human motion and language modeling problems. But how to design an efficient deep RNN structure is still a challenging problem.

We intend to capture the context efficiently while overcoming the vanishing gradient problem, which is still existing in CTRNN and MTRNN. We propose a model considering the advantages of an LSTM and inheriting the biological idea given by CTRNN. The proposed Continuous Timescale Long-Short Term Memory (CTLSTM) builds a temporal hierarchy into the architecture that enhances the model's ability to solve long-term complex sequence modeling problems. We evaluate our model on multiple public datasets to compare with the baselines. We demonstrate the capability of our model in human action classification tasks as well as human intention recognition tasks which consist of longer multiple action sequences. Our results illustrate that our proposed model outperforms the existing baselines.

The remainder of this paper is organized as follows: The proposed model is described in Section Proposed Model. The experiments and results are reported in Section Experiments and Results. Finally, the conclusion and discussion are presented in Section Conclusion and Discussion.

PROPOSED MODEL

We describe the proposed model in this section including the background study as well as the motivation of our model.

Motivation

Dynamic sequence, in general, is a number set (vector) combination in which each vector has a given time or spatial coordinates. A dynamic model can also be considered as a set of relationships between two or more measurable quantities. It relies on one or more fixed rules to describe how the dynamic model works and evolves itself. At any given point of time, a dynamic system has a state given by a set of real numbers (a vector) that can represent the current situation.

Inspired by MTRNN and LSTM, we aim to develop a RNN with multiple timescales structure with better ability to capture the dynamic features in longer sequences such as a series of human actions for understanding human intention. Time constants, which are the key of CTRNN, can be defined separately for each neuron node. Different time constants lead to different neuron activation abilities. For example, neurons with large timescale will activate slowly. That means slow neuron will become inactive to some short-term signals. Once the neuron starts firing, it would last for a longer time according to its timescale. Based on the results of previous research (Tani et al., 2008; Arie et al., 2012; Jeong et al., 2012; Yu and Lee, 2015a,b), we believe that different time scales would bring benefits for dynamic signal modeling. Thus, to inherit the advantages of MTRNN, the model is designed with different time scales. Layers with different time constants work differently. Layers with slower time constants would focus on signal organization and planning, while layers with faster time constants can implement the elemental dynamic sequences. Yu and Lee (2015a) and Kim et al. (2017) have already demonstrated the use of MTRNN in motion based intent recognition tasks. On the other hand, LSTM, which has a more complex structure than the common RNN neuron, is efficient in various applications involving long-term dependency (Gers et al., 2000, 2002). We aim to design a dynamic system, which has the multiple timescales structure but with more efficient neurons.

Continuous Timescale Recurrent Neural Network

CTRNN, which is also an extension of RNN, is a kind of artificial neural network described by Hopfield, Tank, and Beer (Hopfield and Tank, 1986; Beer, 1995). With a plausible biological interpretation and inexpensive computational complexity, CTRNN has always been used to explain biological phenomena (Kier et al., 2006). The structure of CTRNN is shown in Figure 1.

CTRNNs were developed by Beer (1995). The basic hypothesis is:

\[ \tau_m \frac{dy_m}{dt} = -y'_m + \sum_{n}^{N} w_{mn}\theta (y'_n - b_n) + I(t) \]

(1)

where \( \tau_m \) is the membrane time constants of the neuron \( m \); \( y'_m \) is the membrane potential after the deletion of the
action potential; \( b_n \) is the bias of the neuron \( n \) \((n \in \mathbb{N})\); \( I(t) \) is the additional input in time \( t \); \( \theta(.) \) is the activation function which could be logistic sigmoid, softmax or hyperbolic tangent.

Equation (1) was derived based on the RC circuit neural model (Dwyer et al., 2010). Thus, CTRNN has a clear interpretation rule from the biological neurons to the artificial neurons of the engineering model. For this very reason, CTRNNs have been used to explain biological phenomenon.

Similar to RNN, the forward process of CTRNN can be concluded as:

\[
    u^t_m = \frac{1}{\tau_m} u^{t-1}_m + \frac{1}{\tau_m} \left( \sum_t^I w^t_{im} x^t_i + \sum_h^H w^t_{hm} y^t_h \right) \tag{2}
\]

where \( \tau_m \) is the time constant of the neuron \( m \); \( u^t_m \) is the presynaptic value of the \( m^{th} \) neuron in the \( t^{th} \) step and \( x \) is the net inputs of the neuron \( m \); \( w_{hm} \) is the weight between the \( h^{th} \) neuron to the \( m^{th} \) neuron; \( I \) represents the direct inputs of neuron \( m \) and \( H \) denotes all other hidden neurons with have weight connections to \( m \). After the presynaptic values are obtained, the activation output can be calculated with suitable activation function. The importance of \( \tau_m \) is to produce a resistance to reject the input from other neurons and try to keep the history information in the neuron. Larger \( \tau_m \) means stronger resistance and a slower activation process. In other words, a neuron with large time constant attempts to store the history information and needs a longer time to accept new inputs.

Back Propagation Through Time (BPTT) can also be used to update the weights of CTRNN as:

\[
    \frac{\partial E}{\partial u^t_m} = \frac{\partial'}{\partial u^t_m} \left( \sum_o^O w^t_{mo} \frac{\partial E}{\partial u^t_o} + \frac{1}{\tau_m} \sum_h^H w^t_{mh} \frac{\partial E}{\partial u^t_h} \right)
     + \left( 1 - \frac{1}{\tau_m} \right) \frac{\partial E}{\partial u^{t+1}_m} \tag{3}
\]

where \( \tau_h \) is the time constant of the neuron \( h \); \( O \) denotes the output neurons; \( \frac{\partial E}{\partial u^t_m} \) represents the error gradient of the neuron \( u^t_m \). Please note that \( \tau_m \) and \( \tau_h \) can be different. With the derivative and the synaptic outputs, weights between two neurons can be obtained using Equation (4).

\[
    \frac{\partial E}{\partial w^t_{nn}} = \sum_t^T \frac{\partial E}{\partial u^t_m} \frac{\partial u^t_n}{\partial w^t_{mn}} = \sum_t^T \frac{\partial E}{\partial u^t_n} y^t_n \tag{4}
\]

**Long-Short Term Memory**

LSTM was created by Hochreiter and Schmidhuber (1997). Unlike the previous RNN models (mentioned in Section Continuous Timescale Recurrent Neural Network) that focus on biological interpretation, LSTM was developed as an engineering model to solve the vanishing gradient problem (Hochreiter et al., 2001).

The structure of LSTM is shown in Figure 2. In order to solve the vanishing gradient problem, the first model of LSTM defines two kinds of gates: input and output gates. Input gate is used to control whether the cell should accept the input information or not. The output gate decides whether the cell should output the contents stored in the cell. Gers et al. improved this prototype and added a forget gate to the model in 2000 (Gers et al., 2000). The forget gate provides a way to reset the contents of cells. LSTM was further improved by Gers et al. (2002). They added the peephole connections to make it possible for the cells to control the time for gate opening inside the block.

The LSTM cells are key in handling the vanishing gradient problem. LSTM can control the information flow and time and can retain the important information by making the information flow unchanged all along the time steps. LSTM has the ability to add or remove information via the three gates and each gate learns to do so through backpropagation.

**The Proposed Continuous Timescale Long-Short Term Memory (CTLSTM) Model**

As shown in Figure 1, CTRNN neuron is still very similar to a traditional RNN neuron. The difference between a CTRNN neuron and an RNN neuron is that the CTRNN neuron considers a time scale delay after calculating the network input. However, LSTM uses a considerably different structure called block and cells instead of the traditional RNN neurons. An LSTM block includes three different gates and several cells (Only one cell is shown in Figure 2). Each cell has an input and an output. But the same gates control cells in one block. The inputs of gates are similar with net inputs. Both direct inputs and neuron (block) outputs from other hidden layer could be the gate inputs or net inputs. Although these two structures are quite different from...
The forward process of the input gate is shown in Equations (5–13).

\[ u_t^i = \sum_i w_{im} x_t^i + \sum_h w_{ih} y_{t-1}^h + \sum_c w_{ic} s_{t-1}^c \]  

\[ y_t^i = f(u_t^i) \]  

\[ u_t^f = \sum_i w_{if} x_t^i + \sum_h w_{ih} y_{t-1}^h + \sum_c w_{ic} s_{t-1}^c \]  

\[ y_t^f = f(u_t^f) \]  

\[ u_t^o = \sum_i w_{io} x_t^i + \sum_h w_{ih} y_{t-1}^h + \sum_c w_{ic} s_{t-1}^c \]  

\[ y_t^o = \frac{1}{\tau} y_{t-1}^c \left( s_t^c \right) + (1 - \frac{1}{\tau}) y_{t-1}^c \]  

The activation process of the input gate is shown in Equations (5) and (6). \( i, h, \) and \( c \) denote the input, hidden and cell state, respectively. Similarly, forget gate is represented in Equations (7) and (8). Cell input is obtained in Equation (9) and the cell state is calculated using Equation (10). Similar to input and forget gate, the output gate activation function is represented in Equations (11) and (12). States at time \( t \) step \( s_t^c \) are used for the input of the output gate in time \( t \), while the state in \( t - 1 \) step \( s_{t-1}^c \) is used for calculating the input and forget gate values in time \( t \). Finally, the cell outputs are calculated using Equation (13) where we added a time constant \( \tau \) for each cell. Larger \( \tau \) means slower cell outputs, and can make the cell focus on the slow features of the dynamic input signal. The traditional LSTM block would be a special case of CTLSTM when \( \tau = 1 \). \( f(.) \) is the activation function of the gates while \( g(.) \) and \( h(.) \) are the activation function of the cell input and output, respectively. We followed Graves and Schmidhuber (2005) and define \( f(.) \) as logistic sigmoid function while \( g(.) \) and \( h(.) \) are hyperbolic tangent functions.

According to the forward process (Equations 5–13) and the BPTT rules, the backward pass (Equations 14–18) can be derived as:

\[ \frac{\partial E}{\partial y_t^c} = \left( 1 - \frac{1}{\tau} \right) \frac{\partial E}{\partial y_{t+1}^c} + \sum_k \sum_h w_{ck} \frac{\partial E}{\partial u_k^c} \]  

\[ \sum_h w_{hc} \frac{\partial E}{\partial u_c^c} \]  

\[ \frac{\partial E}{\partial u_t^i} = \frac{1}{\tau} f'(u_t^i) \sum_i h(s_t^c) \frac{\partial E}{\partial y_t^c} \]  

\[ \frac{\partial E}{\partial s_t^c} = \frac{1}{\tau} y_{t-1}^c h'(s_t^c) \frac{\partial E}{\partial u_c^c} + y_{t-1}^c \frac{\partial E}{\partial s_{t-1}^c} + w_{ic} \frac{\partial E}{\partial u_t^i} \]
Equation (16) and cell input error is shown in Equation (17).

\[ \frac{\partial E}{\partial u_k^t} = f'(u_k^t) \sum_{c} g(u_c^t) \frac{\partial E}{\partial y_c^t} \]  

(17)

\[ \frac{\partial E}{\partial u_c^t} = f'(u_c^t) \sum_{c} g(u_c^t) \frac{\partial E}{\partial y_c^t} \]  

(18)

\[ \frac{\partial E}{\partial u_f^t} = f'(u_f^t) \sum_{c} g(u_c^t) \frac{\partial E}{\partial y_f^t} \]  

(19)

where the derivative of cell outputs are calculated in Equation (14), \( \sum_k w_ch_k \frac{\partial E}{\partial y_k} \) is the error term from the output layer and \( \sum_h w_{ch} \frac{\partial E}{\partial y_h} \) denotes the error come from other hidden layers.

\( H \) can be cell, gate or the neurons of the RNN. Equations (15), (18), and (19) represent the error term of output gate, forget gate, and input gate, respectively. The cell state error is calculated in Equation (16) and cell input error is shown in Equation (17).

Figure 4 shows an application example of the proposed CTLSTM network. We use two CTLSTM layers to build a CTLSTM model. Similar to Supervised MTRNN (Yu and Lee, 2015a), CTLSTM also has slow and fast context layers and can work for both classification and prediction tasks simultaneously. We believe that the fast CTLSTM layer can focus on the fast fractional work while slow CTLSTM can work for slow organizing tasks. This property will help the CTLSTM model to capture the dynamic context from the longer sequences efficiently.

**EXPERIMENTS AND RESULTS**

In order to evaluate our model, we conducted several experiments using multiple datasets including human motion and intention recognition. The mean results are reported with ± s.d. for the performance over 10 runs for each task. We also report the Wilcoxon signed-rank statistical test results to find the significance of the performance of CTLSTM over the existing model in each task. The details of each experiment and the results are illustrated in this section.

**UCI Character Trajectories Dataset**

We used the character trajectories dataset which is a part of the UCI dataset (Williams et al., 2006). It has a total of 2,858 samples and 20 kinds of character trajectories. The data consist of three dimensions which is x, y, and the pen tip force. This dataset consists of only one stroke characters with a single “PEN-DOWN” segment since the character segmentation was performed using a pen tip force cut-off point. For example, characters like “i” or “f” were not included in the dataset. The details of the 20 kinds of characters are shown in Table 1 and Figure 5. 1,433 randomly selected samples are used for training and the remaining 1,425 samples are used for testing. We train the CTLSTM and LSTM models for 500 epochs. This stopping point was chosen since the error does not decrease after an additional training of 50 epochs. The neuron cell states are initialized as set as 0 in all experiments.

The learning rate for both the baseline model using LSTM and the proposed model using CTLSTM was set to 0.00001. 130 blocks (100 + 30 as two layers) were used in the LSTM model. 100 fast CTLSTM blocks \((r = 1)\) and 30 slow CTLSTM blocks were used to build the CTLSTM model. Each block of CTLSTM or LSTM contain only one cell. For a fair comparison, we chose the same network structure \((100 + 30 \text{ as two layers})\) as described in Figure 4 for both CTLSTM and LSTM. Max pooling was used for classification decision in both models. Both offline and online classification results are shown in Table 2. The definition of offline and online accuracy are given below:

\[ P(x) = \frac{N_x}{N_{total}} \]  

(20)

\[ Acc_{offline} = \begin{cases} 
1 \text{ if } \arg\max_x P(x) = C \\
0 \text{ else }
\end{cases} \]  

(21)

**TABLE 1 |** UCI dataset description.

| Classes | Training set | Test set | Total |
|---------|--------------|----------|-------|
| a       | 97           | 74       | 171   |
| b       | 73           | 68       | 141   |
| c       | 55           | 87       | 142   |
| d       | 82           | 75       | 157   |
| e       | 113          | 73       | 186   |
| g       | 66           | 72       | 138   |
| h       | 57           | 70       | 127   |
| i       | 80           | 94       | 174   |
| m       | 69           | 56       | 125   |
| n       | 56           | 74       | 130   |
| o       | 68           | 73       | 141   |
| p       | 70           | 61       | 131   |
| q       | 70           | 54       | 124   |
| r       | 57           | 62       | 119   |
| s       | 64           | 69       | 133   |
| u       | 67           | 64       | 131   |
| v       | 74           | 81       | 155   |
| w       | 60           | 65       | 125   |
| y       | 67           | 70       | 137   |
| z       | 88           | 83       | 171   |
| Total   | 1,433        | 1,425    | 2,858 |
FIGURE 5 | Twenty samples of characters taken from the mixed dataset. Total size of dataset is 2,858 characters, with over 100 samples of each character.

TABLE 2 | Classification results for UCI dataset.

| Classes | Max-pooling (Offline) | Real-time (Online) |
|---------|-----------------------|--------------------|
|         | CTLSTM                | LSTM               |
|         |                       | CTLSTM             | LSTM               |
| a       | 75.68 ± 15.02         | 45.14 ± 19.35      | 45.59 ± 5.49       | 35.14 ± 9.73 |
| b       | 87.65 ± 13.59         | 80 ± 9.54          | 72.94 ± 5.73       | 64.22 ± 8.28 |
| c       | 87.36 ± 7.69          | 68.85 ± 10.67      | 53.88 ± 5.01       | 42.58 ± 6.63 |
| d       | 98.8 ± 2.1            | 97.6 ± 3.31        | 76.68 ± 4.09       | 74.7 ± 2.68 |
| e       | 97.26 ± 0.61          | 96.44 ± 4.21       | 90.22 ± 1.49       | 89.51 ± 3.97 |
| g       | 97.92 ± 1.55          | 94.17 ± 4.68       | 78.4 ± 3.28        | 72.07 ± 6.8 |
| h       | 83 ± 8.89             | 75.86 ± 9.46       | 66.87 ± 7.75       | 62.02 ± 8.15 |
| l       | 92.34 ± 14.55         | 60.96 ± 16.8       | 71.49 ± 5.92       | 54.1 ± 14.02 |
| m       | 74.82 ± 8.54          | 79.29 ± 11.01      | 67.81 ± 7.1        | 68.63 ± 9.19 |
| n       | 53.78 ± 10.02         | 42.43 ± 12.27      | 43.57 ± 6.2         | 32.48 ± 8.54 |
| o       | 76.99 ± 10.1          | 73.97 ± 11.15      | 54.79 ± 7.36       | 57.16 ± 8.32 |
| p       | 94.59 ± 3.03          | 93.44 ± 5.96       | 77.69 ± 2.37       | 73.7 ± 6.48 |
| q       | 92.22 ± 3.78          | 81.11 ± 10.37      | 70.02 ± 5.3        | 65.43 ± 9.08 |
| r       | 60 ± 12.38            | 70.48 ± 18.18      | 55.64 ± 7.51       | 62.28 ± 15.27 |
| s       | 96.52 ± 5.55          | 96.67 ± 2.25       | 79.49 ± 5.55       | 78.62 ± 2.99 |
| u       | 79.06 ± 10.69         | 80.47 ± 8.16       | 60.79 ± 7.99       | 59.26 ± 6.47 |
| v       | 98.64 ± 2.94          | 98.77 ± 2.34       | 81.24 ± 2.76       | 81.96 ± 2.63 |
| w       | 73.54 ± 9.23          | 85.85 ± 6.8        | 62.21 ± 7.23       | 70.77 ± 5.76 |
| y       | 93.86 ± 3.62          | 96.14 ± 3          | 71.95 ± 4.59       | 75.06 ± 3.57 |
| z       | 100 ± 0               | 100 ± 0            | 98.33 ± 2.42       | 95.17 ± 1.6 |
| Avg.    | **85.7 ± 1.53**       | **80.88 ± 3.74**   | **68.93 ± 0.98**   | **65.74 ± 2.54** |
| Weighted Avg. | **86.2 ± 1.72**       | **80.57 ± 3.75**   | **69.12 ± 1.02**   | **65.5 ± 2.49** |

Numbers in bold show the best performance in the particular task.

\[
\text{Acc}^c_{\text{online}} = P(c) = \frac{N_c}{N_{\text{total}}} \\
\text{Acc}_{\text{weighted avg}} = \frac{\sum_{m} M N_m \text{Acc}^m_{\text{online}}}{\sum_{m} M N_m} 
\]

where \( N_c \) is the frame number which is classified as class \( x \) and \( N_{\text{total}} \) is the total frame number of the current sequence. In simple terms, there are only two cases of offline classification on one sample: 100 or 0%. But online classification requires a real-time per frame accuracy. Since, this dataset is unbalanced, we performed a weighted average according to Equation (27), where \( N_m \) is the frame number of class \( m \), \( M \) is the species number and \( \text{Acc}^m_{\text{online}} \) is the online accuracy of class \( m \).

Theoretically, the timescale should be similar to the length of a dynamic feature. From Figure 7 we can find that a dynamic feature (e.g., from peak to valley) ranges from 10 to 30 frames, thus we chose 20 as the slow context timescale for this task.

With the help of the slow CTLSTM blocks, CTLSTM has better performance than traditional LSTM on multiple character trajectories classification on both cases. Figure 6 shows the real-time classification outputs. The red lines denote the output neuron activation corresponding to the correct class, and the blue lines mean the output neuron activation corresponding to the other classes. The activation of the correct class of CTLSTM goes up and never falls down in Figure 6. The neuron activation illustrated using pixels is shown in Figure 7. Brighter pixels express higher activation value, while the darker ones express...
lower value. X axis denotes the time axis while the Y axis is
the activation of the corresponding neuron number. The neuron
activation of the traditional LSTM layer is shown in the top part
of Figure 7. Out of the 130 neurons in total, the 30 neurons are
in the slow CTLSTM layer. The activation of the slow and fast
CTLSM layer neurons can be seen in the middle part of Figure 7.

As illustrated in Figure 7, we can distinguish the neuron activities
of LSTM and CTLSTM. We can see that the neuron activity of
LSTM is uniform for all the neurons. Whereas, the slower
CTLSTM neurons can be seen to start its activations with a
delay since the timescale is larger for those slow neurons. On
the other hand the faster CTLSTM behaves similar to the LSTM
since the timescale is 1 for these neurons as the case of LSTM.

In the case of LSTM, the activations of all the neurons fires
frequently, similar to the fast CTLSTM cell. However, in CTLSTM,
we can easily distinguish that the cells in the slow CTLSTM
layer have slower activation than the ones of fast CTLSTM
layer. This feature helps the model to become more stable in
the real-time classification task. It would be more easily for a
slow CTLSTM block to capture and hold an important dynamic
feature than a fast CTLSTM (LSTM) block. The classification
accuracy, the error curve of classification and prediction are
shown in Figures 8–10, respectively. The structure illustrated
in Figure 4 is also implemented for LSTM in order to conduct
a fair comparison. We implemented a two layer LSTM with
100 + 30 LSTM blocks and compare it to CTLSTM. Similar
to the classification error decreasing curve shown in Figure 9,
the prediction error of CTLSTM decreases faster than LSTM
as shown in Figure 10. The classification performance is shown
in Table 2. The experiment results show that prediction helps
both CTLSTM and LSTM to converge faster, and CTLSTM
outperforms LSTM in both prediction as well as classification.

The confusion matrix of CTLSTM with prediction is shown in
Figure 11. It can be seen that both algorithms have some
difficulty in classifying similar pairs such as “n” and “h”, “a” and
“e”, “q” and “g,” etc.

Furthermore, Table 3 shows the Wilcoxon signed-rank test
to check the difference in performance between CTLSTM and
LSTM. As illustrated in the results, CTLSTM outperforms LSTM
significantly.

Microsoft Research Cambridge-12 Kinect Gesture Dataset

We use the Microsoft Research Cambridge-12 (Fothergill et al.,
2012) dataset in this experiment. The dataset consists of
sequences of human movements, represented as body-part
locations, and the associated gesture to be recognized by the
system. It included 594 sequences and 719,359 frames—~6 h
and 40 min—collected from 30 people performing 12 gestures.
In total, there are 6,244 gesture instances. Each sequence lasts
about 900–3,000 frames. We use 100 fast with 30 slow blocks

![Figure 6](image_url)

**Figure 6**: An example of real-time classification outputs of the two models (character m).

![Figure 7](image_url)

**Figure 7**: An example of real-time neuron activation of the two models (character m).
for CTLSTM structure. LSTM with 130 blocks is used for comparison. The average length of MRC12 dataset is about 1,000 frames. Thus, the time constant we used for slow blocks is set to 40. The motion files contain tracks of 20 joints estimated using the Kinect Pose Estimation pipeline. X, Y coordinates and the depth distances are recorded in the dataset. The body poses are captured at a sample rate of 30 Hz with an accuracy of about 2 cm in joint positions.

Similar to our previous experiment, we randomly select half of the dataset to be used for training and the other half is used for testing. The stopping point was chosen when the error did not decrease after 30 epochs. The dataset is normalized by using min-max normalization. The details of the 12 kinds of motions, and the sequences used for train and test are shown in Table 4.

We set the learning rate for both LSTM and CTLSTM to 0.00001. In both models, each block included one cell. Max pooling was used for classification decision in both models. The classification performance is shown in Table 5. CTLSTM shows better performance than LSTM. We also report the Wilcoxon signed-rank test in Table 6 and the results indicate the significance of the performance of CTLSTM over LSTM.
and MTRNN. The accuracy and the error curve of the training are shown in Figures 12, 13, respectively. Note that CTLSTM converges much faster and is more stable than LSTM. With the help of slow blocks, CTLSTM manages to outperform LSTM. We also compare our model to another neural recurrent model called Supervised MTRNN (Yu and Lee, 2015b). The neuron number, network structure and timescales used in Supervised MTRNN are the same as CTLSTM. We omit the result of Supervised MTRNN in the first experiment with UCI dataset because Supervised MTRNN did not converge over 1,000 epochs.

We also test the prediction performance of LSTM, CTLSTM and Supervised MTRNN in Figure 14 for the Microsoft Research Cambridge-12 dataset. Similar as the results shown in Figure 10, prediction error of CTLSTM decreases faster than LSTM.

### TABLE 4 | Dataset description of Microsoft Research Cambridge-12 (MRC12) dataset.

| Classes                  | Training set | Test set | Total |
|--------------------------|--------------|----------|-------|
| Iconic gestures          |              |          |       |
| Crouch or hide (G2)      | 25           | 25       | 50    |
| Put on night vision goggles (G4) | 25           | 25       | 50    |
| Shoot a pistol (G6)     | 25           | 24       | 49    |
| Throw an object (G8)    | 25           | 24       | 49    |
| Change weapon (G10)     | 24           | 24       | 48    |
| Kick (G12)               | 25           | 24       | 49    |
| Metaphoric gestures     |              |          |       |
| Start music/raise volume (of music) (G1) | 25           | 25       | 50    |
| Navigate to next menu (G3) | 25           | 25       | 50    |
| Wind up the music (G5)  | 24           | 24       | 48    |
| Take a bow to end music session (G7) | 25           | 25       | 50    |
| Protest the music (G9)  | 25           | 25       | 50    |
| Move up the tempo of the song (G11) | 25           | 24       | 49    |

### TABLE 5 | Online classification results for Microsoft Research Cambridge-12 (MRC12) dataset.

| Classes                  | CTLSTM | LSTM   | MTRNN  |
|--------------------------|--------|--------|--------|
| Iconic gestures          |        |        |        |
| Crouch or hide (G2)      | 73.78 ± 5.69 | 69.04 ± 7.68 | 31.88 ± 19.07 |
| Put on night vision goggles (G4) | 86.69 ± 3.35 | 80.35 ± 6.49 | 65.68 ± 24.68 |
| Shoot a pistol (G6)     | 87 ± 4.2 | 79.01 ± 3.83 | 54.54 ± 25.89 |
| Throw an object (G8)    | 84.69 ± 4.61 | 79.17 ± 6.68 | 39.1 ± 25.88 |
| Change weapon (G10)     | 78.16 ± 4.54 | 74.13 ± 3.84 | 24.54 ± 11.45 |
| Kick (G12)               | 85.59 ± 4.81 | 86.77 ± 6.65 | 56.28 ± 17.74 |
| Metaphoric gestures     |        |        |        |
| Start music/raise volume (of music) (G1) | 82.48 ± 5.34 | 77.17 ± 5.28 | 49.43 ± 29.48 |
| Navigate to next menu (G3) | 89.38 ± 1.52 | 91.36 ± 0.97 | 48.09 ± 18.43 |
| Wind up the music (G5)  | 73.03 ± 4.47 | 72.66 ± 6.69 | 35.3 ± 23.06 |
| Take a bow to end music session (G7) | 69.12 ± 5.97 | 76.62 ± 3.03 | 55.63 ± 17.8 |
| Protest the music (G9)  | 62 ± 6.65 | 61.28 ± 8.6 | 37.55 ± 14.12 |
| Move up the tempo of the song (G11) | 86.95 ± 1.65 | 86.57 ± 1.96 | 37.95 ± 12.45 |
| Average                  | 79.92 ± 1.45 | 77.84 ± 1.71 | 44.66 ± 8.22 |

Number in bold shows the best performance.
Human Action Dataset

This experiment is conducted on a custom dataset collected by us. It consists of long sequences of human movements, represented as body-part locations, and the associated gesture to be recognized by the system. The dataset includes 200 sequences and 200,000 frames—~2 h—collected using 10 volunteers performing 10 actions. Each sequence lasts about 1,000 frames. The details of this dataset and the sequences used for training and testing are listed in Table 7. The data contain tracks of 25 joints estimated collected using Kinect v2. X, Y coordinates and
the depth distance are recorded in the dataset. The body poses are captured at a sample rate of 30 Hz with an accuracy of about

| Classes    | CTLSTM (±) | LSTM (±) | MTRNN (±) |
|------------|------------|----------|-----------|
| Standing   | 90.64 ± 2.78 | 91.31 ± 3.98 | 90.33 ± 5.16 |
| Eating noodles | 81.58 ± 2.76 | 66.55 ± 3.7 | 71.8 ± 6.46 |
| Drinking   | 79.63 ± 3.16 | 78.38 ± 4.45 | 73.89 ± 6.04 |
| Clapping   | 65.32 ± 2.91 | 79.01 ± 5.26 | 96.05 ± 3.75 |
| Raising hand | 96.53 ± 2.4 | 85.83 ± 5.28 | 90.33 ± 4.07 |
| Pointing   | 87.76 ± 2.82 | 77.67 ± 7.06 | 67.77 ± 6.15 |
| Bowing     | 84.11 ± 2.97 | 86.87 ± 4.77 | 86.13 ± 4.99 |
| Crouching  | 96.04 ± 2.58 | 94.07 ± 3.58 | 89.67 ± 5.27 |
| Punching   | 75.53 ± 3.2  | 50.78 ± 7.27 | 35.38 ± 5.82 |
| Kicking    | 85.9 ± 2.81  | 63.65 ± 7.78 | 38.4 ± 5.9  |
| Average    | 84.3 ± 1.43  | 77.41 ± 1.91 | 73.86 ± 1.69 |

Number in bold shows the best performance.

| Method 1 | Method 2 | Z     | **p**  |
|----------|----------|-------|--------|
| LSTM     | CTLSTM   | −2.803| 0.005  |
| MTRNN    | CTLSTM   | −2.803| 0.005  |
| MTRNN    | LSTM     | −2.803| 0.005  |

The classification performance of CTLSTM can be seen to have a significant improvement in the comparison (**p < 0.01**).
### Table 10 | Dataset description for five kinds of intentions.

| Intention classes     | Actions included      | Training set | Test set | Total |
|-----------------------|-----------------------|--------------|----------|-------|
| Having a meal         | Eating + drinking     | 10           | 10       | 20    |
| Fitness exercise      | Standing + crouching  | 10           | 10       | 20    |
| Appreciate            | Clapping + bowing     | 10           | 10       | 20    |
| Seeking attention     | Raising hand + pointing | 10           | 10       | 20    |
| Aggression            | Kicking + punching    | 10           | 10       | 20    |

### Table 11 | Comparison between human action and intention understanding datasets.

| Dataset property                          | Human action dataset | Intention understanding dataset |
|-------------------------------------------|----------------------|---------------------------------|
| Sequence length in each class             | 1,000                | 2,000                           |
| Frames per second                         | 30                   | 30                              |
| Sequence duration of each class in seconds| 33.33                | 66.66                           |

MTRNN with the same block number and learning rate. As shown in Table 6, CTLSTM can be seen to perform well even in the case of longer sequences and outperforms the Supervised MTRNN and LSTM baseline models.

### Intention Understanding Dataset

We also perform the experiment using an intention understanding dataset collected by us. This dataset is more challenging and requires more capability to handle longer sequences because unlike the human action dataset described in the previous experiment, which consists of a single action in each class, this dataset consists of long sequences of multiple human actions in each intention class. The conditions of collecting the data are the same as the previous experiment. The data are collected using 10 volunteers performing two kinds of actions each for five different intention classes. The dataset includes 100 sequences and 200,000 frames. Each sequence lasts for about 2,000 frames. The sequences used for training and test are listed in Table 10. We compare the sequence length of the human action dataset and the intention understanding dataset in Table 11. As we can observe from the tables, in the intention understanding dataset, the sequence length of each class as well as the time duration of each class is double the human action dataset. The longer sequence data need a model which can capture the context well for longer period of time for intention understanding.

We use 150 fast and 50 slow blocks for our CTLSTM model with a time constant of 40 for the slow blocks. In order to compare our model with the baseline, we train a two layer LSTM model with (150 + 50) blocks with one cell each. We set the learning rate for both LSTM and CTLSTM to 0.00001. The classification decision in both the models use max pooling. We omit the comparison results of Supervised MTRNN in this experiment because Supervised MTRNN did not converge even after 1,000 epochs of training. The classification results of the two models are shown in Table 12 and the classification accuracy curve is shown in Figure 17. The error decreasing curve of the training is shown in Figure 18. Table 13 shows the results of Wilcoxon signed-rank test to compare the performance between CTLSTM and LSTM. As the results illustrate, the performance of CTLSTM is consistent with the previous experiments as it converges faster compared to the LSTM model. The results also show that with the help of the timescales in the CTLSTM model to capture the dynamic context from the longer sequences efficiently, it is able to outperform the existing models, thereby making it the most suitable model for intention understanding tasks.

### Conclusion and Discussion

We developed the Continuous Timescale LSTM (CTLSTM) model inspired by the CTRNN and LSTM. Our proposed CTLSTM model benefits from the multiple timescales and is equipped to assign different work on different layers. CTLSTM is proved to have better performance on multiple classification.
tasks. We have shown the effectiveness of our proposed model in longer sequence understanding tasks and we argue that our model will be suitable for human behavior and intention understanding using deep learning techniques.

It should be noted that our model is quite different from the hierarchical multiscale RNN (Chung et al., 2017) and Multi-Timescale Long Short-Term Memory Neural Network (Liu et al., 2015). In their work, they define slow LSTM layer as a normal LSTM layer but with just a slower input rate. That is, not every frame is used as the input to the slow LSTM layer. In this case, the input of the slow LSTM layer is very important and the fast LSTM cells should capture the useful information and make sure the key information is transferred to the slow LSTM layer as inputs. Unlike their work, we define slow LSTM layer using an additional CTRNN activation where each frame is as the input of the slow LSTM layer.

The multiple timescales structure gives CTLSTM more capability to hold the information by improving the organization of the architecture and focusing on different task at different levels. LSTM has a special cell/block structure, which is able to hold important information for a long time if the forget gate is always closed. However, the gate opening time is still determined by BPTT. But BPTT itself may not have much ability to decide the temporal scale of the blocks in order to focus on different contexts. Thus CTLSTM, with the ability to guide the fast and slow blocks for different contexts is able to handle longer sequences efficiently compared to LSTM models.

The capacity of CTLSTM is highly correlated with the timescale constants. Theoretically, we can make one block (a very slow block) fire for longer period by assigning a large time constant (for example, 1,000). But a block with a large time constant would be difficult for training. Due to the restriction of Equation (23) large time constant would decrease the $\frac{\partial E}{\partial u_i}$. If $\frac{\partial E}{\partial u_i} \approx 0$, then $\frac{\partial E}{\partial w_{ij}}$, $\frac{\partial E}{\partial a_{ij}}$, and $\frac{\partial E}{\partial u_j}$ would also approach 0. So the weights would not be updated due to:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_i}$$ (24)

This brings difficulty for slow neurons to get features from the input or other neurons. In our experiments, the timescale constant of the CTLSTM is set to 20 or 40 based on the dataset. In the future, we aim to solve this difficulty of setting the timescale constants by developing an adaptive mechanism for the timescales during the training process.

Fortunately, memory of RNN, which aims to further enhance the memory ability of RNN, already have been on focus in recent times (Sukhbaatar et al., 2015; Graves et al., 2017). We wish to make use of the advantages of timescales on the memory of RNN in our future work.

**AUTHOR CONTRIBUTIONS**

All authors contributed equally and extensively to the work presented in this paper. ZY and ML designed the model and experiments. DM collected, analyzed, and interpreted the data. ZY developed the code and performed the experiments. ZY wrote the manuscript draft. DM and ML revised the paper.

**FUNDING**

This research was supported by the Industrial Strategic Technology Development Program (10044009) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea) (50%) and by the ICT R&D program of MSIP/IITP (R7124-16-0004, Development of Intelligent Interaction Technology Based on Context Awareness and Human Intention Understanding) (50%).

**REFERENCES**

Alnajjar, F., Yamashita, Y., and Tani, J. (2013). “Formulating a cognitive branching task by MTRNN: a robotic neuroscience experiments to simulate the PFC and its neighboring regions,” in Advances in Cognitive Neurodynamics (III), eds Y. Yamaguchi (Dordrecht: Springer Netherlands), 267–273.

Arie, H., Arakaki, T., Sugano, S., and Tani, J. (2012). Imitating others by composition of primitive actions: a neuro-dynamic model. Rob. Auton. Syst. 60, 729–741. doi: 10.1016/j.robot.2011.11.005

Arie, H., Endo, T., Arakaki, T., Sugano, S., and Tani, J. (2009). Creating novel goal-directed actions at criticality: a neuro-robotic experiment. New Math. Natural Comput. 5, 307–334. doi: 10.1142/S1793005709001283
Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2011). "Sequential deep learning for human action recognition," in International Workshop on Human Behavior Understanding (Berlin; Heidelberg: Springer), 29–39.

Beer, R. D. (1995). On the dynamics of small continuous-time recurrent neural networks. Adapt. Behav. 3, 469–509. doi: 10.1177/10597123950300405

Bütepage, J., Black, M. L., Kragic, D., and Kjellström, H. (2017a). "Deep representation learning for human motion prediction and classification," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

Bütepage, J., Kjellström, H., and Kragic, D. (2017b). Anticipating many futures: Online human motion prediction and synthesis for human-robot collaboration, arXiv:1702.08212.

Chung, J., Ahn, S., and Bengio, Y. (2017). "Hierarchical multiscale recurrent neural network," in ICLR 2017 (Toulon).

Cruse, H. (2006). Neural networks as cybernetic systems - part II. Brains Minds Media 2:mmn290. (urn:nbn:de:0010-2009-3-2906).

Dwyer, J., Lee, H., Martell, A., Stevens, R., Hereld, M., and van Drongelen, W. (2010). Oscillation in a network model of neocortex. Neurocomputing 75, 1051–1056. doi: 10.1016/j.neucom.2009.12.021

Fothergill, S., Mentis, H., Kohli, P., and Nowozin, S. (2012). "Instructing people for training gestural interactive systems," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY: ACM), 1737–1746.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: continual prediction with LSTM. Neural Comput. 12, 2451–2471. doi: 10.1162/089976600300015015

Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. J. Machine Learning Res. 3, 115–143. doi: 10.1162/15324430376866139

Ghassayes, M. (Ed.) (1999). Text and Context in Functional Linguistics, Vol. 169. John Benjamins Publishing.

Givón, T. (2005). Context as Other Minds: The Pragmatics of Sociality, Cognition and Communication. John Benjamins Publishing.

Graves, A., and Schmidhuber, J. (2005). Framework phoneme segmentation with bidirectional LSTM and other neural network architectures. Neural Netw. 18, 602–610. doi: 10.1016/j.neunet.2005.06.042

Graves, A., Mohamed, A. R., and Hinton, G. (2013). "Speech recognition with deep recurrent neural networks," in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (IEEE) (Vancouver, BC), 6645–6649. doi: 10.1109/ICASSP.2013.6638947

Graves, E., Joulion, A., and Usunier, N. (2017). “Improving neural language models with a continuous cache,” in ICLR 2017 (Toulon).

Hinton, G. E., and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science 313, 504–507. doi: 10.1126/science.1127647

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural Comput. 9, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies. Available online at: http://www.bioinf.jku.at/publications/older/ch7.pdf

Hoppfield, J. J., and Tank, D. W. (1986). Computing with neural circuits - a model. Science 233, 625–633. doi: 10.1126/science.3755256

Jaeger, H. (2002). Tutorial on Training Recurrent Neural Networks, Covering BPPT, RTRL, EKF and the "Echo State Network" Approach, Vol. 5. GMD/Forschungszentrum Informationstechnik, Bremen, Germany.

Jeong, S., Arie, H., Lee, M., and Tani, J. (2012). Neuro-robotics study on integrative learning of proactive visual attention and motor behaviors. Cogn. Neurodyn. 6, 43–59. doi: 10.1007/s11571-011-9176-7

Kier, R. J., Ames, J. C., Beer, R. D., and Harrison, R. R. (2006). Design and implementation of multipattern generators in analog VLSI. IEEE Trans. Neural Netw. 17, 1025–1038. doi: 10.1109/TNN.2006.875983

Kim, S., Yu, Z., and Lee, M. (2017). Understanding human intention by connecting perception and action learning in artificial agents. Neural Netw. 92, 29–38. doi: 10.1016/j.neunet.2017.01.009

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2324. doi: 10.1109/5.726791

Liu, P., Qiu, X., Chen, X., Wu, S., and Huang, X. (2015). “Multi-timescale long short-term memory neural network for modelling sentences and documents,” in The Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, (EMNLP) (Lisbon), 2326–2335.

Peniak, M., Marocco, D., Tani, J., Yamashita, Y., Fischer, K., and Cangelosi, A. (2011). “Multiple time scales recurrent neural network for complex action acquisition,” in Front. Comput. Neurosci. Conference Abstract: IEEE ICML-EPISOB 2011 (Frankfurt), doi: 10.3389/conf.fncsm.2011.52.00009

Plappert, M., Mandery, C., and Asfour, T. (2017). Learning a bidirectional mapping between human whole-body motion and natural language using deep recurrent neural networks. arXiv:1705.06400.

Schrodt, F., and Butz, M. (2016). Just imagine! Learning to emulate and infer actions with a stochastic generative architecture. Front. Robot. AI 3:5. doi: 10.3389/frobt.2016.00005

Sukhbaatar, S., Weston, J., and Fergus, R. (2015). “End-to-end memory networks,” in Advances in Neural Information Processing Systems (Montreal), 2440–2448.

Tani, J., Nishimoto, R., and Paine, R. W. (2008). Achieving “organic compositionality” through self-organization: reviews on brain-inspired robotics experiments. Neural Netw. 21, 584–603. doi: 10.1016/j.neunet.2008.03.008

Water, A., and Lincoln, M. (2012). “Semantic machine learning for business process content generation,” in OTM Confederated International Conferences On the Move to Meaningful Internet Systems” (Berlin; Heidelberg: Springer), 74–91.

Williams, B., Toussaint, M., and Storkey, A. (2006). Extracting motion primitives from natural handwriting data. Artif. Neural Netw. ICANN 2006, 634–643. doi: 10.1007/11840930_66

Wonnin, B., Thomas, M. B., Federico, R., and Marcus, L. (2015). “Scene labeling with LSTM recurrent neural networks,” in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Boston, MA), 3547–3555.

Yu, Z., and Lee, M. (2015a). Human motion based intent recognition using a deep dynamic neural model. Robot. Auton. Syst. 71, 134–149. doi: 10.1016/j.robot.2015.01.001

Yu, Z., and Lee, M. (2015b). Real-time human action classification using a dynamic neural model. Neural Netw. 69, 29–43. doi: 10.1016/j.neunet.2015.04.013

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2017 Yu, Moirangthem and Lee. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) or licensor are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.