REVIEW **Communicated by Terrence Sejnowski**

A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures

Yong Yu
yuyongep@163.com
Department of Automation, Xi'an Institute of High-Technology, Xi'an 710025, China, and Institute No. 25, Second Academy of China, Aerospace Science and Industry Corporation, Beijing 100854, China

Xiaosheng Si
sxs09@mails.tsinghua.edu.cn
Changhua Hu
hch_reu@sina.com
Jianxun Zhang
zhang200735@163.com
Department of Automation, Xi'an Institute of High-Technology, Xi'an 710025, China

Recurrent neural networks (RNNs) have been widely adopted in research areas concerned with sequential data, such as text, audio, and video. However, RNNs consisting of sigma cells or tanh cells are unable to learn the relevant information of input data when the input gap is large. By introducing gate functions into the cell structure, the long short-term memory (LSTM) could handle the problem of long-term dependencies well. Since its introduction, almost all the exciting results based on RNNs have been achieved by the LSTM. The LSTM has become the focus of deep learning. We review the LSTM cell and its variants to explore the learning capacity of the LSTM cell. Furthermore, the LSTM networks are divided into two broad categories: LSTM-dominated networks and integrated LSTM networks. In addition, their various applications are discussed. Finally, future research directions are presented for LSTM networks.

1 Introduction

Over the past few years, deep learning techniques have been well developed and widely adopted to extract information from various kinds of data (Ivakhnenko & Lapa, 1965; Ivakhnenko, 1971; Bengio, 2009; Carrio, Sampedro, Rodriguez-Ramos, & Campoy, 2017; Khan & Yairi, 2018). Considering different characteristics of input data, there are several types of architectures for deep learning, such as the recurrent neural network (RNN; Robinson & Fallside, 1987; Werbos, 1988; Williams, 1989; Ranzato et al., 2014),

Neural Computation **31, 1235–1270** (2019)
doi:10.1162/neco_a_01199

convolutional neural network (CNN; Fukushima, 1980; LeCun et al., 1989; Weng, Ahuja, & Huang, 1993; Rawat & Wang, 2017), and deep neural network (DNN; Guo, Liu, Georgiou, & Lew, 2017; Sharma & Singh, 2017). Usually the CNN and DNN cannot deal with the temporal information of input data. Therefore, in research areas that contain sequential data, such as text, audio, and video, RNNs are dominant. Specifically, there are two types of RNNs: discrete-time RNNs and continuous-time RNNs (Pearlmutter, 1989; Brown, Yu, & Garverick, 2004; Gallagher, Boddhu, & Vigraham, 2005). The focus of this review is on discrete-time RNNs.

The typical feature of the RNN architecture is a cyclic connection, which enables the RNN to possess the capacity to update the current state based on past states and current input data. These networks, such as fully RNNs (Elman, 1990; Jordan, 1986; Chen & Soo, 1996) and selective RNNs (Šter, 2013), consisting of standard recurrent cells (e.g., sigma cells), have had incredible success on some problems. Unfortunately, when the gap between the relevant input data is large, the above RNNs are unable to connect the relevant information. In order to handle the "long-term dependencies," Hochreiter and Schmidhuber (1997) proposed long short-term memory (LSTM).

Almost all exciting results based on RNNs have been achieved by LSTM, and thus it has become the focus of deep learning. Because of their powerful learning capacity, LSTMs work tremendously well and have been widely used in various kinds of tasks, including speech recognition (Fernández, Graves, & Schmidhuber, 2007; He & Droppo, 2016; Hsu, Zhang, Lee, & Glass, 2016), acoustic modeling (Sak, Senior, & Beaufays, 2014; Qu, Haghani, Weinstein, & Moreno, 2017), trajectory prediction (Altché & Fortelle, 2017), sentence embedding (Palangi et al., 2015), and correlation analysis (Mallinar & Rosset, 2018). In this review, we explore these LSTM networks. This review is different from other review work on RNNs (Deng, 2013; Lipton, Berkowitz, & Elkan, 2015); it focuses only on advances in the LSTM cell and structures of LSTM networks. Here, the LSTM cell denotes the recurrent unit in LSTM networks.

The rest of this review is organized as follows. Section 2 introduces the LSTM cell and its variants. Section 3 reviews the popular LSTM networks and discusses their applications in various tasks. A summary and discussion of future directions are in section 4.

2 LSTM Cells and Their Variants

In RNNs, the recurrent layers or hidden layers consist of recurrent cells whose states are affected by both past states and current input with feedback connections. The recurrent layers can be organized in various architectures to form different RNNs. Therefore, RNNs are mainly distinguished by the recurrent cell and network architecture. Different cells and inner connections enable RNNs to possess different capacities. In order to explore the

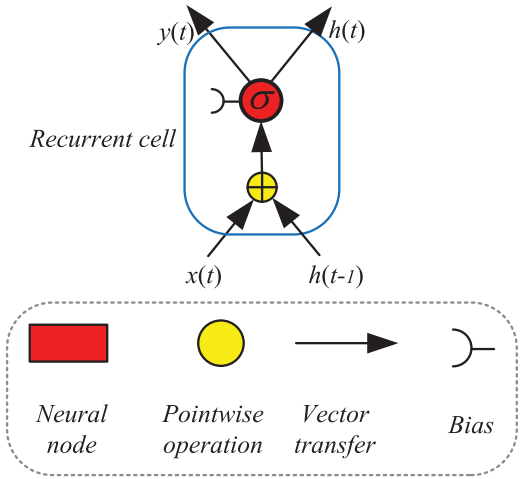


Figure 1: Schematic of the standard recurrent sigma cell.

development of LSTM networks, this section first gives a brief review of the LSTM cell and its variants.

2.1 Standard Recurrent Cell. Usually RNNs are networks that consist of standard recurrent cells such as sigma cells and tanh cells. Figure 1 shows a schematic of the standard recurrent sigma cell. The mathematical expressions of the standard recurrent sigma cell are written as follows:

$$\begin{aligned} h_t &= \sigma(W_h h_{t-1} + W_x x_t + b), \\ y_t &= h_t, \end{aligned} \tag{2.1}$$

where x_t, h_t, and y_t denote the input, the recurrent information, and the output of the cell at time t, respectively; W_h and W_x are the weights; and b is the bias. Standard recurrent cells have achieved some success in some problems (Karpathy, Johnson, & Li, 2015; Li, Li, Cook, Zhu, & Gao, 2018). However, the recurrent networks that consist of standard recurrent cells are not capable of handling long-term dependencies: as the gap between the related inputs grows, it is difficult to learn the connection information. Hochreiter (1991) and Bengio, Simard, and Frasconi (1994) analyzed fundamental reasons for the long-term dependencies problem: error signals flowing backward in time tend to either blow up or vanish.

2.2 LSTM. In order to deal with the problem of "long-term dependencies," Hochreiter and Schmidhuber (1997) proposed the LSTM cell. They improved the remembering capacity of the standard recurrent cell by

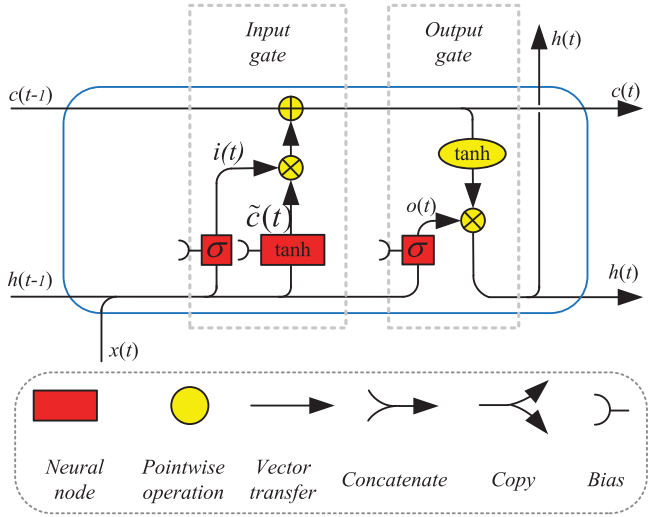


Figure 2: Original LSTM architecture.

introducing a "gate" into the cell. Since this pioneering work, LSTMs have been modified and popularized by many researchers (Gers, 2001; Gers & Schmidhuber, 2000). Variations include LSTM without a forget gate, LSTM with a forget gate, and LSTM with a peephole connection. Usually the term *LSTM cell* denotes LSTM with a forget gate. We first introduce the original LSTM model that possesses only input and output gates.

2.2.1 LSTM without a Forget Gate. The architecture of LSTM with only input and output gates is shown in Figure 2. The mathematical expressions of the LSTM in Figure 2 can be written as follows,

$$\begin{aligned} i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i), \\ \tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}), \\ c_t &= c_{t-1} + i_t \cdot \tilde{c}_t, \\ o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o), \\ h_t &= o_t \cdot \tanh(c_t), \end{aligned} \tag{2.2}$$

where c_t denotes the cell state of LSTM. W_i, $W_{\tilde{c}}$, and W_o are the weights, and the operator '$\cdot$' denotes the pointwise multiplication of two vectors. When updating the cell state, the input gate can decide what new information can be stored in the cell state, and the output gate decides what information can be output based on the cell state.

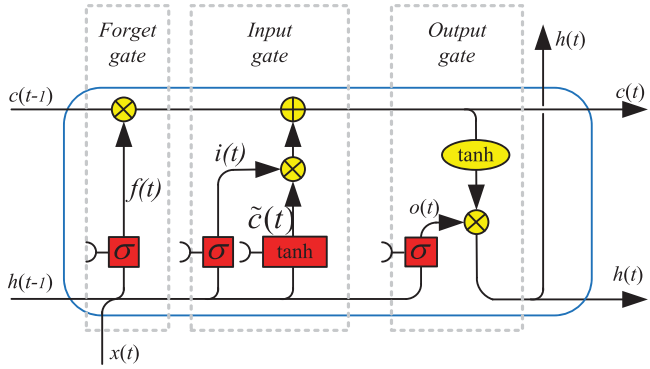


Figure 3: Architecture of LSTM with a forget gate.

2.2.2 LSTM with a Forget Gate. Gers, Schmidhuber, and Cummins (2000) modified the original LSTM in 2000 by introducing a forget gate into the cell. In order to obtain the mathematical expressions of this modified LSTM cell, Figure 3 presents its inner connections.

Based on the connections shown in Figure 3, the LSTM cell can be mathematically expressed as follows:

$$\begin{aligned} f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f), \\ i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i), \\ \tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}), \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \\ o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o), \\ h_t &= o_t \cdot \tanh(c_t). \end{aligned} \tag{2.3}$$

The forget gate can decide what information will be thrown away from the cell state. When the value of the forget gate, f_t, is 1, it keeps this information; meanwhile, a value of 0 means it gets rid of all the information. Jozefowicz, Zaremba, and Sutskever (2015) found that when increasing the bias of the forget gate, b_f, the performance of the LSTM network usually became better. Furthermore, Schmidhuber, Wierstra, Gagliolo, and Gomez (2007) proposed that LSTM was sometimes better trained by evolutionary algorithms combined with other techniques rather than by pure gradient descent.

2.2.3 LSTM with a Peephole Connection. As the gates of the above LSTM cells do not have direct connections from the cell state, there is a lack of essential information that harms the network's performance. In order to

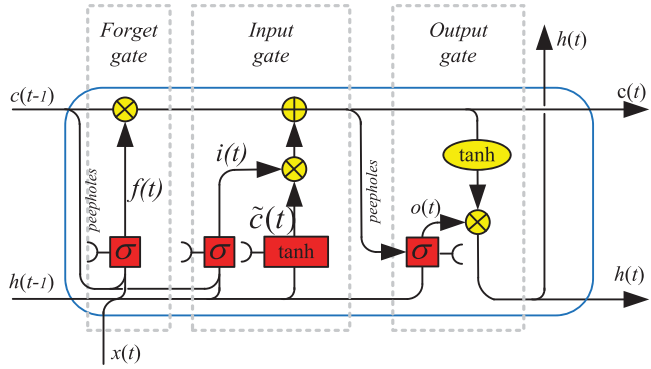


Figure 4: LSTM architecture with a peephole connection.

solve this problem, Gers and Schmidhuber (2000) extended the LSTM cell by introducing a peephole connection, as shown in Figure 4.

Based on the connections shown in Figure 4, the mathematical expressions can be expressed as follows:

$$
\begin{aligned}
f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + P_f \cdot c_{t-1} + b_f),\\
i_t &= \sigma(W_{ih}h_{t-1} + W_{ix}x_t + P_i \cdot c_{t-1} + b_i),\\
\tilde{c}_t &= \tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}),\\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t,\\
o_t &= \sigma(W_{oh}h_{t-1} + W_{ox}x_t + P_o \cdot c_t + b_o),\\
h_t &= o_t \cdot \tanh(c_t),
\end{aligned}
\tag{2.4}
$$

where P_f, P_i, and P_o are the peephole weights for the forget gate, input gate, and output gate, respectively. The peephole connections allow the LSTM cell to inspect its current internal states (Gers & Schmidhuber, 2001), and thus the LSTM with a peephole connection can learn stable and precise timing algorithms without teacher forcing (Gers & Schraudolph, 2002). To explore the LSTM with a peephole connection in depth, Greff, Srivastava, Koutník, Steunebrink, and Schmidhuber (2016) compared the performance of eight variants: no input gate, no forget gate, no output gate, no input activation function, no output activation function, coupled input and forget gate (CIFG), no peepholes, and full gate recurrence. Each variant differed from the original LSTM with a peephole connection by only a single change. The results showed that the forget and output gates are the most critical components, and removing any of them obviously decreases network performance. Moreover, the modified coupled input and forget gate could reduce the number of parameters and lower computational cost

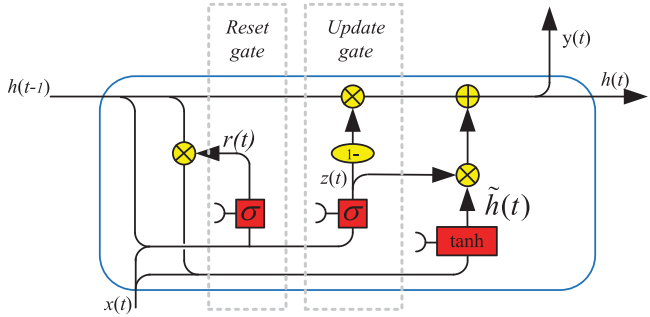


Figure 5: GRU cell architecture and connections.

without a significant decrease in network performance. Because of this powerful capacity, LSTM has become the focus of deep learning and has been applied to multiple tasks.

2.3 Gated Recurrent Unit. The learning capacity of the LSTM cell is superior to that of the standard recurrent cell. However, the additional parameters increase computational burden. Therefore, the gated recurrent unit (GRU) was introduced by Cho et al. (2014). Figure 5 shows the details of the architecture and connections of the GRU cell.

Based on the information shown in Figure 5, the mathematical expressions of the GRU cell are as follows:

$$\begin{aligned} r_t &= \sigma(W_{rh}h_{t-1} + W_{rx}x_t + b_r), \\ z_t &= \sigma(W_{zh}h_{t-1} + W_{zx}x_t + b_z), \\ \tilde{h}_t &= \tanh(W_{\tilde{h}h}(r_t \cdot h_{t-1}) + W_{\tilde{h}x}x_t + b_z), \\ h_t &= (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t. \end{aligned} \tag{2.5}$$

In order to reduce the number of parameters, the GRU cell integrates the forget gate and input gate of the LSTM cell as an update gate. The GRU cell has only two gates: an update gate and a reset gate. Therefore, it could save one gating signal and the associated parameters. The GRU is essentially a variant of vanilla LSTM with a forget gate. Since one gate is missing, the single GRU cell is less powerful than the original LSTM. The GRU cannot be taught to count or to solve context-free language (Weiss, Goldberg, & Yahav, 2018) and also does not work for translation (Britz, Goldie, Luong, & Le, 2017). Chung, Gulcehre, Cho, and Bengio (2014) empirically evaluated the performance of the LSTM network, GRU network, and traditional tanh-RNN and found that both the LSTM cell and GRU cell were superior to the traditional tanh unit, under the condition that each network had

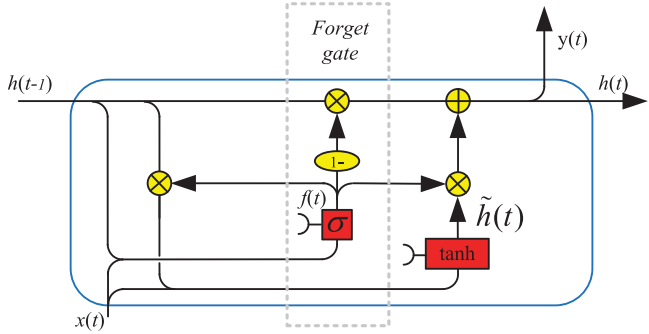


Figure 6: MGU cell schematic.

approximately the same number of parameters. Dey and Salemt (2017) modified the original GRU and evaluated properties of three variants of GRU—GRU-1, GRU-2, and GRU-3—on MNIST and IMDB data sets. The results showed that these three variants could reduce the computational expense while performing as well as the original GRU cell.

2.4 Minimal Gated Unit. To further reduce the number of cell parameters, Zhou, Sun, Liu, and Lau (2016) proposed the minimal gated unit, which only has one gate. The schematic of the MGU is shown in Figure 6. Based on the connections in the figure, the mathematical expressions of the MGU can be written as follows:

$$\begin{aligned} f_t &= \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f), \\ \tilde{h}_t &= \tanh(W_{\tilde{h}h}(f_t \cdot h_{t-1} + W_{\tilde{h}x}x_t + b_{\tilde{h}}), \\ h_t &= (1 - f_t) \cdot h_{t-1} + f_t \cdot \tilde{h}_t. \end{aligned} \tag{2.6}$$

The MGU cell involves only one forget gate. Hence, this kind of cell has a simpler structure and fewer parameters compared with LSTM and the GRU. Evaluation results (Zhou, Sun et al., 2016), based on various sequence data, showed that the performance of the MGU is comparable to that of the GRU. In order to further simplify the MGU, Heck and Salem (2017) introduced three model variants—MGU-1, MGU-2, and MGU-3—which reduced the number of parameters in the forget gate. They found that the performances of these variants were comparable to those of the MGU in tasks on the MNIST data set and the Reuters Newswire Topics data set.

2.5 Other LSTM Variants. In order to determine whether the architecture of the LSTM cell is optimal, Jozefowicz et al. (2015) evaluated over 10,000 different architectures and found three that outperformed both the

LSTM and GRU on the considered tasks: MUT-1, MUT-2, and MUT-3. The architectures of these cells are similar to that of the GRU with only two gates: an update gate and a reset gate. Neil, Pfeiffer, and Liu (2016) extended the LSTM cell by adding a new time gate, and proposed the phased LSTM cell. This new time gate leads to a sparse updating for the cell, which makes the phased LSTM achieve faster convergence than regular LSTMs on tasks of learning long sequences. Nina and Rodriguez (2016) simplified the LSTM cell by removing the input gate and coupling the forget gate and input gate. Removing the input gate produced better results in tasks that did not need to recall very long sequences.

Unlike the variants that modified the LSTM cell through decreasing or increasing gate functions, there are other kinds of LSTM variants. For example, Rahman, Mohammed, and Azad (2017) incorporated a biologically inspired variation into the LSTM cell and proposed the biologically variant LSTM. They changed only the updating of cell state $c(t)$ to improve cell capacity. Moreover, LSTM with working memory was introduced by Pulver and Lyu (2017). This modified version replaced the forget gate with a functional layer, whose input is decided by the previous memory cell value. The gated orthogonal recurrent unit (GORU) was proposed by Jing et al. (2017). They modified the GRU by an orthogonal matrix, which replaced the hidden state loop matrix of the GRU. This made the GORU possess the advantages of both an orthogonal matrix and the GRU structure. Irie, Tüske, Alkhouli, Schlüter, and Ney (2016) added highway connections to the LSTM cell and GRU cell to ensure an unobstructed information flow between adjacent layers. Furthermore, Veeriah, Zhuang, and Qi (2015) introduced a differential gating scheme for the LSTM neural network to solve the impact of spatiotemporal dynamics and then proposed the differential LSTM cell.

Although variants and analogous neural cells have been introduced, they can only be used on one or some specific data sets. Additionally, there is no cell variant that can outperform the LSTM cell on the whole. Thus, the LSTM cell is still the focus of deep learning for the processing of sequential data, and it remains the most popular recurrent cell in the literature. Therefore, in section 3, we give a comprehensive review of networks that consist of LSTM cells.

3 Different LSTM Networks

Due to the limited capacity of the single LSTM cell in handling engineering problems, the LSTM cells have to be organized into a specific network architecture when processing practical data. Despite the fact that all RNNs can be modified as LSTM networks by replacing the standard recurrent cell with the LSTM cell, this review discusses only the verified LSTM networks. We divide these LSTM networks into two broad categories: LSTM-dominated networks and integrated LSTM networks. LSTM-dominated networks are

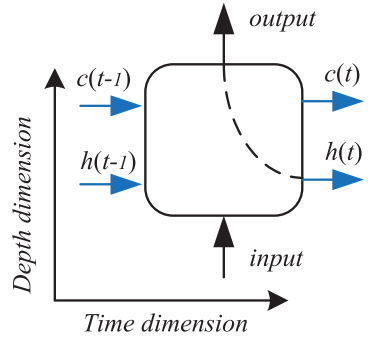


Figure 7: Simplified schematic diagram of the LSTM cell.

neural networks that are mainly constructed by LSTM cells. These networks focus on optimizing the connections of the inner LSTM cells so as to enhance the network properties (Nie, An, Huang, Yan, & Han, 2016; Zhao, Wang, Yan, & Mao, 2016). The integrated LSTM networks consist of LSTM layers and other components, such as a CNN and an external memory unit. The integrated LSTM networks mainly pay attention to integrating the advantageous features of different components when dealing with the target task.

Before introducing the LSTM networks, we simplify the schematic of the LSTM cell in section 2.2 as in Figure 7. In the figure, the dashed line indicates identity transformations. Therefore, the LSTM cell only outputs h_t along the depth dimension with no time delay and transmits both the c_t and h_t along the time dimension with one-unit-interval delay.

3.1 LSTM-Dominated Neural Networks

3.1.1 Stacked LSTM Network. In the usual application, the simplest method to add capacity and depth to the LSTM network is to stack the LSTM layers. Therefore, the stacked LSTM network is the most basic and simplest structure of the LSTM network (Fernández, Graves, & Schmidhuber, 2007). It can also be treated as a multilayer fully connected structure. A block of three recurrent layers is illustrated in Figure 8.

Then we unroll this schematic along the time dimension in Figure 9, where we assume that the sequence length is 4.

In the depth dimension of the stacked LSTM network, the output of the $(L-1)$th LSTM layer at time$t$ is h_t^{L-1}. This output is treated as the input, x_t^L, of the Lth layer. These output-input connections are the only relation between two adjacent layers. In the time dimension, the recurrent connections are only within one layer. Based on the previous analysis, the mathematical

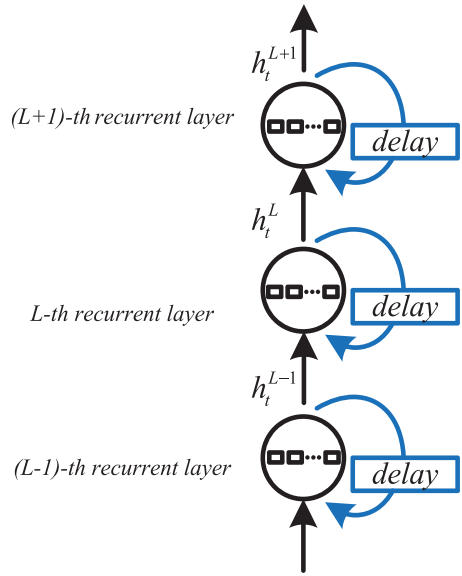


Figure 8: The stacked LSTM network.

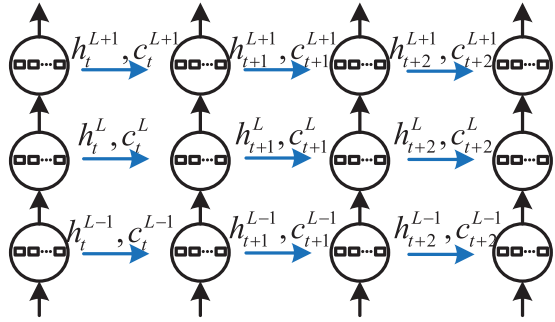


Figure 9: An unrolled stacked LSTM network.

expressions of the Lth LSTM layer could be expressed as follows:

$$
\begin{aligned}
f_t^L &= \sigma(W_{fh}^L h_{t-1}^L + W_{fx}^L h_t^{L-1} + b_f^L),\\
i_t^L &= \sigma(W_{ih}^L h_{t-1}^L + W_{ix}^L h_t^{L-1} + b_i^L),\\
\tilde{c}_t^L &= \tanh(W_{\tilde{c}h}^L h_{t-1}^L + W_{\tilde{c}x}^L h_{t-1}^L + b_{\tilde{c}}^L),\\
c_t^L &= f_t^L \cdot c_{t-1}^L + i_t^L \cdot \tilde{c}_t^L,
\end{aligned}
$$

$$o_t^L = \sigma(W_{oh}^L h_{t-1}^L + W_{ox}^L h_{t-1}^L + b_o^L),$$
$$h_t^L = o_t^L \cdot \tanh(c_t^L), \tag{3.1}$$

where h_t^{L-1} is the output of the $(L-1)$th LSTM layer at time t.

Because of the simple and efficient architecture, the stacked LSTM network has been widely adopted by researchers. Du, Zhang, Nguyen, and Han (2017) adopted the stacked LSTM network to solve the problem of vehicle-to-vehicle communication and discovered that the efficiency of the stacked LSTM-based regression model was much higher than that of logistic regression. Sutskever, Vinyals, and Le (2014) used a stacked LSTM network with four layers, and 1000 cells at each layer, to accomplish an English-to-French translation task. They found that reversing the order of source words could introduce short-term dependencies between the source and the target sentence; thus, the performance of this LSTM network was remarkably improved. Saleh, Hossny, and Nahavandi (2018) constructed a deep stacked LSTM network to predict the intent of vulnerable road users. When evaluated on the Daimler pedestrian path prediction benchmark data set (Schneider & Gavrila, 2013) for intent and path prediction of pedestrians in four unique traffic scenarios, the stacked LSTM was more powerful than the methodology relying on a set of specific hand-crafted features.

3.1.2 Bidirectional LSTM Network. Conventional RNNs are only able to make use of a previous context. In order to overcome this shortcoming, the bidirectional RNN (BRNN) was introduced by Schuster and Paliwal (1997). This kind of architecture could be trained in both time directions simultaneously, with separate hidden layers (i.e., forward layers and backward layers). Graves and Schmidhuber (2005) combined the BRNN with the LSTM cell and proposed the bidirectional LSTM. Figure 10 shows the internal connections of bidirectional LSTM recurrent layers.

In the architecture in Figure 10, the connections of forward layers are the same as in the stacked LSTM network, which computes sequence $(\vec{h}_t^L, \vec{c}_t^L)$ from $t = 1$ to T. However, for the backward layers, hidden sequence $(\overleftarrow{h}_t^L, \overleftarrow{c}_t^L)$ and output are iterated from $t = T$ to 1. Therefore, the mathematical expressions of the LSTM cell in backward layer L, at time t, could be written as follows:

$$\overleftarrow{f}_t^L = \sigma(W_{\overleftarrow{f}h}^L h_{t+1}^L + W_{\overleftarrow{f}x}^L h_t^{L-1} + b_{\overleftarrow{f}}^L),$$
$$\overleftarrow{i}_t^L = \sigma(W_{\overleftarrow{i}h}^L h_{t+1}^L + W_{\overleftarrow{i}x}^L h_t^{L-1} + b_{\overleftarrow{i}}^L),$$
$$\overleftarrow{\tilde{c}}_t^L = \tanh(W_{\overleftarrow{\tilde{c}}h}^L h_{t+1}^L + W_{\overleftarrow{\tilde{c}}x}^L h_t^{L-1} + b_{\overleftarrow{\tilde{c}}}^L),$$
$$\overleftarrow{c}_t^L = \overleftarrow{f}_t^L \cdot \overleftarrow{c}_{t+1}^L + \overleftarrow{i}_t^L \cdot \overleftarrow{\tilde{c}}_t^L,$$

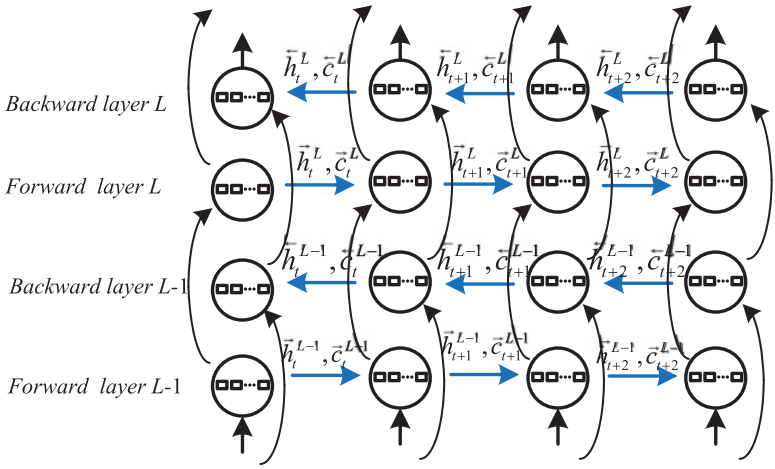


Figure 10: Internal connections of bidirectional LSTM.

$$\overleftarrow{o}_t^L = \sigma(W_{\overleftarrow{o}h}^L h_{t+1}^L + W_{\overleftarrow{o}x}^L h_t^{L-1} + b_{\overleftarrow{o}}^L),$$
$$\overleftarrow{h}_t^L = \overleftarrow{o}_t^L \cdot \tanh(\overleftarrow{c}_t^L). \tag{3.2}$$

The output of this architecture can be expressed as

$$y_t = W_{\vec{h}y}\vec{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y, \tag{3.3}$$

where $\vec{h}_t$ and $\overleftarrow{h}_t$ are the outputs of forward and backward layers, respectively.

Bidirectional LSTM networks have been widely adopted by researchers because of their excellent properties (Han, Wu, Jiang, & Davis, 2017; Yu, Xu, & Zhang, 2018). Thireou and Reczko (2007) applied this architecture to the sequence-based prediction of protein localization. The results showed that the bidirectional LSTM network outperforms the feedforward network and standard recurrent network. Wu, Zhang, and Zong (2016) investigated the different skip connections in a stacked bidirectional LSTM network and found that adding skip connections to the cell outputs with a gated identity function could improve network performance on the part-of-speech tagging task. Brahma (2018) extended the bidirectional LSTM to suffix bidirectional LSTM (SuBiLSTM), which improved the bidirectional LSTM network by encoding each suffix of the sequence. The SuBiLSTM outperformed the bidirectional LSTM in learning general sentence representations.

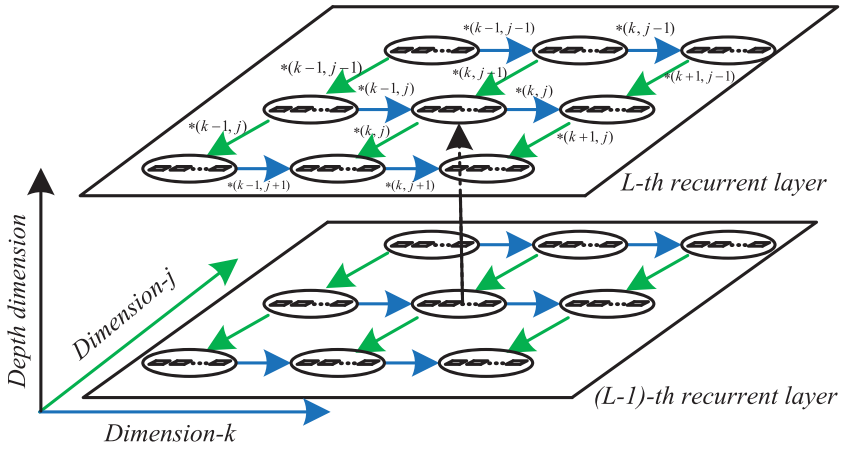
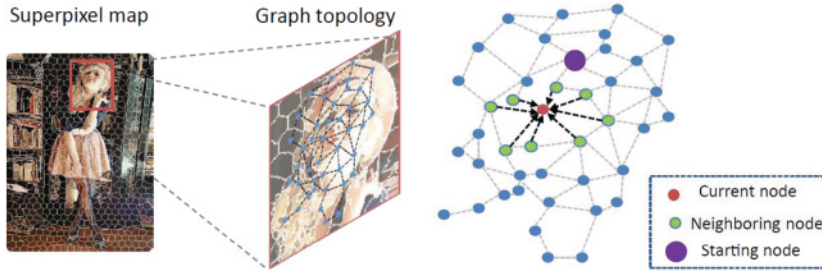


Figure 11: Unrolled 2D LSTM network.

3.1.3 Multidimensional LSTM Network. The standard RNNs can only be used to deal with one-dimensional data. However, in fields with multidimensional data like video processing, the properties of RNNs—the ability to access contextual information and robustness to input warping—are also desired. Graves, Fernández, and Schmidhuber (2007) introduced multidimensional LSTM (MDLSTM) to extend the application fields of RNNs. The core idea of MDLSTM was to build as many recurrent connections as the dimensions of the data. At each point in the data sequence, the recurrent layer L receives both the output of layer $L-1$ and its own activations from one step back along all dimensions. This means that the LSTM cells in layer L have n-forget gates in the n-dimensional LSTM network (one for each of the cell's previous states along every dimension). Figure 11 shows the unrolled two-dimensional (2D) LSTM network case.

In the 2D LSTM network, the blue lines and green lines indicate recurrent connections along dimensions k and j, respectively. The expression $*(k, j)$ denotes the recurrent information $(h^L_{k,j}, c^L_{k,j})$. Then the mathematical expression of the Lth LSTM layer is expressed as follows:

$$
\begin{aligned}
f^L_k &= \sigma(W^L_{f1k} h^L_{k-1,j} + U^L_{f1} h^{L-1}_{k,j} + b^L_{fk}),\\
f^L_j &= \sigma(W^L_{f2j} h^L_{k,j-1} + U^L_{f2} h^{L-1}_{k,j} + b^L_{fj}),\\
i^L_{k,j} &= \sigma(W^L_{ik} h^L_{k-1,j} + W^L_{ij} h^L_{k,j-1} + U^L_{i} h^{L-1}_{k,j} + b^L_{i}),\\
\tilde{c}^L_{k,j} &= \tanh(W^L_{\tilde{c}k} h^L_{k-1,j} + W^L_{\tilde{c}j} h^L_{k,j-1} + U^L_{\tilde{c}} h^{L-1}_{k,j} + b^L_{\tilde{c}}),\\
c^L_{k,j} &= f^L_k \cdot c^L_{k-1,j} + f^L_j \cdot c^L_{k,j-1} + i^L_{k,j} \cdot \tilde{c}^L_{k,j},
\end{aligned}
$$



(A) Superpixel graph topology (B) Corresponding Graph LSTM architecture

Figure 12: The proposed graph LSTM structure.

$$o_{k,j}^{L} = \sigma(W_{ok}^{L} h_{k-1,j}^{L} + W_{oj}^{L} h_{k,j-1}^{L} + U_{o}^{L} h_{k,j}^{L-1} + b_{o}^{L}),$$
$$h_{k,j}^{L} = o_{k,j}^{L} \cdot \tanh(c_{k,j}^{L}), \tag{3.4}$$

where $(h_{k-1,j}^{L}, c_{k-1,j}^{L})$ and $(h_{k,j-1}^{L}, c_{k,j-1}^{L})$ denote the one-step-back recurrent information of point (k, j) along dimension k and j, at LSTM layer L. $h_{k,j}^{L-1}$ is the output of point (k, j) at LSTM layer $L-1$.

Graves et al. (2007) extended the above unidirectional MDLSTM to the multidirectional MDLSTM and adopted this architecture to deal with the Air Freight database (McCarter & Storkey, 2007) and the MNIST database (LeCun, Bottou, Bengio, & Haffner, 1998). The results showed that this architecture is more robust to input warping than the state-of-the-art digit recognition algorithm. Li, Mohamed, Zweig, and Gong (2016a) constructed a MDLSTM network by time-frequency LSTM cells (Li, Mohamed, Zweig, & Gong, 2016b), which performed recurrence along the time and frequency axes. On the Microsoft Windows phone short message dictation task, the recurrence over both time and frequency axes promoted learning accuracy compared with a network that consisted of only time LSTM cells.

3.1.4 Graph LSTM Network. When processing graph-structured data, the existing pixel-wise LSTM structures usually assume that each pixel is influenced by fixed neighboring pixels. These structures include row LSTM (Oord, Kalchbrenner, & Kavukcuoglu, 2016), diagonal BiLSTM (Shabanian, Arpit, Trischler, & Bengio, 2017), and local-global LSTM (Liang, Shen, Xiang et al., 2016). These predefined recurrent sequences usually cause redundant computational costs. Liang, Shen, Xiang et al. (2016) extended the fixed topologies and proposed the graph LSTM network, based on the graph RNN network (Goller & Kuchler, 1996). The graph LSTM model assumes that each superpixel node is defined by its previous states and adaptive neighboring nodes. Figures 12a and 12b show a superpixel graph topology and the corresponding graph LSTM architecture, respectively. Instead of

using the fixed starting node and predefined updating route for all images, the starting node and node updating scheme of graph LSTM are dynamically specified.

The superpixel graph topology is acquired by image oversegmentation using a simple linear iterative cluster (Achanta et al., 2010). For convenience, we first define the average hidden states for neighboring nodes of node k:

$$\bar{h}_{k,t} = \frac{\sum_{j \in N_G(k)} (1(q_j = 1)h_{j,t+1} + 1(q_j = 0)h_{j,t})}{|N_G(k)|}, \tag{3.5}$$

where $N_G(k)$ denotes the total neighboring nodes of the visited node k. q_j indicates the state of node j, which is set at 1 if updated and otherwise 0. Furthermore, neighboring nodes have different influences on the visited node k. Graph LSTM utilizes the adaptive forget gates, $\bar{f}_{kj}$, $j \in N_g(k)$, to describe it. The graph LSTM cell is updated as follows:

$$\begin{aligned}
i_{k,t} &= \sigma(W_{ih}h_{k,t-1} + W_{i\bar{h}}\bar{h}_{k,t-1} + W_{ix}x_{k,t} + b_i),\\
f_{k,t} &= \sigma(W_{fh}h_{k,t-1} + W_{fx}x_{k,t} + b_f),\\
\bar{f}_{kj,t} &= \sigma(W_{\bar{f}h}h_{j,t-1} + W_{\bar{f}x}x_{k,t} + b_{\bar{f}}), (j \in N_G(k)),\\
\tilde{c}_{k,t} &= \tanh(W_{\tilde{c}h}h_{k,t-1} + W_{\tilde{c}\bar{h}}\bar{h}_{k,t-1} + W_{\tilde{c}x}x_{k,t} + b_{\tilde{c}}),\\
c_{k,t} &= f_{k,t} \cdot c_{k,t-1} + i_{k,t} \cdot \tilde{c}_{k,t}\\
&\quad + \frac{\sum_{j \in N_G(k)} (1(q_j = 1)\bar{f}_{kj,t} \cdot c_{j,t} + 1(q_j = 0)\bar{f}_{kj,t} \cdot c_{j,t-1})}{|N_G(k)|},\\
o_{k,t} &= \sigma(W_{oh}h_{k,t-1} + W_{o\bar{h}}\bar{h}_{k,t-1} + W_{ox}x_{k,t} + b_o),\\
h_{k,t} &= \tanh(o_{k,t} \cdot c_{k,t}),
\end{aligned} \tag{3.6}$$

where the input $x_{k,t}$ is the feature of graph node k.

Liang, Shen, Feng et al. (2016) demonstrated the superiority of the graph LSTM network on four data sets: the Fashionista data set (Yamaguchi, 2012), PASCAL-Person-Part data set (Chen et al., 2014), ATR data set (Liang et al., 2015), and Horse-Cow Parsing data set (Wang & Yuille, 2015), and the results showed that this network could perform well in semantic object parsing. Liang et al. (2017) further extended the graph LSTM model to the structure-evolving LSTM. The structure-evolving LSTM merges the graph nodes stochastically and thus learns the multilevel graph structures in a progressive and stochastic way. The effectiveness evaluation shows that the structure-evolving LSTM outperforms the state-of-the-art LSTM models.

3.1.5 Grid LSTM Network. The Grid LSTM network, proposed by Kalchbrenner, Danihelka, and Graves (2015), could also be used to process

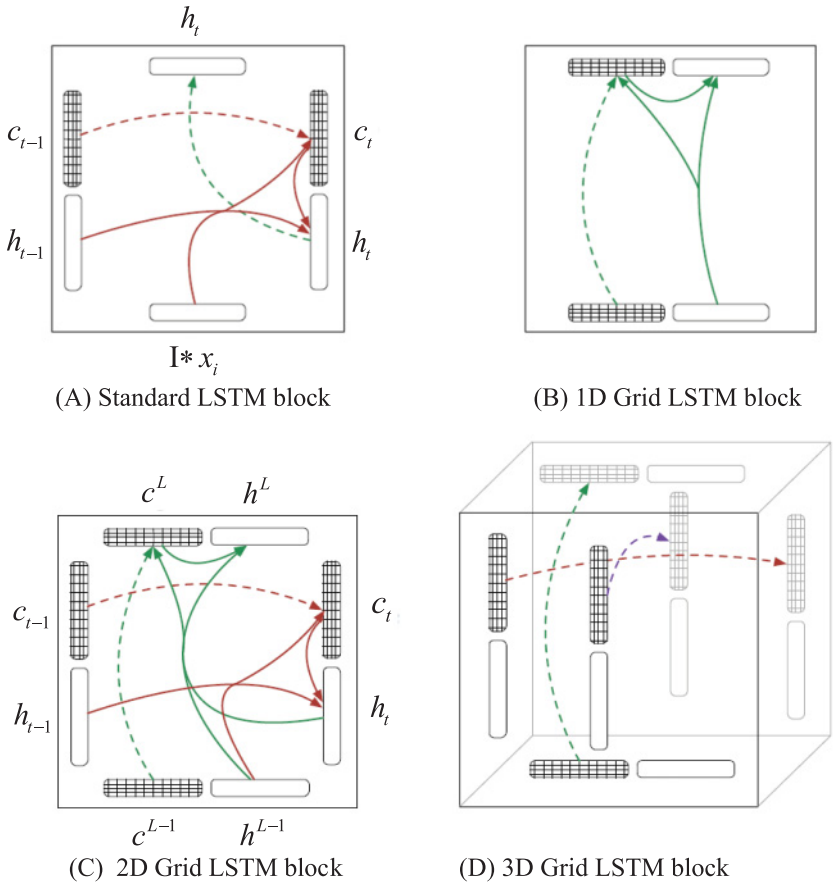


Figure 13: Blocks of the stacked LSTM and the grid LSTM.

multidimensional data. This architecture arranges the LSTM cells in a grid of one or more dimensions. Different from the existing networks, the grid LSTM network has recurrent connections along the depth dimension. In order to explain the architecture of the grid LSTM, the blocks from the standard LSTM and Grid LSTM are shown in Figure 13.

In the blocks, the dashed lines denote identity transformations. The red lines, green lines, and purple lines indicate the transformations along different dimensions. Compared with the standard LSTM block, the 2D Grid LSTM block has the cell memory vector along the vertical dimension.

Kalchbrenner et al. (2015) put forward that the one-dimensional (1D) grid LSTM network is the architecture that replaces the transfer functions (e.g., tanh and ReLU; Nair & Hinton, 2010) of the feedforward network by

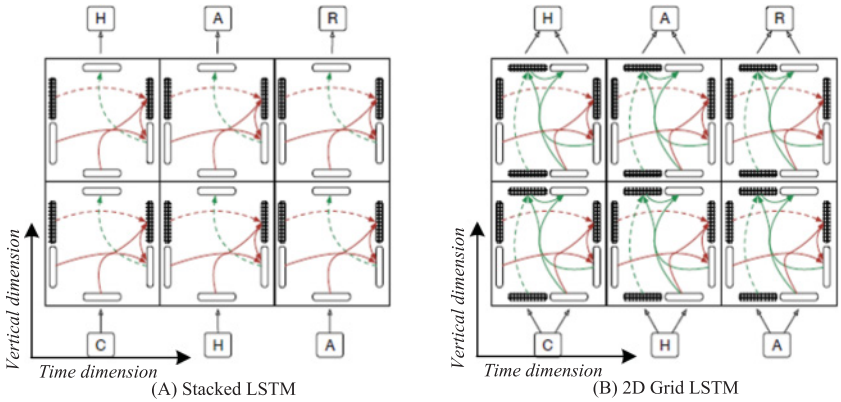


Figure 14: The stacked LSTM and 2D grid LSTM.

the 1D grid LSTM block. Thus, the 2D Grid LSTM network is similar to the stacked LSTM network, but there are recurrent connections along the depth dimension. The grid LSTM with three or more dimensions corresponds to MDLSTM (Graves, 2012), but the N-way recurrent interactions exit along all dimensions. Figure 14 shows the stacked LSTM network and 2D Grid LSTM network.

Based on the mathematical expressions of the LSTM cell block in equation 2.3, in the 2D Grid LSTM network along the time dimension, the expressions of the LSTM cell at layer L and time t are expressed as follows:

$$\begin{aligned}
f_t^L &= \sigma(W_{fh}^L h_{t-1}^L + W_{fx}^L h_t^{L-1} + b_f^L),\\
i_t^L &= \sigma(W_{ih}^L h_{t-1}^L + W_{ix}^L h_t^{L-1} + b_i^L),\\
\tilde{c}_t^L &= \tanh(W_{\tilde{c}h}^L h_{t-1}^L + W_{\tilde{c}x}^L h_t^{L-1} + b_{\tilde{c}}^L),\\
c_t^L &= f_t^L \cdot c_{t-1}^L + i_t^L \cdot \tilde{c}_t^L,\\
o_t^L &= \sigma(W_{oh}^L h_{t-1}^L + W_{ox}^L h_t^{L-1} + b_o^L),\\
h_t^L &= o_t^L \cdot \tanh(c_t^L).
\end{aligned} \tag{3.7}$$

Furthermore, along the vertical dimension, we replace the input x_t of equation 3.1 with the h_t^L in the time dimension. Then the expressions are expressed as follows:

$$\begin{aligned}
f_L^t &= \sigma(W_{fh}^t h_{L-1}^t + W_{fx}^t h_t^L + b_f^t),\\
i_L^t &= \sigma(W_{ih}^t h_{L-1}^t + W_{ix}^t h_t^L + b_i^t),\\
\tilde{c}_L^t &= \tanh(W_{\tilde{c}h}^t h_{L-1}^t + W_{\tilde{c}x}^t h_t^L + b_{\tilde{c}}^t),
\end{aligned}$$

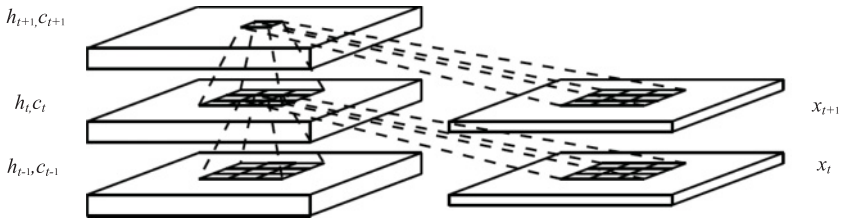


Figure 15: Recurrent structure of the ConvLSTM layer.

$$
\begin{aligned}
c_L^t &= f_L^t \cdot c_{L-1}^t + i_L^t \cdot \tilde{c}_L^t, \\
o_L^t &= \sigma(W_{oh}^t h_{L-1}^t + W_{ox}^t h_t^L + b_o^t), \\
h_L^t &= o_L^t \cdot \tanh(c_L^t).
\end{aligned} \tag{3.8}
$$

Kalchbrenner et al. (2015) found that the 2D grid LSTM outperformed the stacked LSTM on the task of memorizing sequences of numbers (Zaremba & Sutskever, 2014). The recurrent connections along the depth dimension could improve the learning properties of grid LSTM. In order to lower the computational complexity of the grid LSTM network, Li and Sainath (2017) compared four grid LSTM variations and found that the frequency-block grid LSTM reduced computation costs without loss of accuracy on a 12,500-hour voice search task.

3.1.6 Convolutional LSTM Network. The fully connected LSTM layer contains too much redundancy for spatial data (Sainath, Vinyals, Senior, & Sak, 2015). Therefore, to accomplish a spatiotemporal sequence forecasting problem, Shi et al. (2015) proposed the convolutional LSTM (ConvLSTM), which had convolutional structures in the recurrent connections. Figure 15 shows the recurrent structure of the ConvLSTM layer.

The ConvLSTM network uses the convolution operator to calculate the future state of a certain cell, and then the future state is determined by the inputs and past states of its local neighbors. The ConvLSTM cell can be expressed as follows:

$$
\begin{aligned}
f_t &= \sigma(W_{fh} * h_{t-1} + W_{fx} * x_t + b_f), \\
i_t &= \sigma(W_{ih} * h_{t-1} + W_{ih} * x_t + b_i), \\
\tilde{c}_t &= \tanh(W_{\tilde{c}h} * h_{t-1} + W_{\tilde{c}x} * x_t + b_{\tilde{c}}), \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \\
o_t &= \sigma(W_{oh} * h_{t-1} + W_{ox} * x_t + b_o), \\
h_t &= o(t) \cdot \tanh(c(t)),
\end{aligned} \tag{3.9}
$$

where '$*$' is the convolution operator and '$\cdot$', as before, denotes the Hadamard product.

Wei, Zhou, Sankaranarayanan, Sengupta, and Samet (2018) adopted the ConvLSTM network to solve tweet count prediction, a spatiotemporal sequence forecasting problem. The results of experiments on the city of Seattle showed that the proposed network consistently outperforms the competitive baseline approaches: the autoregressive integrated moving average model, ST-ResNet (Zhang, Zheng, & Qi, 2016), and Eyewitness (Krumm & Horvitz, 2015). Zhu, Zhang, Shen, and Song (2017) combined the 3D CNN and ConvLSTM to construct a multimodal gesture recognition model. The 3D CNN and ConvLSTM learned the short-term and long-term spatiotemporal features of gestures, respectively. When verified on the Sheffield Kinect Gesture data set and the ChaLearn LAP large-scale isolated gesture data set, the results showed that the proposed method performs better than other models. Liu, Zhou, Hang, and Yuan (2017) extended both the ConvLSTM and bidirectional LSTM and presented the bidirectional–convolutional LSTM architecture to learn spectral–spatial features from hyperspectral images.

3.1.7 Depth-gated LSTM Network. The stacked LSTM network is the simplest way to construct DNN. However, Yao, Cohn, Vylomova, Duh, and Dyer (2015) pointed out that the error signals in the stacked LSTM network might be either diminished or exploded because the error signals from the top have to be backpropagated through many layers of nonlinear transformations. In order to solve this problem, they proposed the depth-gated LSTM (DGLSTM), in which a depth gate was used to connect memory cells of neighboring LSTM layers. An illustration of the DGLSTM architecture is shown in Figure 16.

In the depth dimension, there is one additional connection between memory cells in the DGLSTM compared with the stacked LSTM. This depth gate controls the information flow between the memory cells of different layers. The function of the depth gate at layer L at time t is defined as

$$d_t^L = \sigma\left(W_{xd}^L x_t^L + \omega_{cd}^L \cdot c_{t-1}^L + \omega_{ld}^L \cdot c_t^{L-1} + b_d^L\right), \tag{3.10}$$

where ω_{cd}^L and ω_{ld}^L are the weight vectors to relate the past memory cell and the lower-layer memory cell, respectively. Moreover, ω_{ld}^L should be a matrix instead of a vector if memory cells in lower and upper layers have different dimensions. Therefore, in the DGLSTM, the memory cell at layer L is computed as

$$c_t^L = i_t^L \cdot \tanh(W_{cx}^L x_t^L + W_{ch}^L h_{t-1}^L) + f_t^L \cdot c_{t-1}^L + d_t^L \cdot c_t^{L-1}. \tag{3.11}$$

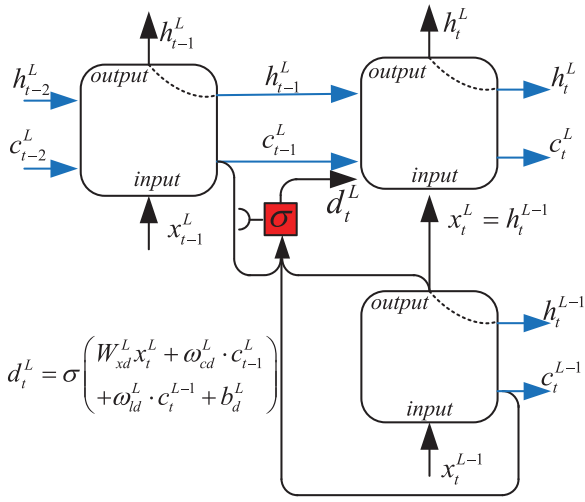


Figure 16: An illustration of DGLSTM.

The DPLSTM architecture is inspired by the highway network (Kim, El-Khamy, & Lee, 2017; Srivastava, Greff, & Schmidhuber, 2015) and grid LSTM (Kalchbrenner et al., 2015). Yao et al. (2015) used the varied depth of DGLSTMs to accomplish the Chinese-to-English machine translation task The results showed that the performance of the DGLSTM network is always better than that of the stacked LSTM network. Zhang, Chen et al. (2015) proposed efficient algorithms to train the DGLSTM network using both frame and sequence discriminative criteria and achieved even better improvements.

3.1.8 Gated-Feedback LSTM Network. To deal with the problem of learning multiple adaptive timescales, Chung, Gulcehre, Cho, & Bengio (2015) proposed the gated-feedback RNN (GF-RNN) and the gated-feedback LSTM (GF-LSTM) network. For each pair of layers, the GF-RNN uses a global gating unit to allow and control signals flowing from upper recurrent layers to lower layers, which are called gated-feedback connections. Figure 17 shows the architecture of the GF-RNN.

In the GF-RNN, the recurrent connections between two units are gated by a logistics unit, which is called a global reset gate. The signals from the ith layer h_{t-1}^i to the jth layer h_t^j are determined by the global reset gate and can be computed as follows:

$$g^{j\to j} = \sigma\left(W_g^{j-1\to j} h_t^{j-1} + \sum_{i'=1}^{L} U_g^{i'\to j} h_{t-1}^{i'}\right), \tag{3.12}$$

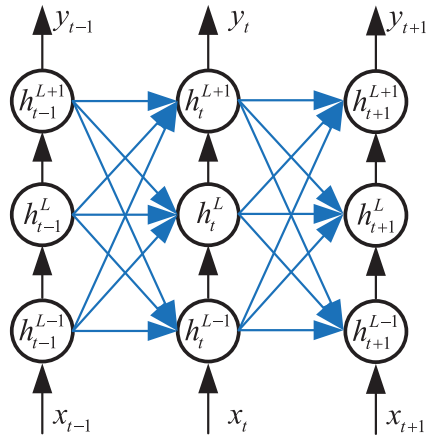


Figure 17: Architecture of the GF-RNN.

where $W_g^{j-1\to j}$ and $U_g^{i\to j}$ are the weight vectors for the input and hidden activations of the ith layer at time step $t-1$, respectively, and L is the number of hidden layers. For $j=1$, h_t^{j-1} is the input x_t. In the GF-LSTM network, the unit-wise gates, such as the input gate, forget gate, and output gate, are not modified by the global reset gate. However, the cell state $c(t)$, in equation 2.3, is controlled by it. At layer j, the memory content is computed by

$$\tilde{c}_t^j = \tanh(W_c^{j-1\to j} h_t^{j-1} + \sum_{i=1}^{L} g^{i\to j} U_c^{i\to j} h_{t-1}^i + b_{\tilde{c}}). \tag{3.13}$$

The proposed GF-RNN was evaluated against the conventional stacked RNN on the task of Python program evaluation (Koutnik, Greff, Gomez, & Schmidhuber, 2014), and the results showed that the GF-RNN outperforms the stacked RNN.

3.1.9 Tree-Structured LSTM Network. Most of the existing LSTM networks are chain-structured. These networks perform well in machine translation and speech recognition. However, when these chain-structured networks need to combine words and phrases in natural language processing, they exhibit poor properties (Li, Luong, Dan, & Hovy, 2015; Zhang, Lu, & Lapata, 2015). Therefore, Zhu, Sobhani, and Guo (2015) and Tai, Socher, and Manning (2015) extended the chain-structured LSTM networks to tree-structured networks, based on the original tree-structured recursive models (Goller & Kuchler, 1996; Sperduti & Starita, 1997; Francesconi et al., 1997; Frasconi, Gori, & Sperduti, 1998). Figure 18 shows a tree-structured LSTM network.

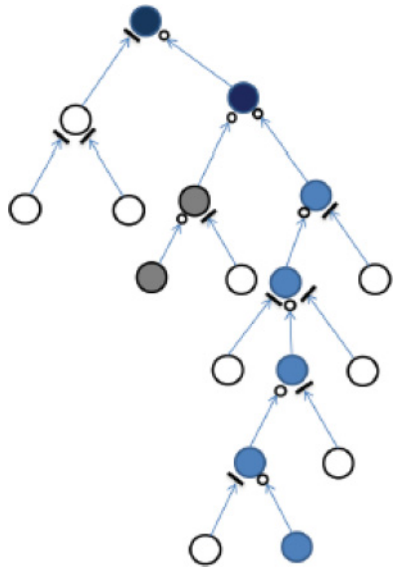


Figure 18: Tree-structured LSTM network. The short line and small circle at each arrowhead indicate a block or pass of information, respectively. In the real model, the gating is a soft version of gating.

The cell state of the tree-structured LSTM depends on the states of its child units. The updating of the LSTM cell can be expressed as follows:

$$
\begin{aligned}
i_t &= \sigma(W^L_{hi}h^L_{t-1} + W^R_{hi}h^R_{t-1} + W^L_{ci}c^L_{t-1} + W^R_{ci}c^R_{t-1} + b_i),\\
f^L_t &= \sigma(W^L_{hf_l}h^L_{t-1} + W^R_{hf_l}h^R_{t-1} + W^L_{cf_l}c^L_{t-1} + W^R_{cf_l}c^R_{t-1} + b_{f_l}),\\
f^R_t &= \sigma(W^L_{hf_r}h^L_{t-1} + W^R_{hf_r}h^R_{t-1} + W^L_{cf_r}c^L_{t-1} + W^R_{cf_r}c^R_{t-1} + b_{f_r}),\\
\tilde{c}_t &= \tanh(W^L_{h\tilde{c}}h^L_{t-1} + W^R_{h\tilde{c}}h^R_{t-1} + b_{\tilde{c}}),\\
c_t &= f^L_t \cdot c^L_{t-1} + f^R_t \cdot c^R_{t-1} + i_t \cdot \tilde{c}_t,\\
o_t &= \sigma(W^L_{ho}h^L_{t-1} + W^R_{ho}h^R_{t-1} + W_{co}c_t + b_o),\\
h_t &= o_t \cdot \tanh(c_t),
\end{aligned}
\tag{3.14}
$$

where f^L and f^R are the left and right forget gates, respectively. At time t, the inputs of the node LSTM cell are hidden vectors of its two children, which are denoted as h^L_{t-1} and h^R_{t-1}.

Zhu et al. (2015) proved that the tree-structured LSTM network outperformed other recursive models in learning distributed sentiment representations for texts. Tai et al. (2015) proposed two tree-structured LSTM

architectures: the Child-Sum Tree-LSTM, and the N-ary Tree-LSTM. The tree nodes in both models are dependent on multiple child units. The Child-Sum Tree-LSTM adopts the sum of child hidden states h_k to update the cell. However, the N-ary Tree-LSTM introduces separate parameter matrices for each child and is able to learn more finely grained information from its children. Teng and Zhang (2016) further extended the tree-structured LSTM to the bidirectional tree-structured LSTM with head lexicalization. The general tree-structured LSTM network constitutes trees in the bottom-up direction. Therefore, only the leaf nodes can use the input word information. In order to propagate head words from leaf nodes to every constituent node, Teng and Zhang (2016) proposed the automatic head lexicalization for a general tree-structured LSTM network, and built a tree LSTM in the top-down direction. Miwa and Bansal (2016) further stacked a bidirectional tree-structured LSTM network on a bidirectional sequential LSTM network to accomplish end-to-end relation extraction. Additionally, Niu, Zhou, Wang, Gao, and Hua (2017) extended the tree-structured LSTM to the hierarchical multimodal LSTM (HM-LSTM) for the problem of dense visual-semantic embedding. Since the HM-LSTM could exploit the hierarchical relations between whole images and image regions and between sentences and phrases, this model outperformed other methods on Flickr8K (Graves, 2014), Flickr30K (Plummer et al., 2017), and MS-COCO (Chen et al., 2015; Lin et al., 2014) data sets.

3.1.10 Other Networks

Coupled LSTM Network. Inspired by the grid LSTM and the multidimensional RNN, Liu, Qiu, and Huang (2016) proposed two coupled LSTMs architectures, the loosely coupled LSTM (LC-LSTM) and the tightly coupled LSTM (TC-LSTM), to model the interactions of sentence pairs. These two-coupled LSTM networks model the strong interactions of two sentences through the connections between hidden states.

Deep Fusion LSTM Networks. Liu, Qiu, Chen, and Huang (2016) adopted a deep fusion LSTM (DF-LSTM) network to extract the strong interaction of text pairs. This network consisted of two interdependent LSTMs. In order to capture more complicated matching patterns, the authors then used external memory to enhance the memory capacity of LSTMs. They gave a qualitative analysis for the model capacity and demonstrated the model efficacy on the Stanford Natural Language Inference Corpus.

Multiplicative LSTM Network. The weight matrices of the usual LSTM networks are fixed for different inputs. These networks are inexpressive when dealing with sequences of discrete mutually exclusive elements. Therefore, Krause, Lu, Murray, and Renals (2016) combined the LSTM with the multiplicative RNN architecture and obtained the multiplicative LSTM, which possessed flexible input-dependent weight matrices. In a series of character-level language modeling tasks, this model outperformed the stacked LSTM.

Nested LSTMs Network. The nested LSTM (NLSTM) network was proposed by Moniz and Krueger (2018). They adopted nesting to add the depth of the network. In the NLSTM network, the memory cell of the outer LSTM cell is computed by the inner LSTM cell, and $c_t^{outer} = h_t^{inner}$. In the task of character-level language modeling, the NLSTM network is a promising alternative to the stacked architecture, and it outperforms both stacked and single-layer LSTM networks with similar numbers of parameters.

3.2 Integrated LSTM Networks. Sometimes the capacity of a single LSTM network cannot meet the practical engineering requirements, and thus there are some integrated LSTM networks that consist of an LSTM layer/network and other components, such as a convolution neural network and an external memory unit. These integrated LSTM networks take advantage of different components.

3.2.1 Neural Turing Machine. Because the RNN needs to update the current cell states by inputs and previous states, the memory capacity is the essential feature of the RNN. In order to enhance the memory capacity of networks, neural networks have been adopted to learn to control end-to-end differentiable stack machines (Sun, 1990; Mozer, & Das, 1993). Schmidhuber (1993) and Schlag and Schmidhuber (2017) extended the above structure, and used the RNN to learn and control end-to-end-differentiable fast weight memory. Furthermore, Graves, Wayne, and Danihelka (2014) proposed the neural Turing machine (NTM) with an LSTM controller, which took inspiration from both the design of digital computers and the biological working memory models.

In Figure 19, the dashed line displays the division between the NTM and the external environment. The RNN receives inputs and emits outputs during each update cycle. At the same time, the RNN also exchanges information with the memory matrix via the read and write heads. The mathematical expression of the neural network is decided by the type of the selected RNN. Graves et al. (2014) selected the LSTM cell to construct the neural network and compared the performance of this NTM and a standard LSTM network. The results showed that NTM possessed advantages over the LSTM network. Xie and Shi (2018) improved the training speed by designing a new read–write mechanism for the NTM, which used convolution operations to extract global memory features.

3.2.2 DBN-LSTM Network. This network is the combination of a deep belief network and LSTM (DBN-LSTM), and was introduced by Vohra, Goel, and Sahoo (2015). It is an extension of the RNN-DBN (Goel, Vohra, & Sahoo, 2014). In the amalgamation of the DBN and LSTM, the multilayer DBN helps in high-level representation of data, while the LSTM network provides temporal information. Compared with the RNN-DBN, the most notable improvement of DBN-LSTM is that it replaces the standard recurrent

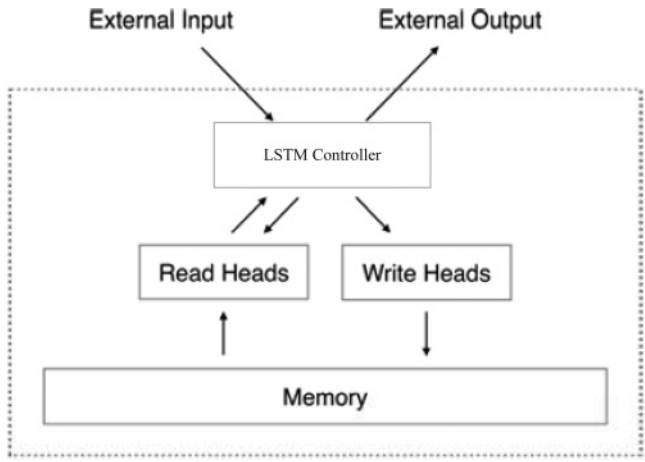


Figure 19: Schematic of the NTM with an LSTM controller.

cell with the LSTM cell, which ensures that the model can deal with a longer time duration.

3.2.3 Multiscale LSTM Network. Cheng et al. (2016) combined a preprocessing block and LSTM network to form the multiscale LSTM (MS-LSTM) network. The preprocessing block helps select a proper timescale for the input data, and the LSTM layer is adopted to model the processed sequential data. In processing dynamic Internet traffic, Cheng et al. (2016) used this model to learn the Internet traffic pattern in a flexible time window. Peng, Zhang, Liang, Liu, and Lin (2016) combined the MS-LSTM with a CNN and constructed a recurrent architecture for geometric scene parsing.

3.2.4 CFCC-LSTM Network. To predict sea surface temperature, Yang et al. (2017) proposed a CFCC-LSTM network, which was composed of a fully connected LSTM (FC-LSTM) and a CNN. They first introduced a 3D grid to preprocess the information and then adopted an FC-LSTM layer to predict the temporal information in the 3D grid and a convolution operation to handle the spatial information. The results on two data sets (the China Ocean data set and the Bohai Sea data set) showed the effectiveness of this model.

3.2.5 C-LSTM Network. The C-LSTM network is also a combination of a CNN and LSTM network. It was proposed by Zhou, Wu, Zhang, and Zhou (2016). This network can extract a sequence of higher-level phrase representations by the CNN, and then this information is fed into the LSTM network to obtain the sentence representation in the task of sentence and document modeling.

3.2.6 LSTM-in-LSTM Network. Song, Tang, Xiao, Wu, and Zhang (2016) combined a Deep-CNN network and an LSTM-in-LSTM architecture to generate rich, finely grained textual descriptions of images. The LSTM-in-LSTM architecture consists of an inner LSTM cell and an outer LSTM cell. This architecture can learn the contextual interactions between visual cues and can thus predict long sentence descriptions.

4 Conclusion

We have systematically reviewed various LSTM cell variants and LSTM networks. The LSTM cell is the basic node of LSTM networks. Different variants outperform the standard LSTM cell on some characteristics and tasks. For example, the GRU has a small number of cell parameters. However, there is no variant that can surpass the standard LSTM cell in all aspects. As for LSTM networks, there are two major categories: LSTM-dominated neural networks and integrated LSTM networks. In order to enhance the network properties of some specific tasks, LSTM-dominated networks focus on optimizing the connections between inner LSTM cells. Integrated LSTM networks mainly pay attention to integrating the advantageous features of different components, such as the convolution neural network and external memory unit, when dealing with the target task. Current LSTM models have acquired incredible success on numerous tasks. Nevertheless, there are still some directions to augment RNNs with more powerful properties.

First, more efficient recurrent cells should be explored through explicit knowledge. The recurrent cells are the basic nodes, and the properties of the networks depend on recurrent cells to some extent. However, the current studies on recurrent cells are mainly empirical explorations. The handling capacity of a specific cell structure is ambiguous. Therefore, the relationship between the cell's handling capacity and data structure needs to be clarified.

Second, RNNs should be combined with external memory to strengthen their memory capacity, such as in NTM and differentiable neural computer (Graves et al., 2016). Differing from the CNN and DNN, the RNNs require memory capacity because they need to update the cell states by inputs and previous states. However, neural networks are not good at storing information. The external memory could greatly increase the memory capacity of RNNs and help them handle the long-term dependencies.

Finally, adopting adaptive computation time is an interesting direction. At present, for each time step, the amount of computation in current RNNs is the same. This is no problem when tasks have the same degree of difficulty. However, more time should be spent on individual tasks of higher difficulty, and researchers should allow the RNNs to execute multiple steps of computation during each time step based on the degree of difficulty of the task. Self-delimiting (SLIM) RNN (Schmidhuber, 2012) is a typical network, and SLIM RNN could decide during the computation which parts of

itself are used at all, and for how long, learning to control time through a "halt unit." For each time step the amount of computation in adaptive computation time RNNs may be different, and they could deal with the tasks with different difficulties well.

Acknowledgments

We thank LetPub (www.letpub.com) for its linguistic assistance during the preparation of this review. This work was supported by the National Nature Science Foundation of China (grants 61773386, 61573365, 61573366, 61573076), the Young Elite Scientists Sponsorship Program of China Association for Science and Technology (grant 2016QNRC001), and the National Key R&D Program of China (grant 2018YFB1306100).

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., & Süsstrunk, S. (2010). *SLIC superpixels*. https://www.researchgate.net/publication/44234783_SLIC_superpixels

Altché, F., & Fortelle, A. D. L. (2017). An LSTM network for highway trajectory prediction. In *Proceedings of the IEEE 20th International Conference on Intelligent Transportation Systems*. Piscataway, NJ: IEEE.

Bengio, Y. (2009). Learning deep Architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Brahma, S. (2018). Suffix bidirectional long short-term memory. http://www.researchgate.net/publication/325262895_Suffix_Bidirectional_Long_Short-Term_Memory

Britz, D., Goldie, A., Luong, M. T., & Le, Q. (2017). *Massive exploration of neural machine translation architectures*. arXiv:1703.03906.

Brown, B., Yu, X., & Garverick, S. (2004). Mixed-mode analog VLSI continuous-time recurrent neural network. In *Circuits, Signals, and Systems: IASTED International Conference Proceedings*.

Carrio, A., Sampedro, C., Rodriguez-Ramos, A., & Campoy, P. (2017). A review of deep learning methods and applications for unmanned aerial Vehicles. *Journal of Sensors*, *2*, 1–13.

Chen, T. B., & Soo, V. W. (1996). A comparative study of recurrent neural network architectures on learning temporal sequences. In *Proceedings of the IEEE International Conference on Neural Networks*. Piscataway, NJ: IEEE.

Chen, X., Fang, H., Lin, T. Y., Vedantam, R., Gupta, S., Dollar, P., & Zitnick, C. L. (2015). *Microsoft COCO captions: Data collection and evaluation server*. arXiv:504.00325v2.

Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., & Yuille, A. (2014). Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE.

Cheng, M., Xu, Q., Lv, J., Liu, W., Li, Q., & Wang, J. (2016). MS-LSTM: A multi-scale LSTM model for BGP anomaly detection. In *Proceedings of the IEEE 24th International Conference on Network Protocols*. Piscataway, NJ: IEEE.

Cho, K., Van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). *Learning phrase representations using RNN encoder-decoder for statistical machine translation*. arXiv:1406.1078v3.

Chung, J., Gulcehre, C., Cho, K. H., & Bengio, Y. (2014). *Empirical evaluation of gated recurrent neural networks on sequence modeling*. arXiv:1412.3555v1.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2015). *Gated feedback recurrent neural networks*. arXiv:1502.02367v1.

Deng, L. (2013). Three classes of deep learning architectures and their applications: A tutorial survey. In *APSIPA transactions on signal and information processing*. Cambridge: Cambridge University Press.

Dey, R., & Salemt, F. M. (2017). *Gate-variants of gated recurrent unit (GRU) neural networks*. In *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems*. Piscataway, NJ: IEEE.

Du, X., Zhang, H., Nguyen, H. V., & Han, Z. (2017). Stacked LSTM deep learning model for traffic prediction in vehicle-to-vehicle communication. In *Proceedings of the IEEE Vehicular Technology Conference*. Piscataway, NJ: IEEE.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211.

Fernández, S., Graves, A., & Schmidhuber, J. (2007a). Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.

Fernández, S., Graves, A., & Schmidhuber, J. (2007b). An application of recurrent neural networks to discriminative keyword spotting. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 220–229). Berlin: Springer.

Francesconi, E., Frasconi, P., Gori, M., Marinai, S., Sheng, J. Q., Soda, G., & Sperduti, A. (1997). Logo recognition by recursive neural networks. In *Proceedings of the International Workshop on Graphics Recognition* (pp. 104–117). Berlin: Springer.

Frasconi, P., Gori, M., & Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, *9*(5), 768–786.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*(4), 193–202.

Gallagher, J. C., Boddhu, S. K., & Vigraham, S. (2005). A reconfigurable continuous time recurrent neural network for evolvable hardware applications. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*. Piscataway, NJ: IEEE.

Gers, F. (2001). *Long short-term memory in recurrent neural networks*. PhD diss., Beuth Hochschule für Technik Berlin.

Gers, F. A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*. Piscataway, NJ: IEEE.

Gers, F. A., & Schmidhuber, J. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Trans. Neural. Netw.*, *12*(6), 1333–1340.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471.

Gers, F. A., & Schraudolph, N. N. (2002). Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, *3*, 115–143.

Goel, K., Vohra, R., & Sahoo, J. K. (2014). Polyphonic music generation by modeling temporal dependencies using a RNN-DBN. In *Proceedings of the International Conference on Artificial Neural Networks*. Berlin: Springer.

Goller, C., & Kuchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*, *1*, 347–352.

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks*. Berlin: Springer.

Graves, A. (2014). *Generating sequences with recurrent neural networks*. arXiv:1308.0850v5.

Graves, A., Fernández, S., & Schmidhuber, J. (2007). Multi-dimensional recurrent neural networks. In *Proceedings of the International Conference on Artificial Neural Networks*. Berlin: Springer.

Graves, A., Wayne, G., & Danihelka, I. (2014). *Neural Turing machines*. arXiv:1410.5401v2.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabskabarwińska, A., & Agapiou, J. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471–476.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5), 602–610.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.

Guo, Y., Liu, Y., Georgiou, T., & Lew, M. S. (2017). A review of semantic segmentation using deep neural networks. *International Journal of Multimedia Information Retrieval*, *2*, 1–7.

Han, X., Wu, Z., Jiang, Y. G., & Davis, L. S. (2017). Learning fashion compatibility with bidirectional LSTMs. In *Proceedings of the 2017 ACM on Multimedia Conference*. New York: ACM.

He, T., & Droppo, J. (2016). Exploiting LSTM structure in deep neural networks for speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 5445–5449). Piscataway, NJ: IEEE.

Heck, J. C., & Salem, F. M. (2017). Simplified minimal gated unit variations for recurrent neural networks. In *Proceedings of the IEEE International Midwest Symposium on Circuits and Systems* (pp. 1593–1596). Piscataway, NJ: IEEE.

Hochreiter, S. (1991). *Untersuchungen zu dynamischen neuronalen Netzen*. PhD diss., Technische Universität München.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hsu, W. N., Zhang, Y., Lee, A., & Glass, J. (2016). Exploiting depth and highway connections in convolutional recurrent deep neural networks for speech recognition. *Cell*, *50*(1), 395–399.

Irie, K., Tüske, Z., Alkhouli, T., Schlüter, R., & Ney, H. (2016). LSTM, GRU, highway and a bit of attention: An empirical overview for language modeling in speech recognition. In *Proceedings of the INTERSPEECH* (pp. 3519–3523). Red Hook, NY: Curran.

Ivakhnenko, A. G. (1971). Polynomial theory of complex systems. *IEEE Transactions on Systems, Man and Cybernetics*, *4*, 364–378.

Ivakhnenko, A. G., & Lapa, V. G. (1965). *Cybernetic predicting devices*. Sacramento, CA: CCM Information Corporation.

Jing, L., Gulcehre, C., Peurifoy, J., Shen, Y., Tegmark, M., Soljačić, M., & Bengio, Y. (2017). *Gated orthogonal recurrent units: On learning to forget*. arXiv:1706.02761.

Jordan, M. (1986). Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Annual Conference of the Cognitive Science Society* (pp. 531–546). Piscataway, NJ: IEEE.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *Proceedings of the International Conference on International Conference on Machine Learning* (pp. 2342–2350). New York: ACM.

Kalchbrenner, N., Danihelka, I., & Graves, A. (2015). *Grid long short-term memory*. arXiv:1507.01526.

Karpathy, A., Johnson, J., & Li, F. F. (2015). *Visualizing and understanding recurrent networks*. arXiv:1506.02078.

Khan, S., & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265.

Kim, J., El-Khamy, M., & Lee, J. (2017). *Residual LSTM: Design of a deep recurrent architecture for distant speech recognition*. arXiv:1701.03360.

Koutnik, J., Greff, K., Gomez, F., & Schmidhuber, J. (2014). *A clockwork RNN*. arXiv:1402.3511v1.

Krause, B., Lu, L., Murray, I., & Renals, S. (2016). *Multiplicative LSTM for sequence modelling*. arXiv:1609.07959.

Krumm, J., & Horvitz, E. (2015). Eyewitness: Identifying local events via space-time signals in Twitter feeds. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. New York: ACM.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, B., & Sainath, T. N. (2017). Reducing the computational complexity of two-dimensional LSTMs. In *Proceedings of the INTERSPEECH* (pp. 964–968). Red Hook, NY: Curran.

Li, J., Luong, M. T., Dan, J., & Hovy, E. (2015). *When are tree structures necessary for deep learning of representations*? arXiv:1503.00185v5.

Li, J., Mohamed, A., Zweig, G., & Gong, Y. (2016a). Exploring multidimensional LSTMS for large vocabulary ASR. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 4940–4944). Piscataway, NJ: IEEE.

Li, J., Mohamed, A., Zweig, G., & Gong, Y. (2016b). LSTM time and frequency recurrence for automatic speech recognition. In *Proceedings of the Conference on Automatic Speech Recognition and Understanding* (pp. 187–191). Piscataway, NJ: IEEE.

Li, S., Li, W., Cook, C., Zhu, C., & Gao, Y. (2018). Independently recurrent neural network: Building a longer and deeper RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 5457–5466). Piscataway, NJ: IEEE.

Liang, X., Lin, L., Shen, X., Feng, J., Yan, S., & Xing, E. P. (2017). Interpretable structure-evolving LSTM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2171–2184). Piscataway, NJ: IEEE.

Liang, X., Liu, S., Shen, X., Yang, J., Liu, L., Dong, J., & Yan, S. (2015). Deep human parsing with active template regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(12), 2402.

Liang, X., Shen, X., Feng, J., Lin, L., & Yan, S. (2016). Semantic object parsing with graph LSTM. In *Proceedings of the European Conference on Computer Vision* (pp. 125–143). Berlin: Springer.

Liang, X., Shen, X., Xiang, D., Feng, J., Lin, L., & Yan, S. (2016). Semantic object parsing with local-global long short-term memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3185–3193). Piscataway, NJ: IEEE.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision* (pp. 740–755). Berlin: Springer.

Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015). *A critical review of recurrent neural networks for sequence learning*. arXiv:1506.000190.

Liu, P., Qiu, X., Chen, J., & Huang, X. (2016). Deep fusion LSTMs for text semantic matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 1034–1043). Stroudsburg, PA: ACL.

Liu, P., Qiu, X., & Huang, X. (2016). *Modelling interaction of sentence pair with coupled-LSTMs*. arXiv:1605.05573.

Liu, Q., Zhou, F., Hang, R., & Yuan, X. (2017). Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification. *Remote Sensing*, *9*(12), 1330–1339.

Mallinar, N., & Rosset, C. (2018). *Deep canonically correlated LSTMs*. arXiv:1801.05407.

McCarter, G., & Storkey, A. (2007). *Air freight image segmentation database*. http://homepages.inf.ed.ac.uk/amos/afreightdata.html

Miwa, M., & Bansal, M. (2016). *End-to-end relation extraction using LSTMs on sequences and tree structures*. arXiv:1601.00770.

Moniz, J. R. A., & Krueger, D. (2018). *Nested LSTMs*. arXiv:1801.10308.

Mozer, M. C., & Das, S. (1993). A connectionist symbol manipulator that discovers the structure of context-free languages. In S. J. Hanson, J. D. Cowan, & C. L. Giles (Eds.), *Advances in neural information processing systems*, *5* (pp. 863–870). Cambridge, MA: MIT Press.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the International Conference on International Conference on Machine Learning* (pp. 807–814). Madison, WI: Omnipress.

Neil, D., Pfeiffer, M., & Liu, S. C. (2016). Phased LSTM: Accelerating recurrent network training for long or event-based sequences. In D. D. Lee, U. von Luxburg, R. Garnett, M. Sugiyama, & I. Guyon (Eds.), *Advances in neural information processing systems*, *16* (pp. 3882–3890). Red Hook, NY: Curran.

Nie, Y., An, C., Huang, J., Yan, Z., & Han, Y. (2016). A bidirectional LSTM model for question title and body analysis in question answering. In *IEEE First International Conference on Data Science in Cyberspace* (pp. 307–311). Piscataway, NJ: IEEE.

Nina, O., & Rodriguez, A. (2016). Simplified LSTM unit and search space probability exploration for image description. In *Proceedings of the International*

Conference on Information, Communications and Signal Processing (pp. 1–5). Piscataway, NJ: IEEE.

Niu, Z., Zhou, M., Wang, L., Gao, X., & Hua, G. (2017). Hierarchical multimodal LSTM for dense visual-semantic embedding. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1899–1907). Piscataway, NJ: IEEE.

Oord, A. V. D., Kalchbrenner, N., & Kavukcuoglu, K. (2016). *Pixel recurrent neural networks*. arXiv:1601.06759.

Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., & Ward, R. (2015). Deep sentence embedding using the long short-term memory network: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio Speech and Language Processing*, *24*(4), 694–707.

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, *1*(2), 263–269.

Peng, Z., Zhang, R., Liang, X., Liu, X., & Lin, L. (2016). Geometric scene parsing with hierarchical LSTM. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 3439–3445). Palo Alto, CA: AAAI Press.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., & Lazebnik, S. (2017). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *International Journal of Computer Vision*, *123*(1), 74–93.

Pulver, A., & Lyu, S. (2017). LSTM with working memory. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 845–851). Piscataway, NJ: IEEE.

Qu, Z., Haghani, P., Weinstein, E., & Moreno, P. (2017). Syllable-based acoustic modeling with CTC-SMBR-LSTM. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop* (pp. 173–177). Piscataway, NJ: IEEE.

Rahman, L., Mohammed, N., & Azad, A. K. A. (2017). A new LSTM model by introducing biological cell state. In *Proceedings of the International Conference on Electrical Engineering and Information Communication Technology* (pp. 1–6). Piscataway, NJ: IEEE.

Ranzato, M. A., Szlam, A., Bruna, J., Mathieu, M., Collobert, R., & Chopra, S. (2014). *Video (language) modeling: A baseline for generative models of natural videos*. arXiv:1412.6604.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural Networks for image classification: A comprehensive review. *Neural Computation*, *29*(9), 1–10.

Robinson, A. J., & Fallside, F. (1987). *The utility driven dynamic error propagation network*. Cambridge: University of Cambridge Department of Engineering.

Sainath, T. N., Vinyals, O., Senior, A., & Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 4580–4584). Piscataway, NJ: IEEE.

Sak, H. I., Senior, A., & Beaufays, F. O. (2014). *Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition*. arXiv:1402.1128v1.

Saleh, K., Hossny, M., & Nahavandi, S. (2018). Intent prediction of vulnerable road users from motion trajectories using stacked LSTM network. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems* (pp. 327–332). Piscataway, NJ: IEEE.

Schlag, I., & Schmidhuber, J. (2017). Gated fast weights for on-the-fly neural program generation. Workshop on Meta-Learning, In *NIPS Metalearning Workshop*.

Schmidhuber, J. (1993). Reducing the ratio between learning complexity and number of time varying variables in fully recurrent nets. In *Proceedings of the International Conference on Neural Networks* (pp. 460–463). London: Springer.

Schmidhuber, J. (2012). *Self-delimiting neural networks*. arXiv:2012.

Schmidhuber, J., Wierstra, D., Gagliolo, M., & Gomez, F. (2007). Training recurrent networks by evolino. *Neural Computation*, *19*(3), 757–779.

Schneider, N., & Gavrila, D. M. (2013). Pedestrian path prediction with recursive Bayesian filters: A comparative study. In *Proceedings of the German Conference on Pattern Recognition* (pp. 174–183). Berlin: Springer.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Shabanian, S., Arpit, D., Trischler, A., & Bengio, Y. (2017). *Variational Bi-LSTMs*. arXiv:1711.05717.

Sharma, P., & Singh, A. (2017). Era of deep neural networks: A review. In *Proceedings of the 8th International Conference on Computing, Communication and Networking Technologies* (pp. 1–5). Piscataway, NJ: IEEE.

Shi, X., Chen, Z., Wang, H., Woo, W. C., Woo, W. C., & Woo, W. C. (2015). Convolutional LSTM Network: A machine learning approach for precipitation nowcasting. In C. Cortes, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 802–810). Red Hook, NY: Curran.

Song, J., Tang, S., Xiao, J., Wu, F., & Zhang, Z. (2016). LSTM-in-LSTM for generating long descriptions of images. *Computational Visual Media*, *2*(4), 1–10.

Sperduti, A., & Starita, A. (1997). Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, *8*(3), 714–735.

Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). *Highway networks*. arXiv:1505.00387.

Šter, B. (2013). Selective recurrent neural network. *Neural Processing Letters*, *38*(1), 1–15.

Sun, G. (1990). Connectionist pushdown automata that learn context-free grammars. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 577–580). Piscataway, NJ: IEEE.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems* (pp. 3104–3112). Red Hook, NY: Curran.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. *Computer Science*, *5*(1), 36.

Teng, Z., & Zhang, Y. (2016). *Bidirectional tree-structured LSTM with head lexicalization*. arXiv:1611.06788.

Thireou, T., & Reczko, M. (2007). Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *4*(3), 441–446.

Veeriah, V., Zhuang, N., & Qi, G. J. (2015). Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 4041–4049). Piscataway, NJ: IEEE.

Vohra, R., Goel, K., & Sahoo, J. K. (2015). Modeling temporal dependencies in data using a DBN-LSTM. In *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics* (pp. 1–4). Piscataway, NJ: IEEE.

Wang, J., & Yuille, A. (2015). Semantic part segmentation using compositional model combining shape and appearance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1788–1797). Piscataway, NJ: IEEE.

Wei, H., Zhou, H., Sankaranarayanan, J., Sengupta, S., & Samet, H. (2018). Residual convolutional LSTM for tweet count prediction. In *Companion of the Web Conference 2018* (pp. 1309–1316). Geneva: International World Wide Web Conferences Steering Committee.

Weiss, G., Goldberg, Y., & Yahav, E. (2018). *On the practical computational power of finite precision RNNs for language recognition*. arXiv:1805.04908.

Weng, J. J., Ahuja, N., & Huang, T. S. (1993, May). Learning recognition and segmentation of 3D objects from 2D images. In *Proceedings of the Fourth International Conference on Computer Vision* (pp. 121–128). Piscataway, NJ: IEEE.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, *1*(4), 339–356.

Williams, R. J. (1989). *Complexity of exact gradient computation algorithms for recurrent neural networks* (Technical Report NU-CCS-89-27). Boston: Northeastern University, College of Computer Science.

Wu, H., Zhang, J., & Zong, C. (2016). *An empirical exploration of skip connections for sequential tagging*. arXiv:1610.03167.

Xie, X., & Shi, Y. (2018). Long-term memory neural Turing machines. *Computer Science and Application*, *8*(1), 49–58.

Yamaguchi, K. (2012). Parsing clothing in fashion photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3570–3577). Piscataway, NJ: IEEE.

Yang, Y., Dong, J., Sun, X., Lima, E., Mu, Q., & Wang, X. (2017). A CFCC-LSTM model for sea surface temperature prediction. *IEEE Geoscience and Remote Sensing Letters*, *99*, 1–5.

Yao, K., Cohn, T., Vylomova, K., Duh, K., & Dyer, C. (2015). *Depth-gated LSTM*. arXiv:1508.03790v4.

Yu, B., Xu, Q., & Zhang, P. (2018). Question classification based on MAC-LSTM. In *Proceedings of the IEEE Third International Conference on Data Science in Cyberspace*. Piscataway, NJ: IEEE.

Zaremba, W., & Sutskever, I. (2014). *Learning to execute*. arXiv:1410.4615.

Zhang, J., Zheng, Y., & Qi, D. (2016). Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing* (pp. 1655–1661). Piscataway, NJ: IEEE.

Zhang, X., Lu, L., & Lapata, M. (2015). *Tree recurrent neural networks with application to language modeling*. arXiv:1511.0006v1.

Zhang, Y., Chen, G., Yu, D., Yao, K., Khudanpur, S., & Glass, J. (2015). Highway long short-term memory RNNs for distant speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 5755–5759). Piscataway, NJ: IEEE.

Zhao, R., Wang, J., Yan, R., & Mao, K. (2016). Machine health monitoring with LSTM networks. In *Proceedings of the 10th International Conference on Sensing Technology* (pp. 1–6). Piscataway, NJ: IEEE.

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. M. (2016). A C-LSTM Neural network for text classification. *Computer Science*, *1*(4), 39–44.

Zhou, G., Wu, J., Zhang, C., & Zhou, Z. (2016). Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, *13*(3), 226–234.

Zhu, G., Zhang, L., Shen, P., & Song, J. (2017). Multimodal gesture recognition using 3D convolution and convolutional LSTM. *IEEE Access*, *5*, 4517–4524.

Zhu, X., Sobhani, P., & Guo, H. (2015). *Long short-term memory over tree structures*. arXiv:1503.04881.

Received November 24, 2018; accepted February 26, 2019.