frontiers in Genetics

ORIGINAL RESEARCH
published: 13 October 2021
doi: 10.3389/fgene.2021.746181

SS-RNN: A Strengthened Skip Algorithm for Data Classification Based on Recurrent Neural Networks

Wenjie Cao[1,2], *Ya-Zhou Shi*[1], *Huahai Qiu*[1] *and Bengong Zhang*[1]*

[1]*Research Center of Nonlinear Science, School of Mathematical and Physical Sciences, Wuhan Textile University, Wuhan, China,* [2]*School of Computer Science and Artificial Intelligence, Wuhan Textile University, Wuhan, China*

Recurrent neural networks are widely used in time series prediction and classification. However, they have problems such as insufficient memory ability and difficulty in gradient back propagation. To solve these problems, this paper proposes a new algorithm called SS-RNN, which directly uses multiple historical information to predict the current time information. It can enhance the long-term memory ability. At the same time, for the time direction, it can improve the correlation of states at different moments. To include the historical information, we design two different processing methods for the SS-RNN in continuous and discontinuous ways, respectively. For each method, there are two ways for historical information addition: 1) direct addition and 2) adding weight weighting and function mapping to activation function. It provides six pathways so as to fully and deeply explore the effect and influence of historical information on the RNNs. By comparing the average accuracy of real datasets with long short-term memory, Bi-LSTM, gated recurrent units, and MCNN and calculating the main indexes (Accuracy, Precision, Recall, and F1-score), it can be observed that our method can improve the average accuracy and optimize the structure of the recurrent neural network and effectively solve the problems of exploding and vanishing gradients.

Keywords: RNN, LSTM, SS-RNN, data classification, deep learning

OPEN ACCESS

Edited by:
Robert Friedman,
Retired, Columbia, SC, United States

Reviewed by:
Huang Yu-an,
Shenzhen University, China
Hong Peng,
South China University of Technology, China

****Correspondence:***
Bengong Zhang
bgzhang@wtu.edu.cn

Specialty section:
This article was submitted to Computational Genomics, a section of the journal Frontiers in Genetics

Received: *23 July 2021*
Accepted: *14 September 2021*
Published: *13 October 2021*

Citation:
Cao W, Shi Y-Z, Qiu H and Zhang B (2021) SS-RNN: A Strengthened Skip Algorithm for Data Classification Based on Recurrent Neural Networks. Front. Genet. 12:746181. doi: 10.3389/fgene.2021.746181

INTRODUCTION

Data classification is one of the most important tasks for different applications, such as text categorization, tone recognition, image classification, microarray gene expression, and protein structure prediction (Choi et al., 2017; Johnson and Zhang, 2017; Malhotra et al., 2017; Aggarwal et al., 2018; Fang et al., 2018; Mikołajczyk and Grochowski, 2018; Kerkeni et al., 2019; Saritas and Yasar, 2019; Yildirim et al., 2019; Chandrasekar et al., 2020). Many types of information (e.g., language, music, and gene) can be represented as sequential data that often contains related information separated by many time steps, and these long-term dependencies are difficult to model as we must retain information from the whole sequence with greater complexity of the model (Trinh et al., 2018; Liu et al., 2019; Shewalkar, 2019; Yu et al., 2019; Zhao et al., 2020).

With the rapid development of artificial intelligence and machine learning, the recurrent neural network (RNN) models have been gaining interest as a statistical tool for dealing with the complexities of sequential data (Chung et al., 2015; Keren and Schuller, 2016; Sadeghian et al., 2019; Yang et al., 2019). In RNNs, the recurrent layers or hidden layers consist of recurrent cells, and whose states are affected by both past states and current input with feedback connections (Yu et al., 2019). However, the errors signal back-propagated through time often suffer from exponential growth or decay, a dilemma commonly referred to as exploding or vanishing gradient. To alleviate this issue, the variants of RNNs with gating mechanisms, such as long short-term memory (LSTM) networks and gated recurrent units (GRU), have been proposed. LSTMs have been shown to learn many difficult sequential tasks effectively, including speech recognition, machine translation, trajectory prediction, and correlation analysis (Elman, 1990; Jordan, 1990; Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997; Cho et al., 2014; Alahi et al., 2016; Zhou et al., 2016; Su et al., 2017; Gupta et al., 2018; Hasan et al., 2018; Li and Cao, 2018; Salman et al., 2018; Vemula et al., 2018; Xu et al., 2018; Yang et al., 2019). In LSTMs, the information from the past can be stored within a hidden state that is combined with the latest input at each time step, allowing long-term dependencies to be captured. In spite of this, LSTMs are unable to capture the history information far from the current time step, given that the hidden state tends to focus on the more recent past, a finding proven by Zhao et al. (2020) along with a statistical perspective.

To address this problem, several improved RNNs have been proposed (Arpit et al., 2018; ElSaid et al., 2018; Abbasvandi and Nasrabadi, 2019; Ororbia et al., 2019). For example, Gui et al. (2019) introduced a novel reinforcement learning-based method to model the dependency relationship between words by computing the recurrent transition functions based on the skip connections. Inspired by the attention mechanism, Ostmeyer and Cowell (2019) developed a new kind of RNN model by calculating a recurrent weighted average (RWA) over every past processing step (not just the preceding step) to capture long-term dependencies, which performs far better than an LSTM on several challenging tasks. Based on the RWA, Maginnis and Richemond (2017) further presented a recurrent discounted attention (RDA) model by allowing it to discount the attention applied to previous time steps in order to carry out tasks requiring equal weighting over all information seen or tasks in which new information is more important than old. Later, DiPietro et al. (2017) introduced a mixed history RNN (MIST RNN) model, a NARX (nonlinear auto-regressive with extra inputs) RNN architecture that allows direct connections from the very distant past, and showed that MIST RNNs can improve performance substantially over LSTM on tasks requiring very long-term dependencies. In addition, Zhao et al. (2020) proposed the long memory filter that can be viewed as a soft attention mechanism, and proved that long-term memory can be acquired by using long memory filter. Very recently, Ma et al. (2021) proposed an end-to-end time series classification architecture called Echo Memory-Augmented Network (EMAN), and which uses a learnable sparse attention mechanism to capture important historical information and incorporate it into the feature representation of the current time step. However, how to well balance the accuracy and efficiency by adding past time information is still difficult to solve.

In this work, we propose a new algorithm called Strengthened Skip RNN (SS-RNN) to enhance the long-term memory ability by using multiple historical information to predict the next time information. To explore the effective method for the addition of historical information, we design six models for SS-RNN to include the past information into the current moment in continuous and discontinuous ways, respectively. For each way, the additional historical information can be directly added or added by weight weighting and function mapping. To test the SS-RNN with different models, five groups of datasets (Arrhythmia dataset, Epilepsy dataset 1, Epilepsy dataset 2, Breast cancer dataset, and Diabetes dataset) were used, and we also calculated these indexes to show the classification efficiency of our model: accuracy, precision, recall, and F1-score. From the results in *Results*, it is observed that Model A with *skip* = 3 has the greatest influence on the network. The important thing is that our SS-RNN method can effectively solve the problems of exploding gradient and vanishing gradient (Gers et al., 2000; Song et al., 2018; Tao et al., 2019; Das et al., 2020; Mayet et al., 2020).

THEORETICAL MODEL ANALYSIS AND DATA COLLECTION

SS-RNN Model Analysis

As for RNNs, the classical LSTM cell is proposed to deal with the problem of "long-term dependencies" by introducing a "gate" into the cell to improve the remembering capacity of the standard recurrent cell.

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

where W_{fh}, W_{fx}, W_{ih}, W_{ix}, $W_{\widetilde{c}h}$, $W_{\widetilde{c}x}$, W_{oh}, and W_{ox} are weight matrices and b_f, b_i, $b_{\widetilde{c}}$, and b_o are biases of LSTM to be learned during training. The above variables can parameterize the transformations of the input gate i_t, forget gate f_t, and output gate o_t, respectively.σ in **Eq. 1** is the sigmoid function and $\cdot$ stands for element-wise multiplication. c_t denotes the cell state of LSTM. x_t includes the inputs of LSTM cell unit, and h_t is the hidden layer (Wang et al., 2016; Kong et al., 2017; Yu et al., 2019). One can find the mathematical models of the RNN and GRU in the **Supplementary Material**.

Based on the LSTM model, we propose our SS-RNN model, which better utilizes historical information and could enhance the long-term memory of the model. The architecture of the SS-RNN model is shown in **Figure 1**. It consists of a feature extractor and a three-layer strengthened skip LSTM (SS-LSTM) network (**Figure 1A**). The feature extractor is added here to process the datasets with multiple features (not time series data) like the Diabetes data and Breast cancer data used in this paper. It extracts the features of multiple feature data. Then, the output of the feature extractor is reshaped to a matrix of 32*4 for further input into the SS-LSTM network (refer to **Supplementary Figure S55** and **Supplementary Material**). For standard time series datasets, such as Arrhythmia dataset, Epilepsy dataset 1, and Epilepsy dataset 2 used in this paper, we input them to SS-LSTM directly for training. **Figure 1B** shows the structure of a neuron in

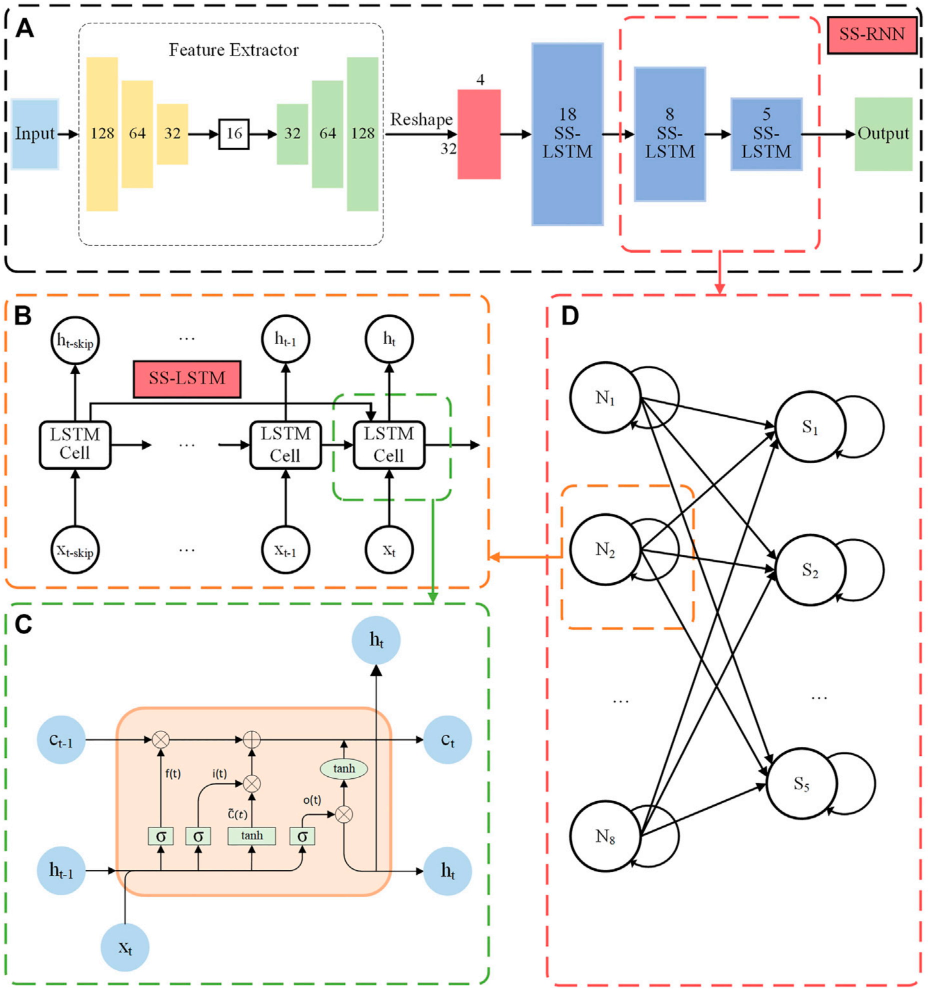

FIGURE 1 | (A) The architecture of the SS-RNN model for data classification. **(B)** The structure of a neuron in the second SS-LSTM layer with the information of moment *t-skip* used to strengthen the long memory at the moment *t*. **(C)** The internal schematic diagram of an LSTM cell. **(D)** The structure of the second layer and the third layer of the SS-LSTM network.

the second layer SS-LSTM, and the information at moment of *t-skip* (*skip* is positive integer) is used to strengthen the memory of the moment *t*.

In comparison with the LSTM model, by adding the information from time $t-1$, the information from the time of *t-skip* is also involved in the input at current time *t* (i.e.,x_t). So, the SS-RNN mathematical model can be written as follows:

$$\begin{cases} f_t = \sigma\left(W_{fskip}h_{t-skip} + W_{fh}h_{t-1} + W_{fx}x_t + b_f\right) \\ i_t = \sigma\left(W_{iskip}h_{t-skip} + W_{ih}h_{t-1} + W_{ix}x_t + b_i\right) \\ \widetilde{c}_t = \tanh\left(W_{\widetilde{c}skip}h_{t-skip} + W_{\widetilde{c}h}h_{t-1} + W_{\widetilde{c}x}x_t + b_{\widetilde{c}}\right) \\ c_t = f_t \cdot c_{t-1} + i_t \cdot \widetilde{c}_t \\ o_t = \sigma\left(W_{oskip}h_{t-skip} + W_{oh}h_{t-1} + W_{ox}x_t + b_o\right) \\ h_t = o_t \cdot \tanh(c_t) \end{cases} \tag{2}$$

where W_{fskip}, W_{iskip}, $W_{\widetilde{c}skip}$, and W_{oskip} are weight matrices for the corresponding inputs of the network activation functions, and h_{t-skip} is the output of the moment *t-skip*.

Obviously, from the above model, there are two important issues to address: 1) information of which historical moments should be involved into the current moment? 2) how should the past information be involved into the current moment? To answer these two questions, we enumerated all the methods to add the historical information to the current recurrent unit. These methods can be divided into continuous addition and discontinuous addition. The last information input consists of adding directly and weight weighting and function mapping for calculation. There are in total six models (Models A–F used in this work) for the addition of historical information, shown in **Figure 2**, and the detailed descriptions can be seen below (also, refer to the **Supplementary Material**).

Model A The information of historical moments (*t-skip*) is directly added to the current moment (*t*) and the method is discontinuous (**Figure 2A**). The mathematical expressions of the LSTM cell can be written as follows:

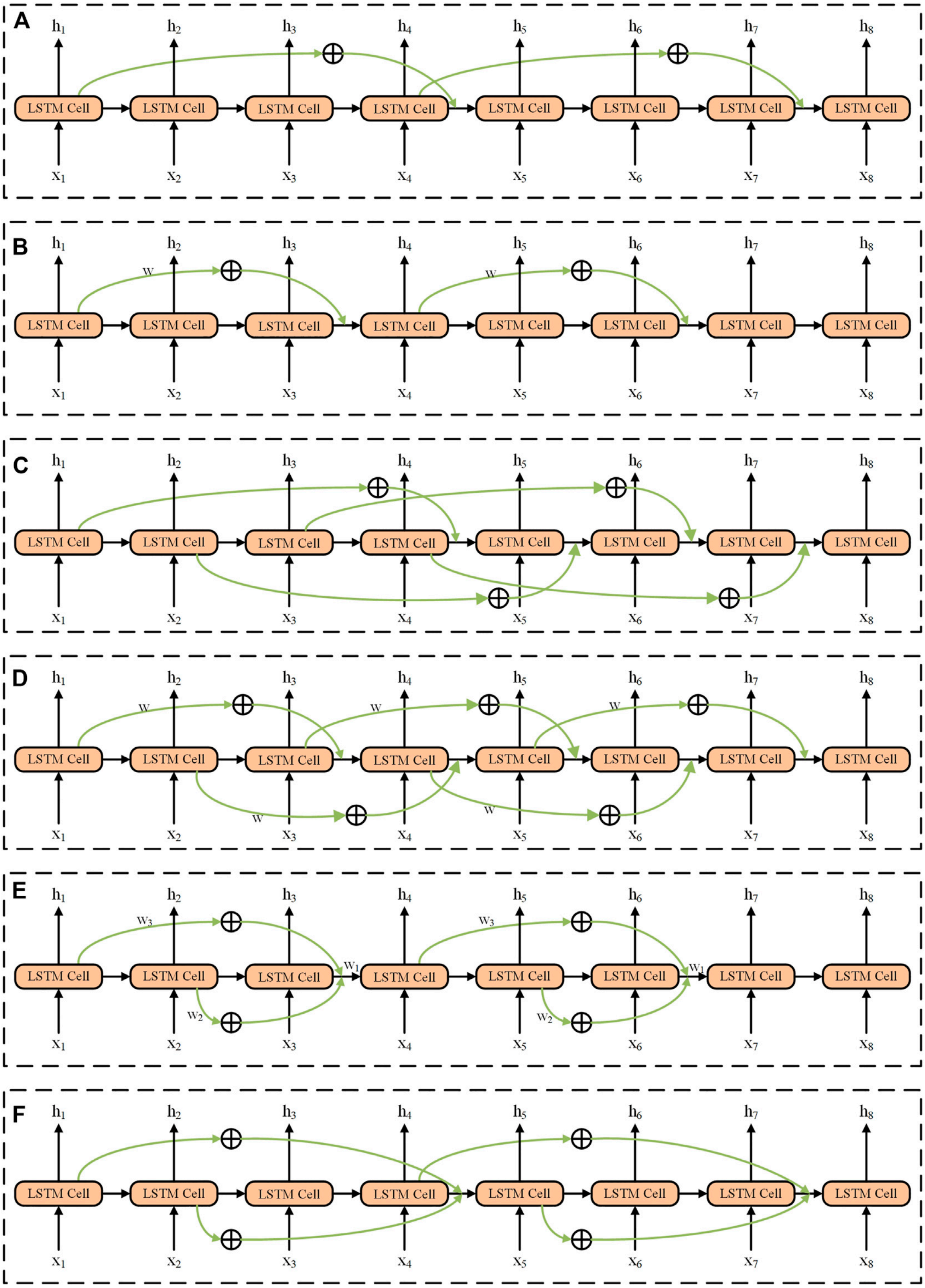
A
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
B
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
w
w
LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
C
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
D
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
w
w
w
LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell
w
w
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
E
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
w_3
w_3
w_1
w_1
LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell
w_2
w_2
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
F
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8
LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell LSTM Cell
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8

FIGURE 2 | Structures of six models (e.g., *skip* = 3) used in the SS-RNN. **(A)** Model A, the method is discontinuous addition without weight weighting and function mapping. **(B)** Model B, the method is discontinuous addition with weight weighting and function mapping. **(C)** Model C, the method is continuous addition without weight weighting and function mapping. **(D)** Model D, the method is continuous addition with weight weighting and function mapping. **(E)** Model E, add all the information of the time by corresponding skip before; the method is discontinuous addition with weight weighting and function mapping. **(F)** Model F, add all the information of the time by corresponding skip before; the method is discontinuous addition without weight weighting and function mapping.

$$\begin{cases} f_t = \sigma\left(W_{fh}h_{t-1} + W_{fx}x_t + b_f\right) \\ i_t = \sigma\left(W_{ih}h_{t-1} + W_{ix}x_t + b_i\right) \\ \tilde{c}_t = \tanh\left(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}\right) \\ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ o_t = \sigma\left(W_{oh}h_{t-1} + W_{ox}x_t + b_o\right) \\ \boldsymbol{N = o_t \cdot \tanh(c_t)} \\ \boldsymbol{h_t} = \begin{cases} \boldsymbol{N + h_{t-skip}}, & \boldsymbol{if\ t = 1 + i \times skip} \\ \boldsymbol{N}, & \boldsymbol{if\ t \neq 1 + i \times skip} \end{cases} \end{cases} \tag{3}$$

where *skip* is the order and $i \in \mathbf{N+}$ (**N+** is the set of positive integers); the part marked in bold indicates that the original formula has been changed. The order of Model A in **Figure 2A** is 3. For example, as shown in **Figure 2A**, when $t = 4$ with $skip = 3$, the input of recurrent unit h_4 comes from h_1, h_3 and x_4, and h_1 is directly added to the original output of h_4 to form a new output of h_4. Every three moments, additional historical information is added to the current moment.

Model B Similar to Model A, but the past information is added to the current moment after the transformation of the activation function by a weight of W_n (**Figure 2B**). The corresponding mathematical expressions can be rewritten as:

$$\begin{cases} M = W_{fh}h_{t-1} + \boldsymbol{W_{fskip} \cdot h_{t-skip}} \\ f_t = \begin{cases} \sigma\left(W_{fx}x_t + b_f + M\right), & \boldsymbol{if\ t = 1 + i \times skip} \\ \sigma\left(W_{fh}h_{t-1} + W_{fx}x + b_f\right), & \boldsymbol{if\ t \neq 1 + i \times skip} \end{cases} \\ N = W_{ih}h_{t-1} + \boldsymbol{W_{iskip} \cdot h_{t-skip}} \\ i_t = \begin{cases} \sigma\left(W_{ix}x_t + b_i + N\right), & \boldsymbol{if\ t = 1 + i \times skip} \\ \sigma\left(W_{ih}h_{t-1} + W_{ix}x_t + b_i\right), & \boldsymbol{if\ t \neq 1 + i \times skip} \end{cases} \\ Q = W_{\tilde{c}h}h_{t-1} + \boldsymbol{W_{\tilde{c}skip} \cdot h_{t-skip}} \\ \tilde{c}_t = \begin{cases} tanh\left(W_{\tilde{c}x}x_t + b_{\tilde{c}} + Q\right), & \boldsymbol{if\ t = 1 + i \times skip} \\ tanh\left(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}\right), & \boldsymbol{if\ t \neq 1 + i \times skip} \end{cases} \\ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ R = W_{oh}h_{t-1} + \boldsymbol{W_{oskip} \cdot h_{t-skip}} \\ o_t = \begin{cases} \sigma\left(W_{ox}x_t + b_o + R\right), & \boldsymbol{if\ t = 1 + i \times skip} \\ \sigma\left(W_{oh}h_{t-1} + W_{ox}x_t + b_o\right), & \boldsymbol{if\ t \neq 1 + i \times skip} \end{cases} \\ h_t = o_t \cdot tanh(c_t) \end{cases} \tag{4}$$

When $t = 4$, the input of loop unit h_4 comes from h_1, $h_{3,}$ and x_4. After h_1 is weighted, the function is transformed to add it to the current moment and form the output of new h_4.

Model C It continuously adds additional historical information to the current moment in a direct addition (**Figure 2C**). The corresponding mathematical expressions can be rewritten as:

$$h_t = o_t \cdot tanh(c_t) + \boldsymbol{h_{t-skip}} \tag{5}$$

The parts in bold represent changes to the original formula. The other part is the basic formula of LSTM. For example, when $t = 4$, the input of loop unit h_4 comes from h_1, h_3, and x_4, and h_1 is directly added to the current moment to form the output of new h_4. Model C can be regarded as the general form of Model A. In both models, the additional historical information is calculated in the same way. Model A adds historical information intermittently, and Model C adds historical information continuously where every current moment adds the historical information of the moment of *t-skip*, and which leads to a greater computational complexity for the model.

Model D It continuously adds historical information to the current moment in the form of weight weighting and function mapping (**Figure 2D**). The corresponding mathematical expressions can be rewritten as:

$$\begin{cases} f_t = \sigma\left(W_{fh}h_{t-1} + W_{fx}x_t + \boldsymbol{W_{fskip} \cdot h_{t-skip}} + b_f\right) \\ i_t = \sigma\left(W_{ih}h_{t-1} + W_{ix}x_t + \boldsymbol{W_{iskip} \cdot h_{t-skip}} + b_i\right) \\ \tilde{c}_t = \tanh\left(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + \boldsymbol{W_{\tilde{c}skip} \cdot h_{t-skip}} + b_{\tilde{c}}\right) \\ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ o_t = \sigma\left(W_{oh}h_{t-1} + W_{ox}x_t + \boldsymbol{W_{oskip} \cdot h_{t-skip}} + b_o\right) \\ h_t = o_t \cdot \tanh(c_t) \end{cases} \tag{6}$$

When $t = 4$, the input of loop unit h_4 comes from h_1, h_3, and x_4, and h_1 is directly added to the current moment to form the output of new h_4. Model D can be regarded as the general form of Model B. In Model B and Model D, additional historical information is calculated in the same way, Model B adds historical information intermittently, and Model D adds historical information continuously.

Model E It intermittently adds additional historical information to the current moment in the form of weight weighting and function mapping (**Figure 2E**). The corresponding mathematical expressions can be rewritten as:

$$\begin{cases} \boldsymbol{M = W_{f1}h_{t-1} + W_{f2}h_{t-2} + W_{f3}h_{t-3}} \\ f_t = \sigma\left(W_{fx}x_t + b_f + \boldsymbol{M}\right) \\ \boldsymbol{N = W_{i1}h_{t-1} + W_{i2}h_{t-2} + W_{i3}h_{t-3}} \\ i_t = \sigma\left(W_{ix}x_t + b_i + \boldsymbol{N}\right) \\ \boldsymbol{Q = W_{\tilde{c}1}h_{t-1} + W_{\tilde{c}2}h_{t-2} + W_{\tilde{c}3}h_{t-3}} \\ \tilde{c}_t = \tanh\left(W_{\tilde{c}x}x_t + b_{\tilde{c}} + \boldsymbol{Q}\right) \\ c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ \boldsymbol{R = W_{o1}h_{t-1} + W_{o2}h_{t-2} + W_{o3}h_{t-3}} \\ o_t = \sigma\left(W_{ox}x_t + b_o + \boldsymbol{R}\right) \\ h_t = o_t \cdot \tanh(c_t) \end{cases} \tag{7}$$

When $t = 4$, the input of loop unit h_4 comes from h_1, h_2, h_3, and x_4, and h_1 and h_2 are added to the current moment through weight weighting and function mapping and constitutes the output of new h_4.

Model F It intermittently adds historical information to the current moment, and the historical information directly adds to the current moment (**Figure 2F**). The corresponding mathematical expressions after the improvement of LSTM can be rewritten as:

$$\begin{cases} \boldsymbol{N = o_t \cdot tanh(c_t)} \\ \boldsymbol{h_t} = \begin{cases} \boldsymbol{N + \sum_{s=2}^{skip} h_{t-s}}, & \boldsymbol{if\ t = 1 + i \times skip} \\ \boldsymbol{N}, & \boldsymbol{if\ t \neq 1 + i \times skip} \end{cases} \end{cases} \tag{8}$$

Data Collection

To test the effect of long-term memory introduced in this work on data classification, we first conduct experiments on three time series datasets (i.e., Arrhythmia dataset, Epilepsy dataset 1, and

TABLE 1 | Description of five datasets used in this work for data classification.

Datasets	Source	Size	Train[a]	Test[a]	Classes[b]	Sources
Arrhythmia dataset	MIT-BIH Arrhythmia Database	109,338	87,470	21,868	5	https://www.physionet.org/content/mitdb/1.0.0/
Epilepsy dataset 1	Epileptologie Bonn	11,500	9,200	2,300	5	https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition
Epilepsy dataset 2	CHB-MIT Scalp EEG Database	361,377	289,102	72,275	2	https://physionet.org/content/chbmit/1.0.0/
Diabetes dataset	UC Irvine Machine Learning Repository	520	416	104	2	http://archive.ics.uci.edu/ml/datasets/Early+stage+diabetes+risk+prediction+dataset
Breast cancer dataset	UC Irvine Machine Learning Repository	116	93	23	2	https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Coimbra

[a] *Sizes of the training and testing sets for the five datasets, respectively.*

[b] *Number of classes of five datasets.*

Epilepsy dataset 2). In addition, due to the potential correlations between the characteristics in some non-time-series biomedical data, we also perform experiments on two disease datasets: Diabetes dataset and Breast cancer dataset, to validate the ability of the model on non-time series data classification. Each dataset was split into training and testing set using the standard split. **Table 1** summarizes the details of the five datasets.

Arrhythmia dataset It contains 109,338 recordings of 48 half-hour excerpts of two-channel ambulatory ECG, and which have been divided into five classes based on the heart rate: one normal and four abnormal.

Epilepsy datasets There are two well-known Epilepsy datasets used in this work. One is from the Department of Epilepsy at the University of Bonn, Germany, and which contains five categories (A–E) of 100 single-channel 23.6-s segments of electroencephalogram (EEG) signals (11,500 in total). The other is from Children's Hospital Boston including 361,377 EEG recordings from 22 epileptic patients and these recordings have been grouped into two classes.

Diabetes dataset It contains 16 features, such as age, sex, and polyuria, and the source is from the University of California at Irvine Machine Learning Repository. This has been collected using direct questionnaires from the patients at Sylhet Diabetes Hospital in Sylhet, Bangladesh, and approved by a medical doctor.

Breast cancer dataset It contains nine features from UC Irvine Machine Learning Repository (see **Supplementary Material**).

The original five datasets are available through the websites listed in **Table 1**, and we also rearranged them for the convenience of use, and which can be found in the **Supplementary Material**.

Evaluation Index

For the classification task, the models are evaluated by the classification accuracy, precision, recall, and F1-score, which are defined by the confusion matrix. It is one of the most intuitive metrics used for evaluating the performance and accuracy of the model in machine learning, especially used for classification problems. The terms associated with confusion matrix can be defined as follows: True positives (TP), when the actual class of the data point is 1 and the predicted outcome is also 1. True negatives (TN) are the cases when the actual class of the given data point is 0 and the predicted result is also 0. False positives (FP) are the cases when the actual class of the data point is 0 and the predicted outcome is 1, which can be assumed that the model predicts incorrectly as the actual class is positive. False negatives (FN) are the cases when the actual class should be 1 and the predicted outcome is 0, where the model predicts incorrectly as negative. The forms are expressed as follows:

$$\begin{cases} Accuracy = \dfrac{TP+TN}{TP+FP+FN+TN} \\ Precision = \dfrac{TP}{TP+FP} \\ Recall = \dfrac{TP}{TP+FN} \\ F1-score = \dfrac{2\times Precision\times Recall}{Precision+Recall} \end{cases} \tag{9}$$

RESULTS

The Workflow of the SS-RNN

In SS-RNN, the information of historical moments (e.g., *t-skip*) can be added to the current moment (i.e., *t*) to accurately classify sequential data with long-term dependences. To determine the best methods of the past information addition and verify the effectiveness of the SS-RNN model, we did six groups of comparison experiments on five datasets, respectively. The six different models (Models A–F) and five datasets are shown in *Theoretical Model Analysis and Data Collection*. For each experiment, there are three steps: data preprocessing, training, and test.

Data preprocessing Outliers and missing values often appear in the dataset, whereas the network model cannot process those data samples. We first fill the missing values with the mean of the variable and delete the samples with outliers, which can be judged from the method of Anomaly Detection. The pre-processed time series datasets (e.g., Arrhythmia and Epilepsy datasets in **Table 1**) can be directly input into the SS-LSTM model. However, for the

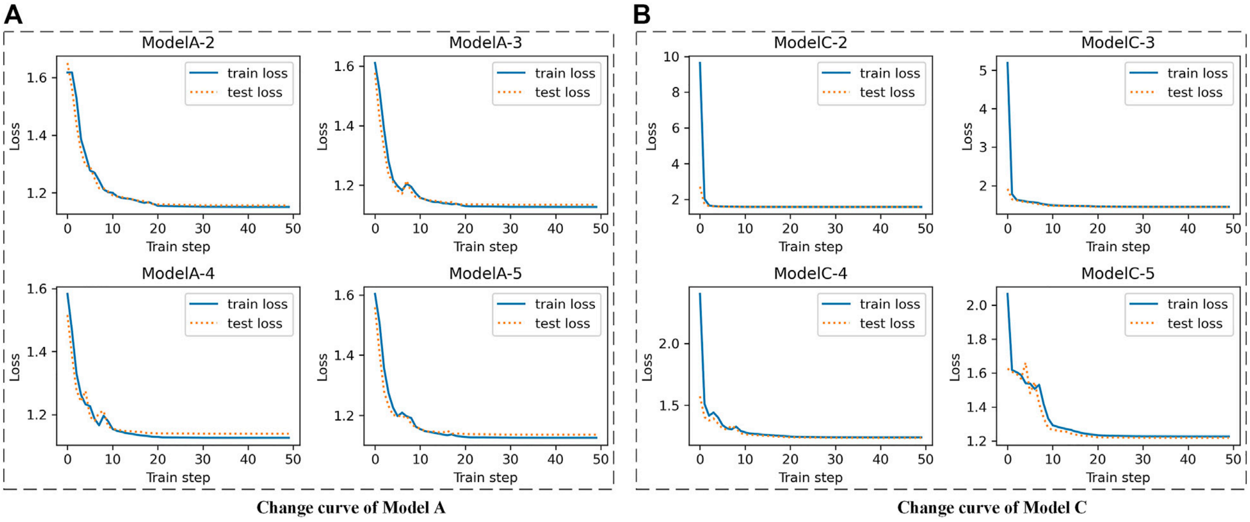

FIGURE 3 | The change curves of loss function between train set and test set with different skip value of Epilepsy dataset 1. e.g., Model A-2 is Model A **(A)** with *skip* = 2, Model C-4 is Model C **(B)** with *skip* = 4.

non-time series data with multiple features and different dimensions (e.g., Diabetes and Breast cancer datasets used in this work), after the above preprocessing, it needs to be fed into the feature extractor to obtain a new set of data and their characters, which can be further transformed to a matrix of 32*4 as input into the SS-LSTM. Taking the Diabetes dataset as an example, we also give detailed descriptions in the **Supplementary Material**.

Training For each dataset (**Table 1**), the training set is used to train the model. The optimized parameters of the network are as follows: dimensions of the network are 128, 64, 32, and 16, respectively (**Figure 1A**). For each dataset, the configuration of the SS-LSTM model is implemented in Pytorch using **Eqs 3–8**, and the dimensions for the three layers of the SS-LSTM model are 18, 8, and 5, respectively. The activation function is *tanh*, and the training algorithm is stochastic gradient descent with a learning rate of 0.01 and a training epoch of 50. Here, we used the cross-entropy loss as the objective function for training the network:

$$\mathrm{Loss} = -\sum_{i=1}^{n} y_i \log\left(\hat{y}_i\right) \tag{10}$$

where y_i is the true value, and $\hat{y}_i$ is corresponding predicted value. The batch size of each dataset after fine-tuning is shown in the **Supplementary Material**.

Test For each dataset, 25 different comparative experiments were performed using different structures of LSTM. One of the experiments adopted the ordinary LSTM, while the others used SS-LSTM with different models (i.e., Models A–F in **Figure 2**). For each model, the values of *skip* were set as 2, 3, 4 and 5, respectively. Furthermore, we also used the other classical models (LSTM, GRU and Bi-LSTM) to create the classification set for three of the datasets (i.e., Arrhythmia, Epilepsy 1 and Diabetes), and made a comparison with our SS-RNN model.

Testing the Models With Data

To test the effect of the addition of past information on the data classification, we used our network with six different SS-LSTM models (Models A–F; **Figure 2**) to classify the data for five datasets (**Table 1**), respectively. For each SS-LSTM model, different values of *skip* (e.g., *skip* = 2, 3, 4, 5) were used. As shown in **Supplementary Figures S1–S10, S12–S17, S19–S24, S26–S31** in the **Supplementary Material**, the loss functions calculated by **Eq. 10** for the experiments in this work always converged before 50 steps, indicating that 50 steps are sufficient for the training and test processes.

Epilepsy Dataset 1

For the Epilepsy dataset, 19,200 samples were used to train our SS-LSTM, which were further tested by the rest of the samples. The loss functions show that Models A and C are more stable than the others, and the loss value of the training set is consistent with the test set, indicating that no overfitting has occurred (**Figure 3**, **Supplementary Figures S1–S4**). As shown in **Figure 4**, the value of loss function of Model A is also the lowest among all models (**Figure 4A**), and the predicted accuracy of Model A is ~47%, which is not only higher than that (~37%) of the original LSTM, but also significantly better than those predicted by SS-LSTM with other models (e.g., ~40% of Model C with *skip* = 4, i.e., Model C-4). The results indicate that the past information (*t-skip*) directly added to the current moment (*t*) could effectively improve the classified accuracy on the Epilepsy dataset 1. However, Model C with *skip* = 2 has the lowest predicted accuracy (~24%), and which could suggest that Model C is not suitable for processing this dataset.

Diabetes Dataset

As shown in **Figure 5A**, the predicted accuracy of most SS-LSTM models is much higher than that (~61%) of the original LSTM

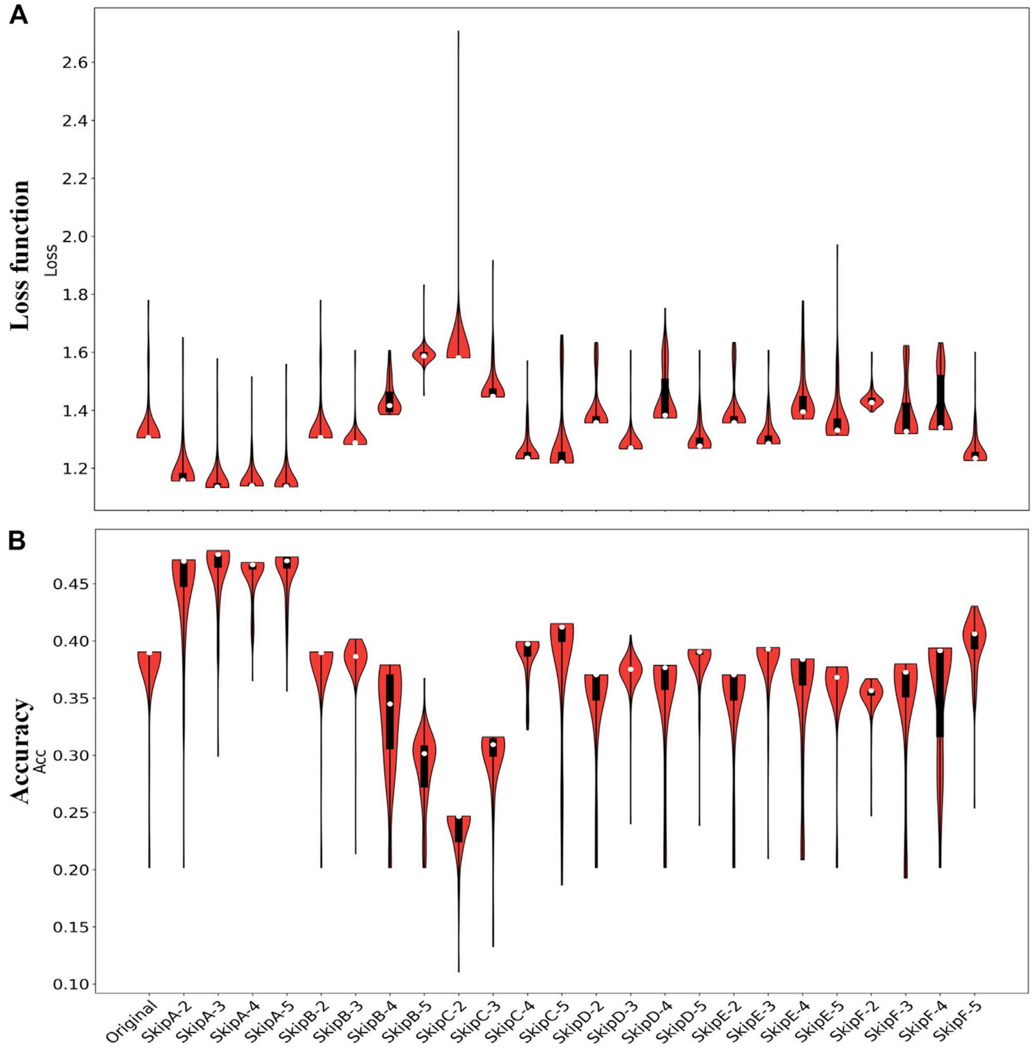

FIGURE 4 | The loss functions **(A)** and predicted accuracy **(B)** of each model for classification of Epilepsy dataset 1. Original represents the results from the original LSTM, and others represent the results from the SS-LSTM with different models, e.g., SkipA-2 is Model A with *skip* = 2 (**Figure 2**). For each violin in **(A, B)**, the top of the black rectangle is the three-quarter digit, the bottom is the quarter digit, the white dots are the mean, and the width of the orange area is the distribution of density.

model for the Diabetes dataset, and Model A with *skip* = 3 has the highest accuracy (~98%). The accuracy of Model B is significantly and positively correlated with the order. The change curve of the loss function of each model in the training process is also shown in the **Supplementary Material**.

Arrhythmia Dataset and Other Datasets

For the Arrhythmia dataset, the long-term memory in Model A can markedly improve the classification accuracy, e.g., the ACC increases from ~82 to 94%, as *skip* increasing from 2 to 5 (**Figure 5B**). Surprisingly, although the large values of *skip* can also be helpful for Model F, the ACC of Model F with any *skip* values is obviously lower than the original LSTM. Furthermore, in the other models (i.e., Model B, Model D, and Model E), no matter how the *skip* changes, the accuracy stays at the same level (~82%), which suggests that the addition of past information could be a burden for the RNN and has no positive effect on data classification (**Figure 5B**). The structure of Models B and D, which both have a common characteristic that adopts the same way of weight weighting and function mapping to put the historical information added to the current time damage the dynamic performance of the RNN. So, this is not an ideal method for the Arrhythmia dataset.

We also have experiments on Epilepsy dataset 2 and Breast cancer dataset, and the relevant results and analysis are shown in the **Supplementary Material**.

Comparison Results With Other Models

Furthermore, we also made classifications for three of the datasets (Arrhythmia dataset, Epilepsy dataset 1, and Diabetes dataset) by using the classical networks such as LSTM, GRU, and Bi-LSTM with default parameters of the torch.nn module, and compared the results with that from the SS-RNN with Model A of *skip* = 3

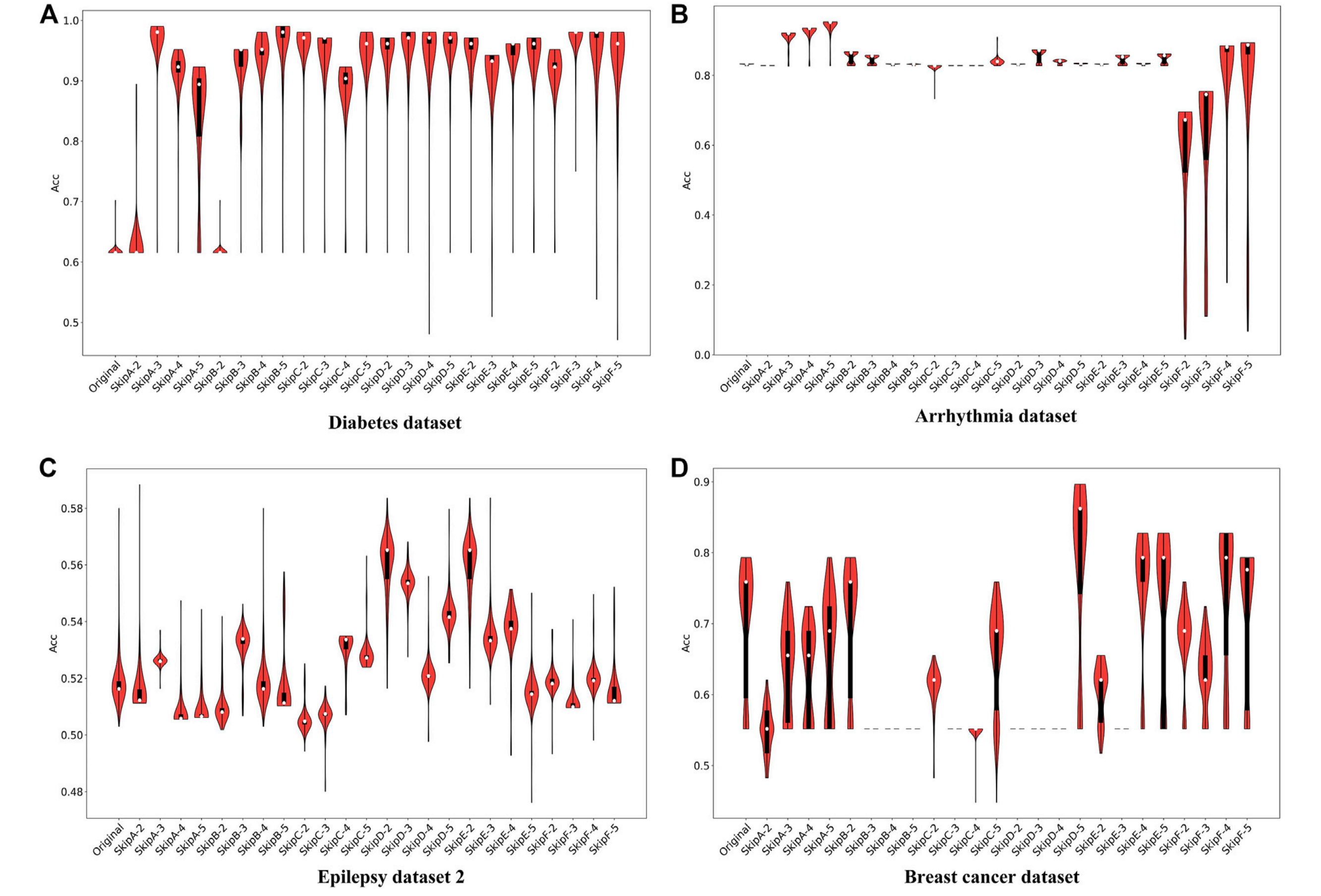

FIGURE 5 | The predicted accuracy of each model for classification on Diabetes dataset **(A)**, Arrhythmia dataset **(B)**, Epilepsy dataset 2 **(C)** and Breast cancer dataset **(D)**.

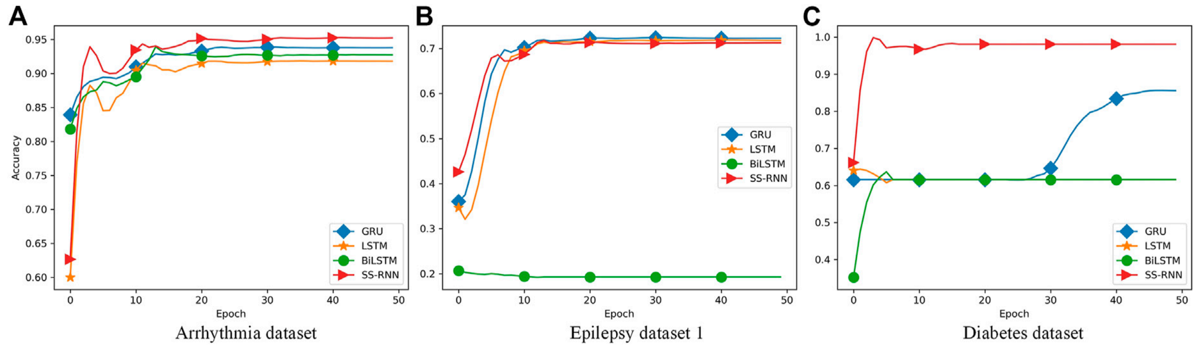

FIGURE 6 | Comparisons between LSTM, GRU, Bi-LSTM, and our SS-RNN (SkipA-3). **(A)** Accuracy of the Arrhythmia dataset. **(B)** Accuracy of the Epilepsy dataset 1. **(C)** Accuracy of the Diabetes dataset.

(**Figure 6**; **Tables 2–4**). We also show the simulation results of other indexes with Model A of *skip* = 3 and Model C of *skip* = 5 in the **Supplementary Material**.

From **Figure 6**, it shows that our SS-RNN method can improve the classification accuracy as compared to the classical methods. Also, from **Tables 2–4**, it can be found that the other main indexes are almost improved. At the same time, we compared our method with the latest methods RNN, RNN+GRU, RNN+LSTM, and MCNN (Zhang et al., 2017; Singh et al., 2018) with the same Arrhythmia dataset. The result is shown in **Table 5**,

TABLE 2 | Arrhythmia dataset classification comparison results with LSTM, GRU, and Bi-LSTM.

	Accuracy	Precision	Recall	F1-score
LSTM	0.9181	0.9564	0.9380	0.9316
GRU	0.9380	0.9660	0.9380	0.9479
Bi-LSTM	0.9274	0.9596	0.9274	0.9384
SS-RNN(SkipA-3)	0.9524	0.9670	0.9524	0.9573

TABLE 3 | Epilepsy dataset 1 classification comparison results with LSTM, GRU, and Bi-LSTM.

	Accuracy	Precision	Recall	F1-score
LSTM	0.7178	0.7190	0.7178	0.2506
GRU	0.7226	0.7240	0.7226	0.2540
Bi-LSTM	0.1926	0.0371	0.1926	0.3276
SS-RNN(SkipA-3)	0.7126	0.7115	0.7126	0.3834

TABLE 4 | Diabetes dataset classification comparison results with LSTM, GRU, and Bi-LSTM.

	Accuracy	Precision	Recall	F1-score
LSTM	0.6154	0.3787	0.6154	0.4689
GRU	0.8556	0.8832	0.8558	0.8467
Bi-LSTM	0.6154	0.3787	0.6154	0.4689
SS-RNN(SkipA-3)	0.9808	0.9817	0.9808	0.9809

TABLE 5 | Arrhythmia dataset classification comparison results with RNN, RNN+GRU, RNN+LSTM, and MCNN.

	Accuracy	Recall (Sensitivity)
RNN	0.8540	0.8060
RNN GRU	0.8250	0.7890
RNN LSTM	0.8810	0.9240
MCNN	0.9110	NA[a]
SS-RNN(SkipA-3)	0.9524	0.9524

[a] *NA means that it is not available in the original paper.*

which also indicates that our SS-RNN method can improve the classification accuracy.

In fact, as a variant of LSTM, GRU reduces the forget gate and input gate, and adds the update gate. GRU has simpler internal structure and less parameters than LSTM, which reduces the risk of overfitting. Although LSTM and GRU partially solve the problem of the vanishing gradient of the RNN, the information loss is still very severe in the propagation of a very long distance. Bi-LSTM, namely, bi-directional LSTM, does not change any internal structure of LSTM itself. LSTM is applied twice in different directions, and then the LSTM results obtained twice are spliced as the final output. For datasets with both forward and backward dependencies, this method can enhance the correlation between data and improve the efficiency of the model. It is often used to capture some specific pre or post features of language and syntax in natural language processing. However, in biological datasets like ECG and EEG, the progression and onset of diseases are irreversible, so the relationship between data in the reverse time direction is not of practical significance for disease classification. In addition, excessive number of parameters may lead to overfitting in network training, so the Bi-LSTM model is not suitable here.

The long-term memory ability of LSTM and GRU is weak, and with the increase of the time step, the farther away the memory, the more information the model forgot and the less it remembered. Our model has enhanced the information in distant moments, which makes up for the defect of long-term dependence in RNNs. Therefore, our SS-RNN method can improve the precision, recall, F1-score, and finally improve the classification accuracy of sequential data compared with other models.

DISCUSSION

The performance of the loss function is different between five datasets and six models. Model A has the best performance. In Epilepsy dataset 1, Model A has the lowest loss function and the highest accuracy of all models. In the Diabetes dataset, the loss function of Model A-3 is the lowest and the accuracy is the highest. In the Arrhythmia dataset, the performance of the loss function of each model is different, and Model A has the best performance, in which the loss function is negatively correlated with the order and the accuracy is positively correlated with the order. In Epilepsy dataset 2, overfitting occurred due to the convergence effect of each model. Therefore, Model A did not show good performance, Model C had the lowest loss function, and Model D-2 had the highest accuracy. As for the Breast cancer dataset, the training effect of the network is not optimal because the data scale is too small, and the average loss of Model D-5 is the lowest and the accuracy is the highest. There is a certain relationship between order and accuracy in each model.

Furthermore, we calculated the average accuracy of the six models by our method on the Arrhythmia dataset, Epilepsy dataset 1, and Diabetes dataset. Comparing the results with the original LSTM, GRU, and Bi-LSTM models, the average accuracy is improved. It is shown in **Figure 7**. It means that our SS-RNN method is generally useful. We also compared the average accuracy of the six models by our method with original LSTM with five groups of datasets. It can be found in **Supplementary Figure S54**. According to the results in **Figure 7**, it shows that Model A is the best with the highest average accuracy among the six ways of adding historical information. By comparing the differences of various adding methodologies, it can be found that the discontinuous adding method is better than the continuous adding method, while the direct adding method is better than the method of weight weighting and function mapping. It does not mean that more historical information is better. Adding more historical information did not improve the memory ability of the RNN. Different data have different dependence intensity, so the same model has different modeling performance for different datasets.

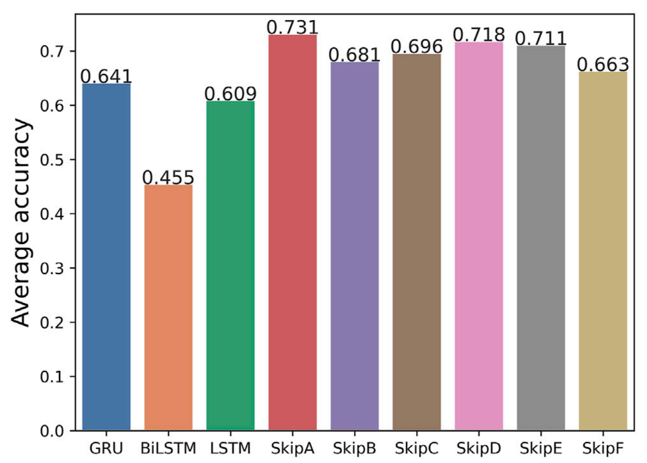

FIGURE 7 | Average accuracy of the Arrhythmia dataset, Epilepsy dataset 1, and Diabetes dataset between the original LSTM, GRU, and Bi-LSTM models and our six models without batch size tuned.

CONCLUSION

In order to effectively capture the long-term dependencies in sequential data, we propose the SS-RNN, which allows the historical information to add to the moment by different methods. We designed six models with different *skips* to simulate the possible patterns of the addition of past information, and tested them on five disease-related datasets with different sizes and data types. By comparing our method with the original LSTM, GRU, and Bi-LSTM and the recent methods RNN+GRU, RNN+LSTM, and MCNN, the simulation results suggest that our method can significantly improve the accuracy of sequential data classification. Furthermore, the best method to add the past information could be the method discontinuous addition without weight weighting and function mapping. It can effectively solve the problems of exploding gradient and vanishing gradient. There is a certain correlation between the model performance and the order.

The SS-RNN provides a new idea to improve the classification accuracy of sequential data by optimizing the LSTM model. Therefore, users can also optimize their own network model by adding the SS-RNN module, which is of great significance for the classification diagnosis and precision treatment of diseases. Although the SS-RNN generally has a good classification effect for large datasets, the performance of the model for small sample datasets needs to be further improved. In the future, few-shot learning could be further introduced to train the SS-RNN network to improve the classification efficiency for small sample datasets. The code of the SS-RNN model can be available through github (https://github.com/WTU-RCNS-Bioinformatics-Lab/SS-RNN).

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

BZ, WC, and YS designed the research. WC, HQ, and YS performed the experiments. WC, HQ, and BZ analyzed the data. WC, YS, and BZ wrote the manuscript. All authors discussed the results and reviewed the manuscript.

FUNDING

This work was supported by grants from the National Natural Science Foundation of China (No. 11971367).

ACKNOWLEDGMENTS

We are grateful to Professors Jian Zhang (Nanjing University) and Wenbing Zhang (Wuhan University) for valuable discussions.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fgene.2021.746181/full#supplementary-material

REFERENCES

Abbasvandi, Z., and Nasrabadi, A. M. (2019). A Self-Organized Recurrent Neural Network for Estimating the Effective Connectivity and its Application to EEG Data. *Comput. Biol. Med.* 110, 93–107. doi:10.1016/j.compbiomed.2019.05.012

Aggarwal, A., Singh, J., and Gupta, D. K. (2018). A Review of Different Text Categorization Techniques. *Int. J. Eng. Technol. (Ijet)* 7 (3.8), 11–15. doi:10.14419/ijet.v7i3.8.15210

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). "Social Lstm: Human Trajectory Prediction in Crowded Spaces," in 2016 Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), (Las Vegas, NV, USA: IEEE), 961–971. doi:10.1109/CVPR.2016.110

Arpit, D., Kanuparthi, B., Kerg, G., Ke, N. R., Mitliagkas, I., and Bengio, Y. (2018). H-Detach: Modifying the LSTM Gradient towards Better Optimization. arXiv [Preprint]. Available at: arXiv:1810.03023.

Chandrasekar, V., Sureshkumar, V., Kumar, T. S., and Shanmugapriya, S. (2020). Disease Prediction Based on Micro Array Classification Using Deep Learning Techniques. *Microprocessors and Microsystems* 77, 103189. doi:10.1016/j.micpro.2020.103189

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., et al. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. arXiv [Preprint]. Available at: https://arxiv.org/abs/1406.1078. doi:10.3115/v1/d14-1179

Choi, K., Fazekas, G., Sandler, M., and Cho, K. (2017). "Convolutional Recurrent Neural Networks for Music Classification," in 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (New Orleans, LA, USA: IEEE), 2392–2396. doi:10.1109/ICASSP.2017.7952585

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. (2015). A Recurrent Latent Variable Model for Sequential Data. *Adv. Neural Inf. Process. Syst.* 28, 2980–2988. Available at: https://proceedings.neurips.cc/paper/2015/file/b618c3210e934362ac261db280128c22-Paper.pdf.

Das, M., Pratama, M., Zhang, J., and Ong, Y. S. (2020). A Skip-Connected Evolving Recurrent Neural Network for Data Stream Classification under Label Latency Scenario. *Assoc. Adv. Artif. Intelligence* 34 (04), 3717–3724. doi:10.1609/aaai.v34i04.5781

DiPietro, R., Rupprecht, C., Navab, N., and Hager, G. D. (2017). Analyzing and Exploiting NARX Recurrent Neural Networks for Long-Term Dependencies. arXiv [Preprint]. Available at: arXiv:1702.07805.

Elman, J. L. (1990). Finding Structure in Time. *Cogn. Sci.* 14 (2), 179–211. doi:10.1207/s15516709cog1402_1

ElSaid, A., El Jamiy, F., Higgins, J., Wild, B., and Desell, T. (2018). Optimizing Long Short-Term Memory Recurrent Neural Networks Using Ant colony Optimization to Predict Turbine Engine Vibration. *Appl. Soft Comput.* 73, 969–991. doi:10.1016/j.asoc.2018.09.013

Fang, C., Shang, Y., and Xu, D. (2018). MUFOLD-SS: New Deep Inception-Inside-Inception Networks for Protein Secondary Structure Prediction. *Proteins* 86 (5), 592–598. doi:10.1002/prot.25487

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Comput.* 12 (10), 2451–2471. doi:10.1162/089976600300015015

Gui, T., Zhang, Q., Zhao, L., Lin, Y., Peng, M., Gong, J., et al. (2019). Long Short-Term Memory with Dynamic Skip Connections. *Assoc. Adv. Artif. Intelligence* 33 (01), 6481–6488. doi:10.1609/aaai.v33i01.33016481

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., and Alahi, A. (2018). "Social gan: Socially Acceptable Trajectories with Generative Adversarial Networks," in 2018 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (Salt Lake City, UT, USA: IEEE), 2255–2264. doi:10.1109/CVPR.2018.00240

Hasan, I., Setti, F., Tsesmelis, T., Del Bue, A., Galasso, F., and Cristani, M. (2018). "Mx-lstm: Mixing Tracklets and Vislets to Jointly Forecast Trajectories and Head Poses," in 2018 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (Salt Lake City, UT, USA: IEEE), 6067–6076. doi:10.1109/CVPR.2018.00635

Hochreiter, S., and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.* 9 (8), 1735–1780. doi:10.1162/neco.1997.9.8.1735

Johnson, R., and Zhang, T. (2017). Deep Pyramid Convolutional Neural Networks for Text Categorization. *Proc. 55th Annu. Meet. Assoc. Comput. Linguistics* 1, 562–570. doi:10.18653/v1/P17-1052

Jordan, M. I. (1990). "Attractor Dynamics and Parallelism in a Connectionist Sequential Machine," in Artificial neural networks: concept learning. 112–127. Available at: https://dl.acm.org/doi/abs/10.5555/104134.104148.

Keren, G., and Schuller, B. (2016). "Convolutional RNN: an Enhanced Model for Extracting Features from Sequential Data," in 2016 International Joint Conference on Neural Networks (IJCNN) (Vancouver, BC, Canada: IEEE), 3412–3419. doi:10.1109/IJCNN.2016.7727636

Kerkeni, L., Serrestou, Y., Mbarki, M., Raoof, K., Ali Mahjoub, M., and Cleder, C. (2019). "Automatic Speech Emotion Recognition Using Machine Learning," in *Social Media and Machine Learning*. Editor A. Cano (London: IntechOpen). doi:10.5772/intechopen.84856

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., and Zhang, Y. (2019). Short-term Residential Load Forecasting Based on LSTM Recurrent Neural Network. *IEEE Trans. Smart Grid* 10 (1), 841–851. doi:10.1109/TSG.2017.2753802

Li, Y., and Cao, H. (2018). Prediction for Tourism Flow Based on LSTM Neural Network. *Proced. Comput. Sci.* 129, 277–283. doi:10.1016/j.procs.2018.03.076

Liu, F., Zhou, X., Cao, J., Wang, Z., Wang, H., and Zhang, Y. (2019). "A LSTM and CNN Based Assemble Neural Network Framework for Arrhythmias Classification," in ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Brighton, UK: IEEE), 1303–1307. doi:10.1109/ICASSP.2019.8682299

Ma, Q., Zheng, Z., Zhuang, W., Chen, E., Wei, J., and Wang, J. (2021). Echo Memory-Augmented Network for Time Series Classification. *Neural Networks* 133, 177–192. doi:10.1016/j.neunet.2020.10.015

Maginnis, B., and Richemond, P. H. (2017). Efficiently Applying Attention to Sequential Data with the Recurrent Discounted Attention Unit. arXiv [Preprint]. Available at: arXiv:1705.08480.

Malhotra, P., TV, V., Vig, L., Agarwal, P., and Shroff, G. (2017). TimeNet: Pre-trained Deep Recurrent Neural Network for Time Series Classification. arXiv [Preprint]. Available at: https://arxiv.org/abs/1706.08838.

Mayet, T., Lambert, A., Leguyadec, P., Le Bolzer, F., and Schnitzler, F. (2020). SkipW: Resource Adaptable RNN with Strict Upper Computational Limit. International Conference on Learning Representations. Available at: https://openreview.net/forum?id=2CjEVW-RGOJ.

Mikolajczyk, A., and Grochowski, M. (2018). "Data Augmentation for Improving Deep Learning in Image Classification Problem," in 2018 international interdisciplinary PhD workshop (IIPhDW) (Poland: IEEE), 117–122. doi:10.1109/IIPHDW.2018.8388338

Ororbia, A., ElSaid, A., and Desell, T. (2019). "Investigating Recurrent Neural Network Memory Structures Using Neuro-Evolution," in 2019 Proceedings of the genetic and evolutionary computation conference, (Prague Czech Republic: Association for Computing Machinery), 446–455. doi:10.1145/3321707.3321795

Ostmeyer, J., and Cowell, L. (2019). Machine Learning on Sequential Data Using a Recurrent Weighted Average. *Neurocomputing* 331, 281–288. doi:10.1016/j.neucom.2018.11.066

Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., and Savarese, S. (2019). "Sophie: An Attentive gan for Predicting Paths Compliant to Social and Physical Constraints," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), (Long Beach, CA, USA: IEEE), 1349–1358. doi:10.1109/CVPR.2019.00144

Salman, A. G., Heryadi, Y., Abdurahman, E., and Suparta, W. (2018). Single Layer & Multi-Layer Long Short-Term Memory (LSTM) Model with Intermediate Variables for Weather Forecasting. *Proced. Comput. Sci.* 135, 89–98. doi:10.1016/j.procs.2018.08.153

Saritas, M. M., and Yasar, A. (2019). Performance Analysis of ANN and Naive Bayes Classification Algorithm for Data Classification. *Int. J. Intell. Syst. Appl.* 7 (2), 88–91. doi:10.18201/ijisae.2019252786

Schuster, M., and Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Trans. Signal. Process.* 45 (11), 2673–2681. doi:10.1109/78.650093

Shewalkar, A., Nyavanandi, D., and Ludwig, S. A. (2019). Performance Evaluation of Deep Neural Networks Applied to Speech Recognition: RNN, LSTM and GRU. *J. Artif. Intelligence Soft Comput. Res.* 9 (4), 235–245. doi:10.2478/jaiscr-2019-0006

Singh, S., Pandey, S. K., Pawar, U., and Janghel, R. R. (2018). Classification of Ecg Arrhythmia Using Recurrent Neural Networks. *Proced. Comput. Sci.* 132, 1290–1297. doi:10.1016/j.procs.2018.05.045

Song, I., Chung, J., Kim, T., and Bengio, Y. (2018). "Dynamic Frame Skipping for Fast Speech Recognition in Recurrent Neural Network Based Acoustic Models," in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Calgary, AB, Canada: IEEE), 4984–4988. doi:10.1109/ICASSP.2018.8462615

Su, H., Zhu, J., Dong, Y., and Zhang, B. (2017). "Forecast the Plausible Paths in Crowd Scenes," in 2017 Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), (Melbourne, Australia: International Joint Conferences on Artificial Intelligence), 1–2. doi:10.24963/ijcai.2017/386

Tao, J., Thakker, U., Dasika, G., and Beu, J. (2019). "Skipping Rnn State Updates without Retraining the Original Model," in 2019 Proceedings of the 1st Workshop on Machine Learning on Edge in Sensor Systems, (Coimbra, Portugal: Association for Computing Machinery), 31–36. doi:10.1145/3362743.3362965

Trinh, T., Dai, A., Luong, T., and Le, Q. (2018). "Learning Longer-Term Dependencies in Rnns with Auxiliary Losses," in 2018 Proceedings of the 35th International Conference on Machine Learning (Stockholmsmässan, Stockholm, Sweden: Proceedings of Machine Learning Research (PMLR)), 4965–4974. Available at: http://proceedings.mlr.press/v80/trinh18a.html.

Vemula, A., Muelling, K., and Oh, J. (2018). "Social Attention: Modeling Attention in Human Crowds," in 2018 IEEE international Conference on Robotics and

Automation (ICRA) (Brisbane, Australia: IEEE), 4601–4607. doi:10.1109/ICRA.2018.8460504

Wang, Y., Huang, M., Zhu, X., and Zhao, L. (2016). "Attention-based LSTM for Aspect-Level Sentiment Classification," in 2016 Proceedings of the 2016 conference on empirical methods in natural language processing, (Austin, TX, USA:Association for Computational Linguistics), 606–615. doi:10.18653/v1/d16-1058

Xu, Y., Piao, Z., and Gao, S. (2018). "Encoding Crowd Interaction with Deep Neural Network for Pedestrian Trajectory Prediction," in 2018 Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR), (Salt Lake City, UT, USA: IEEE), 5275–5284. doi:10.1109/CVPR.2018.00553

Yang, B., Sun, S., Li, J., Lin, X., and Tian, Y. (2019). Traffic Flow Prediction Using LSTM with Feature Enhancement. *Neurocomputing* 332, 320–327. doi:10.1016/j.neucom.2018.12.016

Yildirim, O., Baloglu, U. B., Tan, R.-S., Ciaccio, E. J., and Acharya, U. R. (2019). A New Approach for Arrhythmia Classification Using Deep Coded Features and LSTM Networks. *Comput. Methods Programs Biomed.* 176, 121–133. doi:10.1016/j.cmpb.2019.05.004

Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Comput.* 31 (7), 1235–1270. doi:10.1162/neco_a_01199

Zhang, Q., Zhou, D., and Zeng, X. (2017). Heartid: a Multiresolution Convolutional Neural Network for Ecg-Based Biometric Human Identification in Smart Health Applications. *IEEE Access* 5, 11805–11816. doi:10.1109/ACCESS.2017.2707460

Zhao, J., Huang, F., Lv, J., Duan, Y., Qin, Z., Li, G., and Tian, G. (2020). "Do rnn and Lstm Have Long Memory?," in 2020 International Conference on Machine Learning (Vienna, Austria: Proceedings of Machine Learning Research (PMLR)), 11365–11375. Available at: http://proceedings.mlr.press/v119/zhao20c.html.

Zhou, G.-B., Wu, J., Zhang, C.-L., and Zhou, Z.-H. (2016). Minimal Gated Unit for Recurrent Neural Networks. *Int. J. Autom. Comput.* 13 (3), 226–234. doi:10.1007/s11633-016-1006-2

Conflict of Interest: 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.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.