This is a repository copy of *Numerical-discrete-scheme-incorporated recurrent neural network for tasks in natural language processing*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/195787/

Version: Published Version

ORIGINAL RESEARCH

Numerical-discrete-scheme-incorporated recurrent neural network for tasks in natural language processing

Mei Liu[1,2] | **Wendi Luo**[1,2] | **Zangtai Cai**[2] | **Xiujuan Du**[2] | **Jiliang Zhang**[3] | **Shuai Li**[1,2]

[1]School of Information Science and Engineering, Lanzhou University, Lanzhou, China

[2]The State Key Laboratory of Tibetan Intelligent Information Processing and Application, Qinghai Normal University, Xining, China

[3]Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, UK

Correspondence

Shuai Li, School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China.
Email: lishuai@lzu.edu.cn

Funding information

National Natural Science Foundation of China, Grant/Award Number: 62176109; Natural Science Foundation of Gansu Province, Grant/Award Number: 21JR7RA531; Key Laboratory of IoT of Qinghai, Grant/Award Number: 2022-ZJ-Y21; National Key Research and Development Program of China, Grant/Award Number: 2017YFE0118900; Supercomputing Center of Lanzhou University; Tibetan Information Processing and Machine Translation Key Laboratory of Qinghai Province, Grant/Award Number: 2021-Z-003; Sichuan Science and Technology Program, Grant/Award Number: 2022nsfsc0916

Abstract

A variety of neural networks have been presented to deal with issues in deep learning in the last decades. Despite the prominent success achieved by the neural network, it still lacks theoretical guidance to design an efficient neural network model, and verifying the performance of a model needs excessive resources. Previous research studies have demonstrated that many existing models can be regarded as different numerical discretizations of differential equations. This connection sheds light on designing an effective recurrent neural network (RNN) by resorting to numerical analysis. Simple RNN is regarded as a discretisation of the forward Euler scheme. Considering the limited solution accuracy of the forward Euler methods, a Taylor-type discrete scheme is presented with lower truncation error and a Taylor-type RNN (T-RNN) is designed with its guidance. Extensive experiments are conducted to evaluate its performance on statistical language models and emotion analysis tasks. The noticeable gains obtained by T-RNN present its superiority and the feasibility of designing the neural network model using numerical methods.

KEYWORDS
deep learning, natural language processing, neural network, text analysis

1 | INTRODUCTION

Deep learning has made rapid progress and fulfilled great success in a wide spectrum of applications, such as natural language processing (NLP) [1, 2], speech recognition [3], and computer vision [4, 5]. Behind these accomplishments lies the powerful function approximation capability of deep neural networks. In NLP tasks, modelling sequential data is a challenging problem. Plenty of work has been presented to solve the problem of sequential data modelling. Among them, recurrent neural networks (RNNs) achieve satisfactory performance due to their recurrent mechanism [6, 7]. Despite the

accomplishment obtained by the RNNs, the training of the RNN is rather difficult. Back-propagation through time (BPTT) is needed to train the model. When the input sequence is long, the multiplication term produced by the chain derivation mechanism is numerically unstable, which makes RNN suffer from the gradient explosion and gradient vanishing problem [8]. In Ref. [9], linear time-delayed connections were added to RNNs to alleviate the gradient vanishing problem during training. However, it is only examined for some small-scale tasks. The advent of the long short-term memory (LSTM) conspicuously enhanced the performance of the recurrent architecture and solved the gradient vanishing

problem from the perspective of structural design [10]. LSTM designs a fairly elaborated structure to create a gating mechanism. This kind of gating mechanism uses learnable gates to implement a sophisticated feedback approach, which makes it easier for gradient information to flow back. Nevertheless, due to its structural complexity, training the LSTM is time-consuming and it is hard for LSTM to scale for large tasks. Gated feedback connections were used in Ref. [11] between layers of stacked RNNs to adaptively adjust the connection patterns between the adjoint layers. To capture long-term contextual semantic information in sequential data, more memory units were used in Ref. [12] to keep track of previous hidden states, where the weighted connections were directly linked to multiple preceding hidden states. With feedback paths provided by these connections, residual signals can propagate to the farther preceding hidden states to better model the long-term memory. In addition, these connections give more feedback paths to a model to smooth its update during training. In Ref. [13], the authors discussed the high-order connections in the Markov property framework. With the reduction in parameters by projecting hidden-state vectors to a low dimension and weighted connections on hidden states, the model presented achieves noticeable gains both in accuracy and efficiency when applied to acoustic modelling. However, its wiring pattern does not change in essence. It aggregates historical information by adding weighted connections. By combining high- and low-order LSTM with a pruning technique, Zhang et al. [14] introduced a recurrent model called MO-BILSTM, which achieved promising results evaluated in two named entity recognition datasets. A generalisation of LSTM called multiple-history LSTM was investigated in Ref. [15], where different LSTM units were connected with high-order feedback and maintained historical information at different time steps.

From the topology of RNNs, it allows connections among the preceding hidden units. Through these connections, RNNs maintain a special mechanism of recurrent feedback that summarises the past sequence of the inputs, enabling themselves to capture correlations between the temporally distant events in the data. Previous research has shown that the recurrent mechanism of the RNN model has the same form as the explicit forward Euler scheme. If unfold in time, simple RNN can be regarded as a kind of discrete forward Euler scheme from the perspective of numerical analysis. Inspired by the connection between RNNs and ordinary differential equations (ODEs), this study hammers at designing a Taylor-type recurrent neural network (T-RNN) guided by a Taylor-type discrete scheme. Therefore, considering the shortcomings of RNN, such as the difficulty of obtaining context information in a long distance, it can be explained by the fact that the forward Euler scheme only uses first-order derivative information for estimation, which causes a large truncation error. To improve the simple RNN from the perspective of numerical analysis, a Taylor-type discrete scheme with a tiny truncation error deduced from the Taylor expansion is presented in this paper. In addition, the Taylor-type discrete scheme is used as an orientation to construct the T-RNN to further explore the connection between neural networks and ODEs and to improve the performance of the RNNs. As shown in Figure 1, extensive experiments are conducted to evaluate the performance of T-RNN. Experimental results indicate that T-RNN has higher accuracy and can capture longer contextual information compared to the existing RNN model. The main contributions in this paper are summarised as follows:

- Relevant analysis is given to show the connection between the numerical formulas and the neural network structure. Detailed derivation is given to introduce the Taylor-type numerical formulas, and on this base, the T-RNN is designed.
- Sufficient experiments are conducted on benchmark datasets to evaluate the T-RNN. Experimental results demonstrate its superiority when compared to simple RNN.
- The performance gains obtained by the T-RNN echoes the connection between the neural networks and the numerical discretisation schemes and indicates that it is promising to improve the neural networks from the perspective of numerical analysis.
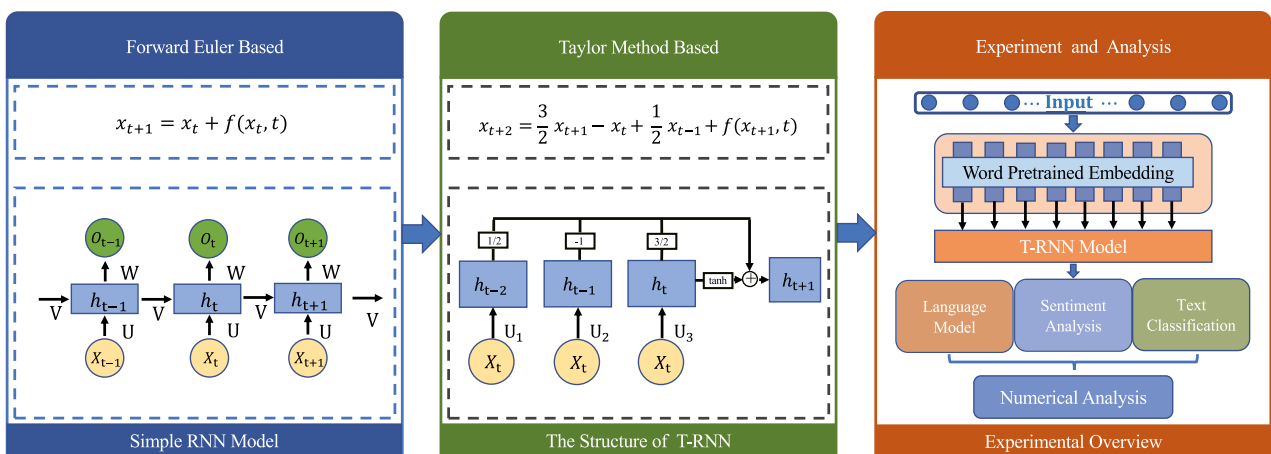

FIGURE 1 Overview of this paper.

2 | RELATED WORK

The performance of the neural network especially relies on its topological structure, which is an important issue in deep learning. The past decade has witnessed the emergence of many neural network architectures. As for a convolution neural network (CNN), it is typically a concatenation of many non-linear layers and ends with a fully connected layer. An obvious experience of CNN architectures is that the number of layers is rising, ranging from the AlexNet [16] with five convolutional layers, the VGG [17] network with 19 layers to the GoogleNet [18] with 22 layers. Nevertheless, as the layer goes deep, it suffers from gradient explosion, gradient vanishing problems, and model degradation [8]. The advent of ResNet sheds light on training very deep neural networks by introducing the mechanism of identity mapping, which can maintain the stability and correlation of gradient during back-propagation [19]. From then on, the idea of introducing the skipping connection between layers springs up. In Ref. [20], the DenseNet was presented by connecting each layer to every other layer in a forward fashion. For each layer, its input is the output of all preceding layers. Similarly, the cliqueNet constructed the layers as a loop, where each layer was both the input and output of any other layer in a block [21]. These kinds of elaborate wiring patterns make full use of the feature information between layers. Analogously, authors in [12] tried to connect preceding hidden states with weighted links to gather more historical information. In Ref. [13], a neural network was presented under the Markov theory framework, which improved the neural network by introducing the skip connection in essence. In addition to adding skipping connections, Zagoruyko and Komodakis [22] increased the number of channels to explore what the width of a network influences. In addition, authors in [23] used the LSTM to construct a controller and introduced a neural network by resorting to reinforcement learning. In Ref. [24], the random graph was used to design a randomly wired neural network. Despite the abundance of neural architecture, the design of neural networks still lacks guidance. It is time-consuming and artificially involved to verify the performance of a neural network model. In Refs. [25, 26], the connection between neural networks and dynamic systems was revealed and authors indicated that deep neural networks could be deemed as different discretizations of ODEs. Chen et al. [27] provided an original perspective of neural network and brought the neural ODEs to view. In Ref. [28], authors indicated that several existing networks, such as ResNet [19] and RevNet [29], were related to correspondingly numerical discrete schemes and put forward a neural network architecture oriented by the linear multistep method solving ODEs, which further verified the relationship between numerical discrete formulations and multi-layer neural networks. Luo et al. [30] used the Runge-Kutta discrete schemes as a principle to guide the stack of layers to design a neural network and brought remarkable performance improvement. In Ref. [31], partial differential equations were used to design the CNNs and related numerical techniques were also used to solve and optimise the neural network. As for the T-RNN, by reconsidering the RNN from the perspective of numerical analysis and deeming it as a discrete form of the forward Euler scheme, the T-RNN is proposed based on a higher-order Taylor-type discrete scheme deduced from the Taylor expansion.

At present, with its unique position encoding to capture long-range word-order information, transformer has achieved the state-of-art records in many fields, such as language model tasks [32] and computer vision [33]. Despite its success in many tasks, for data with strict timing information requirements, there are still some disadvantages to using transform for time series data [34]. However, RNNs have innate advantages for time series data processing with its continuous-time hidden-state mechanisms [35]. In addition, the structure of transform contains six layers of encoder and six layers of decoder [36]. The complicated structure makes it require a large amount of data for training. When input sequence length increases, the consumption of memory and computation is massive if its parallel computation is not supported. Therefore, in some application scenarios with limited computation or few data samples, it is still necessary and appliable to resort to RNNs.

3 | PRELIMINARIES

In this section, the deduction of the Taylor-type discrete scheme is demonstrated in detail. With the connections between RNN and the discrete forward Euler scheme explained, the T-RNN is constructed guided by the Taylor-type discrete scheme.

3.1 | Deduction

Due to its recurrent structure allowing connections among hidden units relevant to a time delay, the RNNs have achieved the prominent performance on a large number of tasks [37]. For a typical simple RNN, at time step t, given an input x_t and a hidden state h_{t-1} generated in the previous time step, the update process of RNN is demonstrated as

$$h_t = \Phi(W_{xh}x_t + W_{hh}h_{t-1} + b_{xh} + b_{hh}), \tag{1}$$

The symbol $\Phi(\cdot)$ represents the non-linear activation function usually being rectified linear (ReLU) or hyperbolic tangent function (Tanh), which adds non-linear factors and improves the expressiveness of the neural network. W_{xh} is the weighted matrix in the input connection, and W_{hh} is the weighted matrix of the hidden state. Typically, the RNN accepts the input and the hidden state of the previous time step as the input of the activation function, and the output of the activation function becomes the hidden state at the current time. The corresponding interpretation is demonstrated in Figure 2.

With the Taylor expansion, the following rules can be obtained:
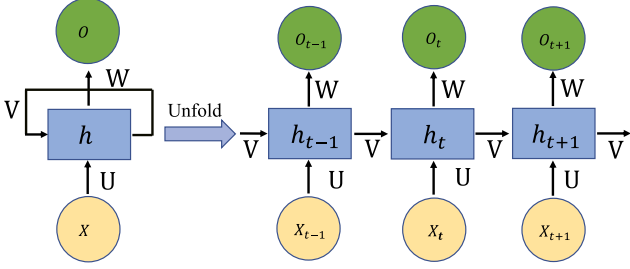
FIGURE 2 Demonstration of the unfold recurrent neural network structure without the output layer.

$$\Phi(t_3) = \Phi(t_2 + \tau)$$

$$= \Phi(t_2) + \tau\dot{\Phi}(t_2) + \frac{\tau^2}{2!}\Phi^{(2)}(t_2) + \frac{\tau^3}{3!}\Phi^{(3)}(h_1), \quad (2)$$

where $h_1 \in (t_2, t_3)$. Subsequently, we obtain the term

$$\dot{\Phi}(t_2) = \frac{\Phi(t_3) - \Phi(t_2)}{\tau} - \frac{\tau}{2!}\Phi^{(2)}(t_2) - \frac{\tau^2}{3!}\Phi^{(3)}(h_1). \quad (3)$$

Likewise, by applying one- and two-step backward expansion, we have

$$\Phi(t_1) = \Phi(t_2 - \tau)$$

$$= \Phi(t_2) - \tau\dot{\Phi}(t_2) + \frac{\tau^2}{2!}\Phi^{(2)}(t_2) - \frac{\tau^3}{3!}\Phi^{(3)}(h_2), \quad (4)$$

and

$$\Phi(t_0) = \Phi(t_2 - 2\tau)$$

$$= \Phi(t_2) - 2\tau\dot{\Phi}(t_2) + \frac{4\tau^2}{2!}\Phi^{(2)}(t_2) - \frac{8\tau^3}{3!}\Phi^{(3)}(h_3), \quad (5)$$

where h_2 and h_3 lie between (t_1, t_2) and (t_0, t_2), respectively. The above equations can be paraphrased as follows:

$$\dot{\Phi}(t_2) = \frac{\Phi(t_2) - \Phi(t_1)}{\tau} + \frac{\tau}{2!}\Phi^{(2)}(t_2) - \frac{\tau^2}{3!}\Phi^{(3)}(h_2), \quad (6)$$

and

$$\dot{\Phi}(t_2) = \frac{\Phi(t_2) - \Phi(t_0)}{2\tau} + \tau\Phi^{(2)}(t_2) - \frac{2\tau^2}{3}\Phi^{(3)}(h_3). \quad (7)$$

Then, let Equation (5) add Equation (7), then minus Equation (6), and we have

$$\dot{\Phi}(t_2) = \frac{2\Phi(t_3) - 3\Phi(t_2) + 2\Phi(t_1) - \Phi(t_0)}{2\tau}$$

$$+ \tau^2\left(-\frac{1}{3!}\Phi^{(3)}(c_1) - \frac{2}{3}\Phi^{(3)}(c_3) - \frac{1}{3!}\Phi^{(3)}(c_2)\right). \quad (8)$$

As the term $\left(-\frac{1}{3!}\Phi^{(3)}(c_1) - \frac{2}{3}\Phi^{(3)}(c_3) - \frac{1}{3!}\Phi^{(3)}(c_2)\right)$ is independent to t, we can paraphrase Equation (8) as

$$\dot{\Phi}(t_2) = \frac{2\Phi(t_3) - 3\Phi(t_2) + 2\Phi(t_1) - \Phi(t_0)}{2\tau} + O(\tau^2). \quad (9)$$

With the item $O(\tau^2)$ discarded, we have

$$\dot{\Phi}(t_2) \approx \frac{2\Phi(t_3) - 3\Phi(t_2) + 2\Phi(t_1) - \Phi(t_0)}{2\tau}. \quad (10)$$

Note that Equation (10) is the Taylor-type 1-step-ahead numerical differential scheme because it has the term $\Phi(t_3)$, which is one step ahead of the $\Phi(t_2)$. By setting the interval τ to be 1 and moving item $\Phi(t_3)$ to the left, we obtain

$$\Phi(t_3) \doteq \frac{3}{2}\Phi(t_2) - \Phi(t_1) + \frac{1}{2}\Phi(t_0) + \dot{\Phi}(t_2), \quad (11)$$

where $\doteq$ denotes the computational assignment operation. The different time points, t_1, t_2, and t_3, can be discretized into different and adjacent layers in neural networks, and Equation (11) can be rewritten as

$$\Phi(t_{k+2}) \doteq \frac{3}{2}\Phi(t_{k+1}) - \Phi(t_k) + \frac{1}{2}\Phi(t_{k-1}) + \dot{\Phi}(t_{k+1}). \quad (12)$$

Now, Equation (12) is the Taylor-type discrete scheme, which uses the information of the three historical time steps and the gradient information of the previous moment to estimate the value of the current step. It is worth pointing out that the Taylor-type discrete scheme has longer-term dependencies upon historical data compared to the discretisation of the forward Euler method. Mathematically, the truncation error of the Taylor-type discrete scheme is $O(\tau^3)$, which has higher precision.

3.2 | T-RNN with a Taylor-type discrete scheme

In fact, we paraphrase Equation (1) in a general form, and Equation (13) is obtained,

$$y_t = f(y_{t-1}, x_t, t). \quad (13)$$

Traditionally, for an ODE $\dot{x}(t) = f(x(t), t)$, the forward Euler scheme is a commonly used numerical solution whose general form is presented as

$$x_{t+h} = x_t + hf(x_t, t), \quad (14)$$

where h denotes the step size. It can be found that Equation (14) is in compliance with Equation (13), which indicates the connections between the RNN and the forward Euler scheme. The same observations are also found in the field of CNN, where ResNet [19], RevNet [29], and LM-ResNet [28] all

can be connected to different numerically discrete schemes. Inspired by this connection, this paper hammers at constructing a recurrent network structure based on the presented Taylor-type discrete scheme. This part begins with demonstrating the meaning of Equation (12) and explaining the connections between T-RNN and it. Actually, Equation (12) predicts the function value of the current data point based on the linear combination of three historical data points and the derivative of the previous step. It has long-term dependencies upon historical data because it needs three pieces of information of the past. When modelling the sequential data, it is crucial to capture the contextual semantic information. Inspired by this idea, the T-RNN is constructed guided by the Taylor-type discrete scheme.

Specifically, three neural units are used to represent the three historical terms of the Taylor-type scheme. At each moment, three neural units calculate the relationship between the current input and three hidden states of history. We use the activation value of the Tanh function at the previous moment to replace the derivative term in the Taylor discrete scheme. The output result is calculated in the form of the Taylor discrete scheme and is the hidden state at the current moment. More specifically, at time $t + 1$, if the bias term is omitted, the update process of T-RNN can be presented from Equations (15) to (18).

$$Q_1 = \Phi\left(W_{xh}^{t-2}x_t + W_{hh}^{t-2}h_{t-2}\right), \tag{15}$$

$$Q_2 = \Phi\left(W_{xh}^{t-1}x_t + W_{hh}^{t-1}h_{t-1}\right), \tag{16}$$

$$Q_3 = \Phi\left(W_{xh}^{t}x_t + W_{hh}^{t}h_t\right), \tag{17}$$

$$h_{t+1} = \frac{3}{2}Q_3 - Q_2 + \frac{1}{2}Q_1 + \tanh(Q_3). \tag{18}$$

From Equations (15) to (18), the three terms Q_1, Q_2, and Q_3 represent three correspondingly historical data in the formula and demonstrate the longer-term context in the neural network model. The terms W_{xh}^{t-2}, W_{xh}^{t-1}, W_{xh}^{t}, W_{hh}^{t-2}, W_{hh}^{t-1}, W_{hh}^{t2} are correspondingly weighted matrix and the new hidden state at $t + 1$ is denoted as h_{t+1}. The corresponding interpretation is demonstrated in Figure 3.
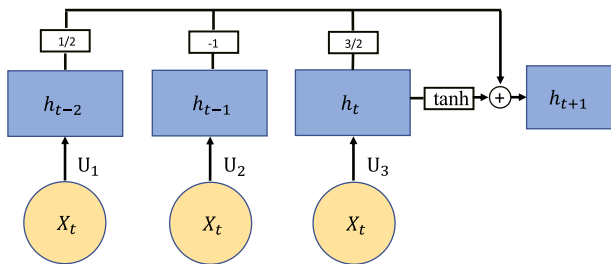
FIGURE 3 Illustration of Taylor-type recurrent neural network structure without the output layer.

4 | DATASETS AND EXPERIMENTS

In this section, different experiments are conducted to evaluate the performance of T-RNN. The corresponding results are also manifested.

4.1 | Sentiment analysis

The target of sentiment analysis is to analyse subjective texts with emotional colours to determine the views, preferences, and emotional tendencies of the text. Our model is evaluated on sentiment analysis tasks based on Internet Movie Database (IMDB). The IMDB dataset contains 50,000 reviews, which are divided into two equal parts being a training set and a test set. For each part, positive and negative reviews each account for half. In our experiments, we set the learning rate to 0.1 and use the mini-batch stochastic gradient descent (SGD) algorithm to train the models. In addition, the batch size is fixed to 64 and the training epoch is set to 550. A hard clipping is set to 1.0 to avoid gradient explosion during training. The vocabulary is composed of the top 25,000 words with the highest frequency. It is worth mentioning that the text sequence length of the IMDB dataset varies a lot, we use the zero padding technique to fill the input sequence to a fixed length. To reduce the impact of the zero padding, we set the sequence length to 16. Moreover, the input size is set to 100 and the hidden size is set to 256 in each hidden layer. We use the dropout regularisation [38] in all experiments with the dropout coefficient being 0.5. In addition, we use the Glove representation technique to initiate the input vector to improve the training efficiency [39].

The corresponding results are demonstrated in Figure 4, where we can find that T-RNN outperforms simple RNN on test accuracy. As shown in Figures 5 and 6, we can find that T-RNN has a slow rate in the initial period and suffers from oscillation in the early training process. However, the T-RNN reaches a higher accuracy eventually. We discuss the reasons for this phenomenon in the discussion section. In addition, after 100 epochs, the test loss of RNN rises, which means that the RNN occurs overfitting, but T-RNN avoids this situation.

4.2 | Statistic language model

Traditionally, a statistical language model is used to describe the probability distribution over different grammatical units of words, sentences, and even the entire document in natural languages. In language modelling tasks, it is quite important to take advantage of the long-term dependency of natural language. We evaluate the proposed architecture on the Penn Treebank (PTB) and Text8. It is worth mentioning that considering the limitation of computing resources, we have tailored the Text8 corpus and the corresponding details are shown in Table 1. In our experiments, all neural networks are trained by the SGD algorithm. In order to circumvent the gradient explosion during the BPTT, a simple strategy that sets a hard-clipping to 1.0 is adopted during training. When training
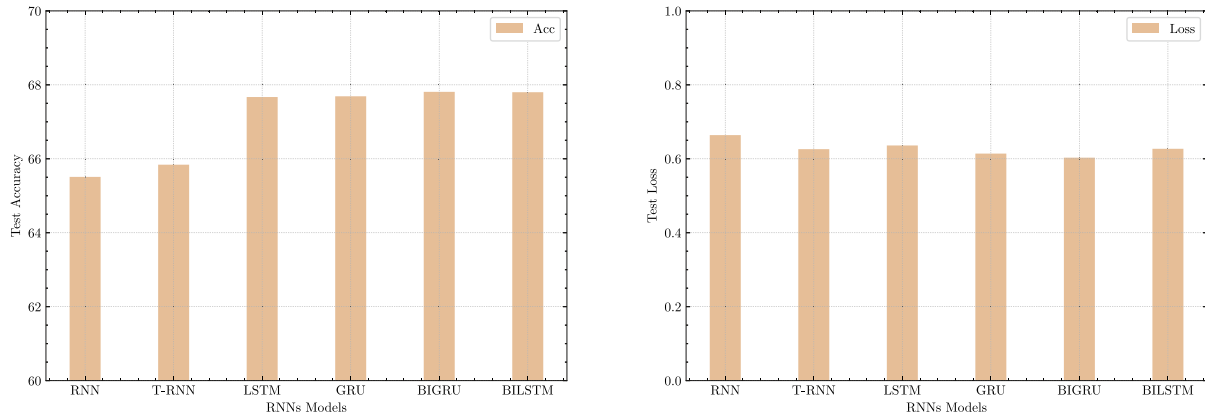
FIGURE 4 Result about different RNNs models on the IMDB dataset. Left: percent of test accuracy, right: test loss. IMDB, Internet Movie database; RNNs, recurrent neural networks.
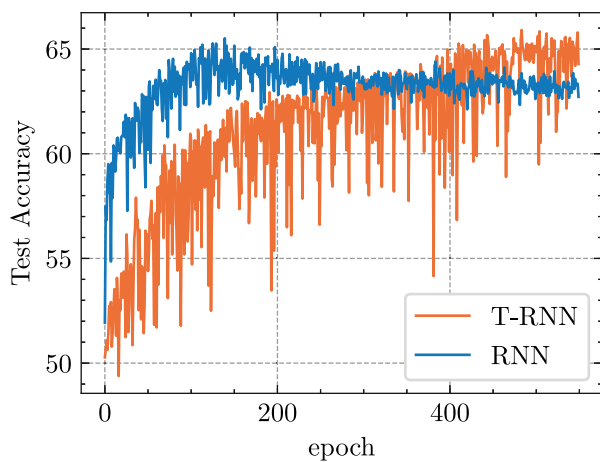
FIGURE 5 Demonstration of accuracy change when training the T-RNN and RNN. RNN, recurrent neural network; T-RNN, Taylor-type RNN.
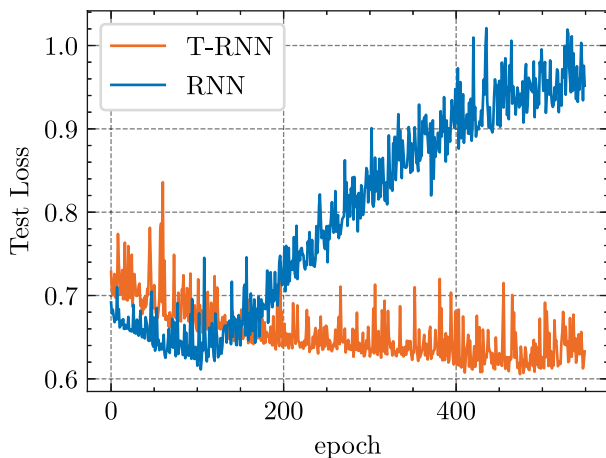
FIGURE 6 Demonstration of loss change when training the T-RNN and RNN. RNN, recurrent neural network; T-RNN, Taylor-type RNN.

the model, we set the initial learning rate to 0.5 and halve the learning rate every 100 epochs. The vocabulary size of PTB is limited to 10k and that of Text8 is set to 50k. To improve the generalisation of the model, the dropout regularisation is

TABLE 1 Details about PTB and Text8 datsets

Corpus	Train size	Test size
PTB	4.9M	0.44M
Text8	20M	1.2M

Abbreviation: PTB, Penn Treebank.

adopted and the dropout coefficient is set to 0.5 [38]. We use 500 nodes in each hidden layer and the input size to 300 for the PTB and Text8 data set. To improve the training efficiency, we use Glove representation technique [39]. In our experiment, we set different input sequence lengths ranging from 64, 128 to 256 on PTB. Similarly, we set sequence lengths ranging from 128, 200 to 256 on Text8.

The corresponding experimental results on different datasets are demonstrated in Table 2, where we can find that the T-RNN can achieve lower test perplexity compared to the RNN on two datasets, which means the superiority of T-RNN compared to RNN.

4.3 | Text classification

Text classification is an important part in text process, which has abundant applications, such as Garbage Filtering, News classification, and Part-Of-Speech Tagging. In this part, we evaluate the proposed T-RNN and RNNs models on text classification tasks. R8 and R52 datasets are chosen to train the models, which are all the subset of Reuters-21578 datasets. Their details are shown in Table 3. As for the experimental setting, we set the hard-clipping to 1.0, the learning rate to 0.1, and the back-propagation step to 64 for two datasets. When training on the R8 dataset, the input size is 100, the hidden size is fixed to 256 and the batch size is 64, and the training epoch is 800. As for the R52 dataset, the input size is 300, the hidden size is fixed to 500, and the batch size is 32, and the training epoch is 1000. In addition, we use the hidden state of the last step as the result of model to predict the final result. We choose the highest test accuracy as the final result and the complete results are presented in Table 4. We choose accuracy,

loss value, and F1 score as the criterion to demonstrate the degree of performance gains, Different RNN models, such as bi-directional recurrent neural network (BIRNN) and bi-directional gated recurrent unit (BIGRU), are used as comparisons to evaluate the performance of T-RNN.

From Table 4, we find that the accuracy of T-RNN on the R8 dataset is 81.6% and 5.1% higher than that of RNN, and the accuracy on the R52 dataset is 1.8% higher. There are also corresponding improvements in test loss and F1 score. Although the performance of T-RNN is lower than that of LSTM or GRU, it is worth pointing out that the performance improvement of T-RNN compared to RNN demonstrates the promising feasibility to improve the simple RNN model from a numerical method perspective, which provides a reference for improving the performance of neural network models.

5 | DISCUSSION

When conducting experiments on deep learning tasks, we observe some consistent experimental phenomena on three experimental tasks, which are shown in Figures 5–10. In the

TABLE 2 Results about different RNNs models on PTB and Text8 datasets

Models	PTB			Text8			Criteria
	64	128	256	64	200	256	
LSTM	94.9	101.5	115.9	195.9	215.7	226.1	Test PPL
GRU	96.4	96.4	107.5	195.9	204.0	211.5	Test PPL
RNN	110.0	110.1	117.4	256.3	251.8	255.6	Test PPL
T-RNN	108.4	106.2	110.5	238.3	230.3	232.3	Test PPL

Abbreviations: GRU, gate recurrent unit; LSTM, long short term memory; PPL, perplexity; PTB, Penn Treebank; RNN, recurrent neural network.

TABLE 3 Details about R8 and R52 datasets

Corpus	Total size	Train item	Test item	Class
R8	4.5M	5485	2189	8
R52	5.5M	6097	3003	52

experiment of sentiment analysis, we conclude from Figures 5 and 6 that, at the beginning of the training, the test accuracy of T-RNN fluctuates and its convergence rate is slow, but eventually T-RNN reaches a higher accuracy in the end. Similarly, when evaluated on statistical language model tasks, as shown in Figures 9 and 10, for the same input sequence length, T-RNN reaches the test perplexity that lags behind RNN in the early stage of training. However, after a certain epoch, the performance of T-RNN exceeds RNN. For the fixed epoch, as the input sequence increases, the performance gap between models becomes larger, which is demonstrated by the increasing margin between two curves in Figure 7. The results on Text8 also have the same tendency and the corresponding difference values are 18, 21, 23 ranging from (a), (b) to (c) in Figure 8. We explain this phenomenon by resorting to numerical experiments.

5.1 | Numerical experiment

Nonlinear function optimisation problem is fairly common in many scientific problems. Numerical methods are widely used to solve them. For example, a non-linear function optimisation problem is presented as

$$
\begin{aligned}
\min_{f(\zeta_i) \in \mathbb{R}^4} & \mathsf{F}\left(f(\zeta_i), \zeta_i\right) \\
= & (f_1(\zeta_i) + \sin(\zeta_i))f_3(\zeta_i) \\
& + 0.1(\zeta_i - 1)f_3(\zeta_i)f_4(\zeta_i) \\
& - (f_1(\zeta_i) + \log(0.1\zeta_i + 1))(f_2(\zeta_i) + \sin(\zeta_i)) \\
& + (f_3(\zeta_i) - \exp(-\zeta_i))^2 + (f_4(\zeta_i) + \exp(-\zeta_i))^2 \\
& + (f_1(\zeta_i) + \zeta_i)^2 + (f_2(\zeta_i) + \zeta_i)^2.
\end{aligned}
$$

$$(19)$$

We use the forward Euler scheme and the Taylor-type discrete scheme to solve this non-linear optimisation problem. In our experiments, $f(\zeta_i) = [f_1(\zeta_i), f_2(\zeta_i), f_3(\zeta_i), f_4(\zeta_i)]$. We use the L2-norm of $\delta(\zeta_i) = \partial_{\mathsf{F}}(f(\zeta_i), \zeta_i)/\partial f(\zeta_i)$ as the residual error and fix calculation interval to [0, 10]. Different sampling intervals h are set to analyse the convergence performance of
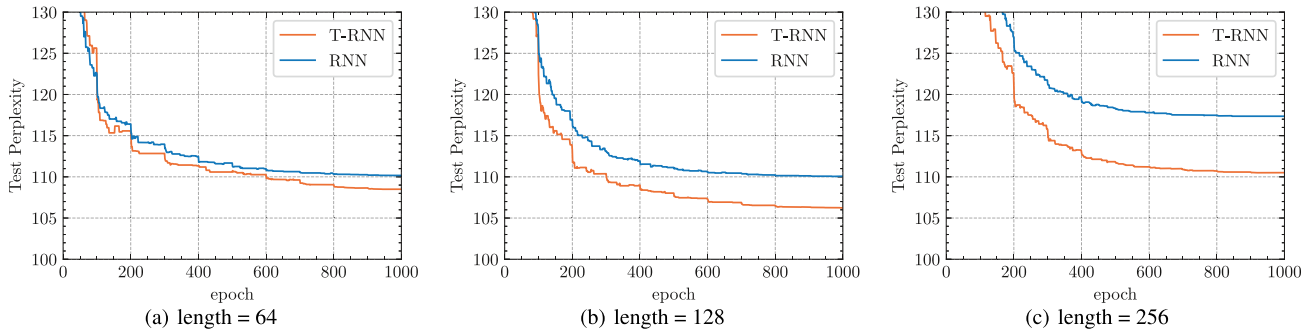
TABLE 4 Comparison of different RNNs models for three evaluation metrics under two datasets (Acc: accuracy; loss: test loss; F1: F1 score)

Models	R8 dataset			R52 dataset		
	Acc	Loss	F1	Acc	Loss	F1
BIRNN	0.875 ± 0.002	0.016 ± 0.001	0.870 ± 0.002	0.862 ± 0.003	0.034 ± 0.002	0.852 ± 0.003
BILSTM	0.920 ± 0.006	0.006 ± 0.001	0.931 ± 0.006	0.899 ± 0.005	0.020 ± 0.001	0.899 ± 0.005
BIGRU	0.927 ± 0.006	0.006 ± 0.001	0.927 ± 0.006	0.891 ± 0.004	0.025 ± 0.003	0.886 ± 0.004
LSTM	0.913 ± 0.007	0.011 ± 0.003	0.915 ± 0.007	0.859 ± 0.007	0.036 ± 0.003	0.857 ± 0.008
GRU	0.911 ± 0.009	0.012 ± 0.000	0.912 ± 0.010	0.842 ± 0.004	0.044 ± 0.002	0.836 ± 0.005
RNN	0.765 ± 0.005	0.019 ± 0.001	0.750 ± 0.004	0.664 ± 0.004	0.059 ± 0.010	0.633 ± 0.010
T-RNN	0.816 ± 0.013	0.015 ± 0.003	0.809 ± 0.012	0.682 ± 0.009	0.049 ± 0.004	0.660 ± 0.007

Abbreviations: BIGRU, bi-directional gate recurrent unit; BIRNN, bi-directional recurrent neural network; GRU, gate recurrent unit; LSTM, long short term memory; RNN, recurrent neural network.

FIGURE 7 Test perplexity of two models with different sequence lengths on Penn Treebank.
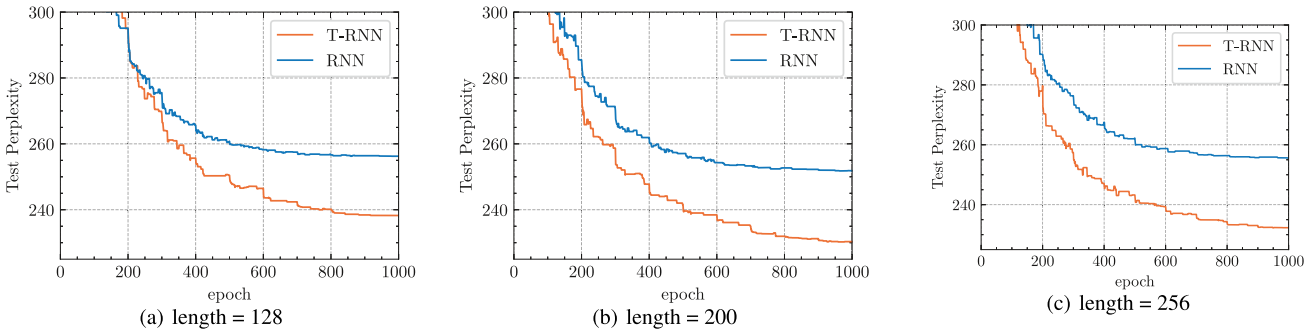

FIGURE 8 Test perplexity of two models with different sequence lengths on Text8.
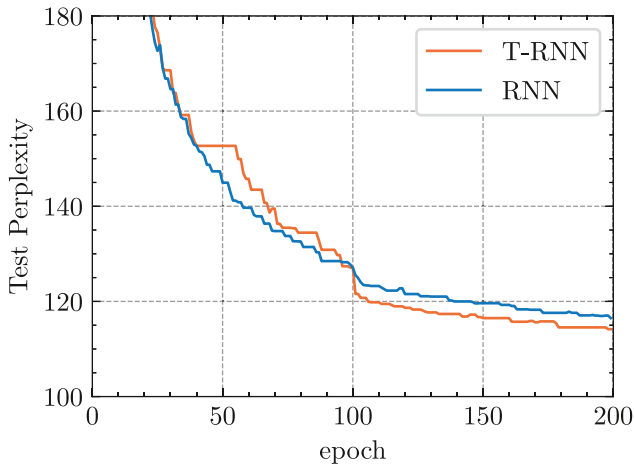

FIGURE 9 Test perplexity of two models with a sequence length fixed to 128 at early epoch on Penn Treebank.
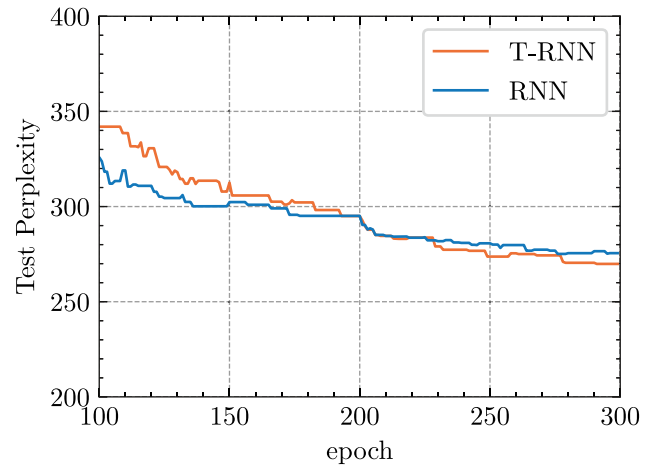

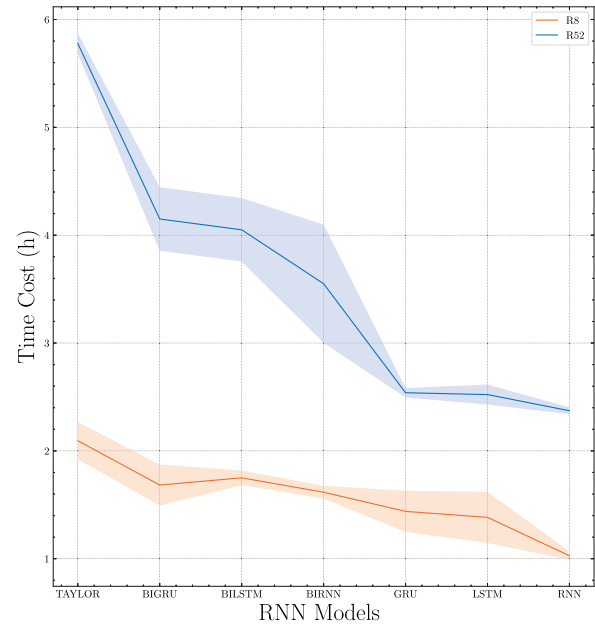**FIGURE 10** Test perplexity of two models with a sequence length fixed to 128 at early epoch on Text8.

the two discrete schemes. The corresponding results are demonstrated in Figure 11, we conclude that compared with the forward Euler scheme, the Taylor-type discrete scheme achieves higher accuracy in the end, which shows its superiority. In addition, at the initial stage of the iteration, the Taylor-type discrete scheme suffers from fluctuations. The historical term can be understood as the time delay that affects the stability of the dynamic system. The Taylor-type discrete scheme needs three previous pieces of information to estimate the value of the next step, so it is more prone to instability at the beginning of the iteration. Therefore, the Taylor-type

scheme requires more iterations to be stable and achieve higher accuracy. This explains the experimental phenomenon in Figures 5 and 6. Similarly, in the statistical language model tasks, the perplexity of T-RNN lags behind the RNN, but it eventually performs better. When the number of the epoch is fixed, the long input sequence means a large number of iterations, which makes the performance gap between T-RNN and RNN wider. Our results in the numerical experiment are consistent with the results of T-RNN in different tasks. The Taylor-type scheme suffers from vibration in the beginning but reaches stable and results in less error at the end of iteration.
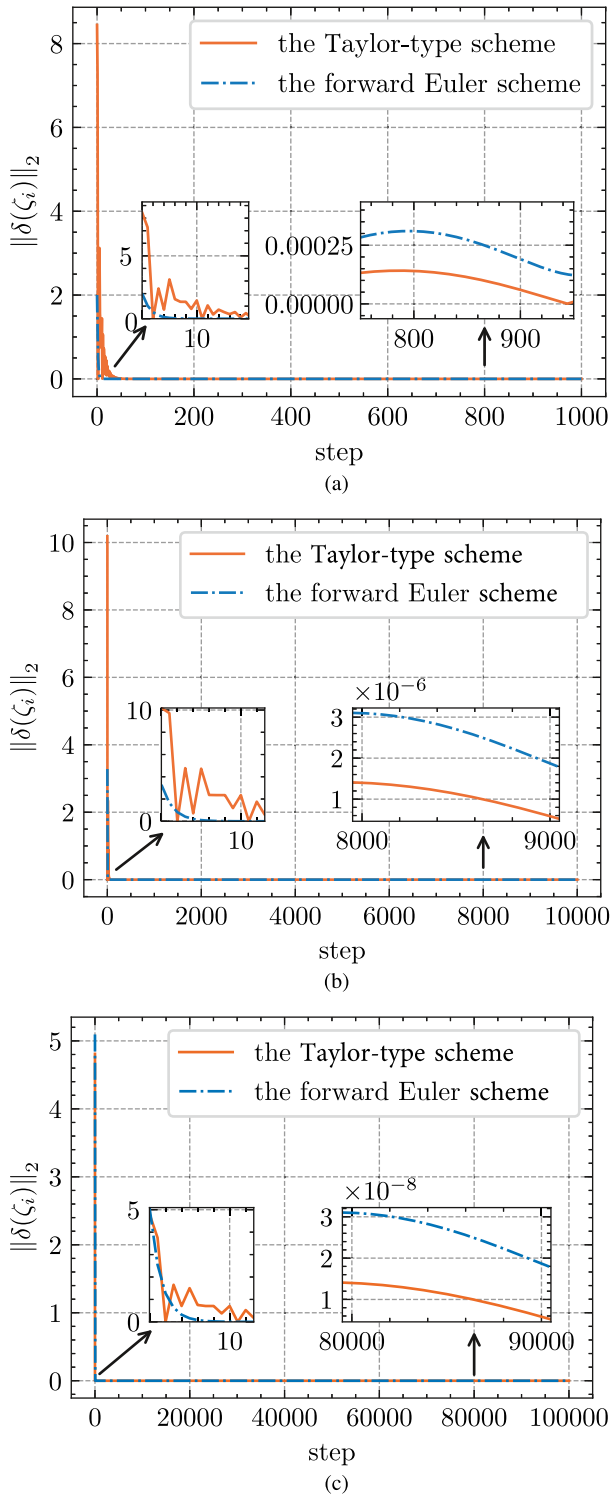
FIGURE 11 Residual errors of two discrete schemes with different sample intervals. (a) $h = 0.01$. (b) $h = 0.001$. (c) $h = 0.0001$.

FIGURE 12 The mean and standard deviation of time cost of different models on text classification.

out that T-RNN is suitable for processing sequence datasets with small quantity such as text or voice data as RNN. Compared with RNN, T-RNN can utilise longer-distance context information, which results in its superiority.

6 | CONCLUSION

In this paper, spurred by the connection between neural networks and discretizations of ODEs, we have proposed the T-RNN guided by a Taylor-type discrete scheme deduced from the Taylor expansion. Systematic experiments have been conducted to testify the performance of the proposed model. The noticeable performance gains upon many tasks have indicated that it is feasible to design effective and powerful neural networks by following certain discrete schemes. The relations between neural networks and the discrete numerical scheme also manifest that plenty of mathematical tools from optimal control and dynamic systems can be used to design optimisation algorithms to train the neural network. In addition, the research about robustness and generalisation of the corresponding neural networks is also worthy of diving into from the perspective of the numerical analysis in the future.

Taking the time cost and accuracy into consideration, we can conclude from Figure 12 that training of T-RNN is more time-consuming than RNN, which attributes to the three history items of T-RNN. The three items need more time to learn the long-distance contextual information, which can be used to memorise longer contextual information. It is worth pointing

CONFLICT OF INTEREST

The author declares that there is no conflict of interest that could be perceived as prejudicing the impartiality of the research reported.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Shuai Li https://orcid.org/0000-0002-0215-0307

REFERENCES

1. Zhou, B., et al.: Cross-domain sequence labelling using language modelling and parameter generating. CAAI Trans. Intell. Technol. 7(4), 710–720 (2022). https://doi.org/10.1049/cit2.12107
2. Yang, Z.L., et al.: Breaking the softmax bottleneck: a high-rank RNN language model. arXiv preprint arXiv:1711.03953 (2017)
3. Fan, C., et al.: Gated recurrent fusion with joint training framework for robust end-to-end speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process. 29, 198–209 (2021). https://doi.org/10.1109/taslp.2020.3039600
4. Dolecek, G.J., Cho, N.: Advances in image processing using machine learning techniques. CAAI Trans. Intell. Technol. 15, 615–618 (2022)
5. Shivakumara, P., et al.: CNN-RNN based method for license plate recognition. CAAI Trans. Intell. Technol. 3, 169–175 (2018). https://doi.org/10.1049/trit.2018.1015
6. Zhang, X.Y., et al.: Design and analysis of recurrent neural network models with non-linear activation functions for solving time-varying quadratic programming problems. CAAI Trans. Intell. Technol. 6(4), 394–404 (2021). https://doi.org/10.1049/cit2.12019
7. Tian, C.W., et al.: Enhanced CNN for image denoising. CAAI Trans. Intell. Technol. 4(1), 17–23 (2019). https://doi.org/10.1049/trit.2018.1054
8. Liu, M., et al.: Activated gradients for deep neural networks. IEEE Transact. Neural Networks Learn. Syst., 1–13 (2021). https://doi.org/10.1109/tnnls.2021.3106044
9. Hihi, S.E., Bengio, Y.: Hierarchical recurrent neural networks for long-term dependencies. Adv. Neural Inf. Process. Syst. 8, 493–499 (1995)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. 9(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
11. Chung, J.Y., et al.: Gated feedback recurrent neural networks. Int. Conf. Mach. Learn. 37, 2067–2075 (2015)
12. Soltani, R., Jiang, H.: Higher order recurrent neural networks. arXiv preprint arXiv:1605.00064 (2016)
13. Zhang, C., Woodland, P.C.: High order recurrent neural networks for acoustic modelling. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5849–5853. Calgary (2018)
14. Zhang, Y., et al.: Does higher order LSTM have better accuracy for segmenting and labeling sequence data? arXiv preprint arXiv:1711.08231 (2017)
15. Huang, H.G., Mak, B.: To improve the robustness of LSTM-RNN acoustic models using higher-order feedback from multiple histories. In: INTERSPEECH, pp. 3862–3866 (2017)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25, 1097–1105 (2012)
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
18. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9. Boston (2015)
19. He, K.M., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778. Las Vegas (2016)
20. Huang, G., et al.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708. Honolulu (2017)
21. Yang, Y.B., et al.: Convolutional neural networks with alternately updated clique. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2413–2422. Salt Lake City (2018)
22. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
23. Barret, Z., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)
24. Xie, S.N., et al.: Exploring randomly wired neural networks for image recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1284–1293 (2019)
25. Weinan, E.: A proposal on machine learning via dynamical systems. Commun. Math. Stat. 5, 1–11 (2017). https://doi.org/10.1007/s40304-017-0103-z
26. Eldad, H., Lars, R.: Stable architectures for deep neural networks. Inverse Probl. 34(1), 014004 (2017). https://doi.org/10.1088/1361-6420/aa9a90
27. Chen, R.T.Q., et al.: Neural ordinary differential equations. Adv. Neural Inf. Process. Syst. 31, 6572–6583 (2018)
28. Lu, Y.P., et al.: Beyond finite layer neural networks: bridging deep architectures and numerical differential equations. In: International Conference on Machine Learning, pp. 3276–3285 (2018)
29. Gomez, A.N., et al.: The reversible residual network: backpropagation without storing activations. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 2211–2221 (2017)
30. Luo, Z.B., et al.: Rethinking ResNets: improved stacking strategies with high order schemes. arXiv preprint arXiv:2103.15244 (2021)
31. Lars, R., Haber, E.: Deep neural networks motivated by partial differential equations. J. Math. Imag. Vis. 62(3), 352–364 (2020). https://doi.org/10.1007/s10851-019-00903-1
32. Devlin, J., et al.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
33. Dosovitskiy, A., et al.: An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
34. Li, S.Y., et al.: Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. Adv. Neural Inf. Process. Syst. 32, 5243–5253 (2019)
35. Lechner, M., Hasani, R.: Learning long-term dependencies in irregularly-sampled time series. arXiv preprint arXiv:2006.04418 (2020)
36. Vaswani A, A., et al.: Attention is all you need. Adv. Neural Inf. Process. Syst. 30, 6000–6010 (2017)
37. Zhu, C.B., et al.: Neural saliency algorithm guide bi-directional visual perception style transfer. CAAI Trans. Intell. Technol. 5, 1–8 (2020). https://doi.org/10.1049/trit.2019.0034
38. Pachitariu, M., Sahani, M.: Regularization and nonlinearities for neural language models: when are they needed? arXiv preprint arXiv:1301.5650 (2013)
39. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)

How to cite this article: Liu, M., et al.: Numerical-discrete-scheme-incorporated recurrent neural network for tasks in natural language processing. CAAI Trans. Intell. Technol. 1–10 (2023). https://doi.org/10.1049/cit2.12172