AUSTRALIAN JOURNAL OF INTELLIGENT INFORMATION PROCESSING SYSTEMS

Volume 17, No. 2
2019
Editorial Policy

The Australian Journal of Intelligent Information Processing Systems is an interdisciplinary forum for providing the latest information on research developments and related activities in the design and implementation of intelligent information processing systems.

The areas of interest include, but are not limited to:
- artificial intelligence
- artificial neural networks
- human centred computing
- fuzzy systems
- virtual reality

The journal both theoretical and application-oriented papers in diverse areas related to intelligent systems. The journal also publishes survey articles, short notes, and project reports.

Editorial Board

**Tom Gedeon**, Editor in Chief  
*Australian National University, Australia*

**Kevin Wong**  
*Murdoch University, Australia*

**James C. Bezdek**  
*University of Western Florida, USA*

**Tony Constantinides**  
*Imperial College of Science, Technology and Medicine, U.K.*

**Nikola Kasabov**  
*Auckland University of Technology, New Zealand*

**László Kóczy**  
*Budapest University of Technology & Economics, Hungary*

**Takeshi Furuhashi**  
*Nagoya University, Japan*

**Peter Eklund**  
*University of Wollongong, Australia*

**Marimuthu Palaniswami**  
*University of Melbourne, Australia*

**M. V. Srinivasan**  
*University of Queensland, Australia*

**A. Venetsanopoulos**  
*University of Toronto, Canada*

Responsibility of the content of the papers rests upon the authors and not with the publisher. Data and conclusions developed by the authors are for information only and are not intended for use without independent substantiating investigation on the part of the potential user.

The Australian Journal of Intelligent Information Processing Systems is published regularly. All correspondence including manuscripts and advertising inquiries should be addressed to the Editor in Chief: Professor Tom Gedeon, Research School of Computer Science, The Australian National University, Canberra ACT 2600, Australia. Alternatively, please email papers@ajiips.com.au.

Abstracting is permitted with due credit to the Australian Journal of Intelligent Information Processing Systems.

Article Processing Cost Recovery

AJIIPS is an open access journal, with cost recovery. On paper acceptance, an article processing charge of $200 is required, unless alternative arrangements are in place, e.g. for some special issues.
CONTENTS

SPIKING NEURON AND RELATED MODELS

A Ring-Type Digital Spiking Neural Network and Spike-Train Approximation
Hiroaki Uchida, Toshimichi Saito ................................................................. 1

Spatial Structure of Receptive Fields and Surrounds in Response to Local Figure-Ground Configuration in Monkey V4
Kouji Kimura, Yukako Yamane, Hiroshi Tamura, Ko Sakai................................. 7

Modelling Speaker-dependent Auditory Attention Using A Spiking Neural Network with Temporal Coding and Supervised Learning
Yating Huang, Jiaming Xu, Bo Xu .................................................................... 14

Simulation of multi-peak discharge Pattern Pattern Based on Improved Deterministic Chay Model
Zhongting Jiang, Dong Wang, Huijie Shang, Yuehui Chen .................................. 26

TIME-SERIES AND RELATED MODELS

Echo State Networks Composed of Units with Time-Varying Nonlinearity
Gouhei Tanaka, Ryosho Nakane, Akira Hirose .................................................. 34

Multilayer LSTM with Global Access Gate for Predicting Students Performance Using Online Behaviors
Shaofie Qu, Kan Li, Bo Wu, Kai Tang, Weimin Liu ........................................... 40

A Hybrid Time Series Model based on Dilated Conv1D and LSTM with Applications to PM2.5 Forecasting
Liqing Zhao, Bo Cheng, Junliang Chen ............................................................ 49

Dissect Sliced-RNN in Multi-Attention view
Zeping Yu, Linqing Shi, Gongshen Liu .................................................................. 61

Discovering Sequences in Systems Logs by Neural Networks
Jakub Nowak, Marcin Korytkowski, Marcin Wozniak, Rafal Scherer .................. 69

This is a refereed journal.  
Abstracted by INSPEC.

Publication Date December 2019
A Ring-Type Digital Spiking Neural Network and Spike-Train Approximation

Hiroaki Uchida and Toshimichi Saito
Hosei University, Koganei, Tokyo, 184-8584 Japan, tsaito@hosei.ac.jp

Abstract. This paper studies ring-type digital spiking neural networks that can exhibit multi-phase synchronization phenomena of various periodic spike-trains. First, in order to realize approximation of a class of spike-trains, a winner-take-all switching is applied to the network. Second, in order to design efficient networks, relationship between approximation error and the network size is investigated. Executing Verilog simulation, approximation function is confirmed experimentally.

Keywords: spiking neural networks, multi-phase synchronization, spike-train approximation.

1 introduction

This paper presents a network of digital spiking neurons (DSN [1][2]) and considers its application to time-series approximation. The DSN can be regarded as a digital version of analog spiking neurons that have been studied from both fundamental and application viewpoints [3][4]. Repeating integrate-and-fire behavior between a periodic base signal and constant threshold, the DSN can output a variety of periodic spike-trains (PSTs). Applying delayed ring connection to multiple DSNs, a ring-type digital spiking neuron (RDSNN) is constructed. The RDSNN can realize multi-phase synchronization of various PSTs [5][6]. Adjusting the base signal, stability of the synchronization can be reinforced. The PSTs and synchronization of them are applicable to various engineering systems including spike-based encoding communication [7], central pattern generators [8], and time series approximation [9][10].

In order to realize time-series approximation, winner-take-all (WTA) switching [4] is applied in the RDSNN where an objective time series is represented by a PST. This paper gives two main results. First, the WTA switching enables RDSNN to approximate a target PST automatically if the number of spikes in the PST satisfies some condition. Second, relationship between approximation error and the number of DSNs is investigated for typical examples of target PSTs. Executing Verilog simulation, typical synchronization phenomena and basic approximation performance are confirmed experimentally.

The results of the paper will be developed into systematic analysis of nonlinear dynamics in RDSNNs, optimal design of RDSNN for approximation of target PSTs, and its application to various systems including reservoir computing systems for time series approximation. As novelty of this paper, it should be noted that existing papers discuss neither the WTA-based switching nor the relationship between approximation error and the number of DSNs.

Fig. 1. DSN and digital spike map. (a) PST with period $6N_p$. (b) Periodic orbit with period 6 for $\delta = (5, 5, 5, 8, 8, 8, 17, 17, 2, 2, 2, 11, 11, 11, 14, 14, 14)$. 
2 Digital Spiking Neurons

We introduce the DSN that is a building block of the RDSNN. Let \( x(\tau) \) denote a discrete state variable at discrete time \( \tau \). Repeating integrate-and-fire behavior between a base signal \( b(\tau) \) with period \( N_p \) and a constant threshold \( N_x \), the DSN outputs a spike-train \( y(\tau) \).

\[
\begin{align*}
\text{Integrating:} & \quad x(\tau + 1) = x(\tau) + 1, \quad y(\tau) = 0 \text{ if } x(\tau) < N_x \\
\text{Self-firing:} & \quad x(\tau + 1) = b(\tau), \quad y(\tau) = 1 \text{ if } x(\tau) = N_x \\
\end{align*}
\]  

where \( x(\tau) \in \{0, 1, \ldots, N_x\} \) and \( b(\tau + N_p) = b(\tau) \). Fig. 1 (a) illustrates the dynamics. For simplicity, we assume the following condition.

\[
\tau - 2N_p + 1 \leq b(\tau) - N_x \leq \tau - N_p \text{ for } \tau \in \{0, \ldots, N_p - 1\}, \quad N_x \leq 2N_p - 1.
\]

In this case, the DSN outputs one spike per one period of \( b(\tau) \) and outputs a spike-train

\[
y_o(\tau) = \begin{cases} 
1 & \text{for } \tau = \tau_n \\
0 & \text{for } \tau \neq \tau_n 
\end{cases} \quad \tau_n \in I_n = [(n - 1)N_p, nN_p)
\]

where \( \tau_n \) denote the \( n \)-th spike-position. Let \( \theta_n = \tau_n \mod N_p \) \((\theta_n \in \{1, \ldots, N_p\})\) be the \( n \)-th spike-phase. A spike-position is given by \( \tau_n = \theta_n + N_p(n - 1) \) and a spike-train \( y(\tau) \) is governed by the digital spike map \( F \).

\[
\theta_{n+1} = F(\theta_n) = f(\theta_n) \mod N_p, \quad f(\theta_n) = \theta_n - b(\theta_n) + N_x + 1
\]

The digital spike map is represented by a characteristic vector \( \delta \) of integers:

\[
\delta \equiv (\delta_1, \ldots, \delta_{N_p}), \quad F(i) = \delta_i, \quad \delta_i \in \{1, \ldots, N_p\}, \quad i \in \{1, \ldots, N_p\}
\]

Fig. 1 (b) shows an example of digital spike map with periodic orbit with period 6. This periodic orbit is stable and corresponds to PST with period \( 6N_p \) \((y_o(\tau + 6N_p) = y_o(\tau))\) in Fig. 1 (a). Adjusting the base signal, the DSN can generate various stable PSTs. More detailed discussion of periodic orbits and their stability can be found in [2] [6].

3 Ring-coupled Digital Spiking Neural Networks

Connecting \( M \) pieces of DSNs with a common base signal \( b(\tau) \) in ring topology, the RDSNN is constructed (see Fig. 2). The dynamics is described by

\[
\begin{align*}
\text{Integrating:} & \quad x_i(\tau + 1) = x_i(\tau) + 1, \quad y_i(\tau) = 0 \text{ if } x_i(\tau) < N_x \\
\text{Self-firing:} & \quad x_i(\tau + 1) = b(\tau), \quad y_i(\tau) = 1 \text{ if } x_i(\tau) = N_x \\
\text{Cross-firing:} & \quad x_{j+1}(\tau + 1) = N_x - N_p + 1, \quad z_j(\tau) = 1 \text{ if } x_j(\tau) = N_x \quad \text{and} \quad x_{j+1}(\tau) \leq N_x - N_p \\
\text{Connection signal:} & \quad z_i(\tau) = \begin{cases} 
1 & \text{if } x_i(\tau) = N_x \text{ and } x_{i+1}(\tau) \leq N_x - N_p \\
0 & \text{otherwise} 
\end{cases}
\end{align*}
\]

The integrating and self-firing are the same as the single DSN. Fig. 3 shows the cross-firing that connects DSNs in ring topology. The cross-firing is characterized by the connection signal \( z_i \). For simplicity, we consider the case where each DSN outputs a PST with period \( MN_p \) and the RDSNN consists of \( M \) pieces of DSNs. We define the \( M \)-phase synchronization (M-SYN) with period \( MN_p \).

\[
\begin{align*}
x_i(\tau) &= x_i(\tau + MN_p), \quad y_i(\tau) = y_i(\tau + MN_p), \quad i \in \{1, \ldots, M\} \\
x_j(\tau) &= x_{j+1}(\tau + N_p), \quad y_j(\tau) = y_{j+1}(\tau + N_p), \quad j \in \{1, \ldots, M\} \\
z_i(\tau) &= 1 \text{ for some } \tau \in \{1, \ldots, MN_p\}
\end{align*}
\]

where \( x_{M+1} \equiv x_1 \) and \( y_{M+1} \equiv y_1 \). Note that non-zero connection signal \( z_i(\tau) \) is required for the M-SYN because all the DSNs are isolated if \( z_i(\tau) = 0 \) for all \( \tau \). Fig. 4 illustrates an M-SYN of PSTs with period \( MN_p \) for \( M = 6 \) and corresponding output spike-train of the RDSNN. The existence and stability of M-SYN is discussed in [6].

The output spike-train \( y \) is given through time dependent selection switches \( S_i \) operation of which is represented by the connection matrix \( W = (w_{ij}) \):

\[
y(\tau) = \sum_{i=1}^{M} w_{ij} y_i(\tau) \text{ for } \tau \in I_j, \quad j \in \{1, \ldots, M\}, \quad y(\tau + T) = y(\tau)
\]
Fig. 2. RDSNN for $M = 6$.

$$w_{ij} = \begin{cases} 
1 & \text{if the } i\text{-th DSN is selected } (S_i = \text{on}) \text{ for } \tau \in I_j \\
0 & \text{if the } i\text{-th DSN is not selected } (S_i = \text{off}) \text{ for } \tau \in I_j 
\end{cases} \quad (10)$$

where one period of PST ($I \equiv [0, MN_p]$) is divided into $M$ subintervals

$$I_1 = [0, N_p), I_2 = [N_p, 2N_p), \ldots, I_M = [(M-1)N_p, MN_p).$$

and $w_{ij}$ is constant in each subinterval. If the $i$-th DSN is selected in the $j$-th subinterval $I_j$ ($w_{ij} = 1$) then $I_j$ is said to be activated by $S_i$.

Since possible connection number of DSNs is zero to $M$ in each subinterval and since each DSN outputs one spike in each subinterval, the RDSNN can output zero to $M$ spikes in each subinterval. Fig. 4 shows an example of output $y$ for $M = 6$ where the selection is given by the following selection matrix and the PST of the first DSN is given by Fig. 1.

$$W = \begin{pmatrix} 
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 
\end{pmatrix} \quad (11)$$

The output PST is characterized by six spike-phases: $\{\theta_a, \theta_b, \theta_c, \theta_d, \theta_e, \theta_f\}$. Adjusting the selection matrix, the RDSNN can output various PSTs consisting of any combination of 6 spike-phases $\{\theta_a, \theta_b, \theta_c, \theta_d, \theta_e, \theta_f\}$. In general, if a DSN outputs a PST with period $MN_p$ consisting of $M$ spike-phases then the RDSNN can output various PSTs consisting of any combination of the $M$ spike-phases. It goes without saying that such an output is impossible in the single DSN [5].
4 Spike-train Approximation

We apply the RDSNN to spike-train approximation. First, we define a target PST $y_k(\tau)$ with period $T = MN_p$:

$$y_k(\tau) = \begin{cases} 1 & \text{for } \tau = \tau_k \\ 0 & \text{otherwise} \end{cases}$$

$k \in \{1, \ldots, Q\}, \ 0 < \tau_1 < \cdots < \tau_Q < T \tag{12}$

where $\tau_n$ denotes the $n$-th spike-position and $y_k(\tau + T) = y_k(\tau)$. The target PST consists of $Q$ spikes per one period. For convenience, let the target spike-train be represented by inter-spike intervals (ISIs)

$$D = (d_1, d_2, \cdots, d_{Q-1}), \ d_l = \tau_{l+1} - \tau_l, \ l \in \{1, \cdots, Q - 1\} \tag{13}$$

where $d_l$ is the $l$-th ISI and $D$ is the target ISI sequence.

Here we consider approximate of a target PST by the output of the RDSNN. In order to realize the approximation, suitable operation of the selection switches $S_i$ is necessary. We present the WTA switching to determine the selection matrix for the suitable operation of $S_i$:

$$w_{ij} = \begin{cases} 1 & \text{if } y_k(\tau) = 1 \text{ and } x_i(\tau) \text{ is the maximum at time } \tau (x_i(\tau) > x_k(\tau), k \neq j) \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where $\tau \in I_j$. Fig. 5 illustrate the WTA switching. This switching tries to select a DSN having the highest approximation potential when the target spike arrives.

If the WTA switching selects the $i$-th DSN at the $j$-th subinterval $I_j$ ($w_{ij} = 1$) then the $I_j$ is said to be activated by $S_j$. As an target PST $y_k(\tau)$ is applied, the RDSNN with the WTA switching outputs an
approximated PST $y_a(\tau)$ where $0 \leq \tau < T = MN_p$. For simplicity, we consider the case where $y_t(\tau)$ and $y_a(\tau)$ consist of the same number of spikes. The approximated PST $y_a$ is characterized by an ISI sequence $D' = (d'_1, d'_2, \cdots, d'_{Q-1})$. The approximation accuracy is evaluated by the matrix of ISI error:

$$\varepsilon_p = \frac{1}{Q-1} \sum_{i=1}^{Q-1} |d_i - d'_i|. \quad (15)$$

If the WTA misses some target spike then the ISI error is calculated after removing the missing spike(s). The number of the missing spikes is used as the other evaluation matrix.

We have investigated the approximation function in Verilog simulation. The Verilog simulation is a first step to realize a utility hardware. Fig. 6 shows the target PST characterized by $D = (21, 11, 9, 7, 6, 7, 6, 9)$ and approximated PST from the RDSNN with the WTA switching. In the RDSNN, the selection matrix is given by

$$W = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix} \quad (16)$$

The target PST is given by discrete exponential distribution discussed afterward. In the Verilog simulation, the RDSNN is constructed by shift registers, D flip-flops and several switching elements. The basic circuit design can be found in [6]. The approximation is evaluated by $\varepsilon_p = 0.5$.

![Fig. 6. Target spike-train and approximated spike-train in Verilog simulation.](image)

![Fig. 7. Discrete exponential distribution ($d_{min} = 6, \lambda = 0.25$).](image)

In order to consider the approximation function in more detail, we have performed fundamental numerical experiments. For simplicity, we consider the approximation of target PST with period $T = 6N_p$ by RDSNN.
of 6 DSNs ($M = 6$). The target PST is given by discrete exponential distribution $f(d)$ as shown in Fig. 7 where $d$ is a random variable corresponding to ISI. If the firing frequency of spikes per subinterval follows the Poisson distribution, the firing interval of spikes (ISI) follows the exponential distribution. In the exponential distribution, we have fixed parameters as $d_{\text{min}} = 6$ and $\lambda = 0.25$: in some subinterval $t_n$, 0 to 3 spikes can appear. In order not to missing spikes, the number of DSNs must be 3 or more ($M \geq 3$). We have investigated influence of the number of DSNs in the following two cases.

Case 1: The number of DSN is reduced in the order of activation frequency in the subinterval selection. For example, if the $j$-th DSN has the lowest activation frequency then the $j$-th DSN is disconnected ($S_j = \text{off}$) from the output.

Case 2: The number of DSN is reduced randomly.

We have executed 10 trials and the results are measured by average of ISI error ($\text{Avg of } \epsilon_p$), standard deviation of ISI error ($\text{SD of } \epsilon_p$), and spike missing rate (SMR). In Tables 1 and 2, we can see that if the number of DSNs is 6, the approximation property of almost the same in the Case 1 and Case 2. As the number of DSNs decreases, the Case 1 exhibits better approximation performance than Case 2 in both ISI error and spike missing rate. Especially, in the case of three DSNs, the Case 1 exhibits much better approximation performance than the Case 2. These results suggest that existence of an optimal combination of DSNs for efficient approximation and the WTA switching is effective to determine the selection matrix.

### Table 1. Results in Case 1.

<table>
<thead>
<tr>
<th>#DSN</th>
<th>AVG of $\epsilon_p$</th>
<th>SD of $\epsilon_p$</th>
<th>SMR [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.96</td>
<td>0.18</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>1.08</td>
<td>0.22</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>1.20</td>
<td>0.28</td>
<td>0.94</td>
</tr>
<tr>
<td>3</td>
<td>1.59</td>
<td>0.69</td>
<td>0.94</td>
</tr>
</tbody>
</table>

### Table 2. Results in Case 2.

<table>
<thead>
<tr>
<th>#DSN</th>
<th>AVG of $\epsilon_p$</th>
<th>SD of $\epsilon_p$</th>
<th>SMR [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.96</td>
<td>0.18</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>2.01</td>
<td>0.79</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>2.21</td>
<td>0.60</td>
<td>7.55</td>
</tr>
<tr>
<td>3</td>
<td>2.88</td>
<td>0.78</td>
<td>11.3</td>
</tr>
</tbody>
</table>

5 Conclusions

The RDSNN is presented and its application to spike-train approximation is considered in this paper. The RDSNN can realize stable multi-phase synchronization of various PSTs. Applying the WTA switching to selection of suitable DSNs, the RDSNN can approximate target PSTs if the number of spikes per period of base signal does not exceed the number of DSNs. The relation between approximation error and the number of DSNs are investigated and basic information for efficient network design is given. Presenting a Verilog simulation, basic approximation function is confirmed experimentally.

Future problems include analysis of the optimal combination of DSNs for spike-train approximation, implementation of RDSNN on an FPGA board and development into large scale spike-based digital reservoir computing system.

References

Spatial Structure of Receptive Fields and Surrounds in Response to Local Figure-Ground Configuration in Monkey V4

Kouji Kimura¹, Yukako Yamane², Hiroshi Tamura² and Ko Sakai¹

¹ Department of Computer Science, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

² Graduate School of Frontier Biosciences, Osaka University
1-4 Yamadaoka, Suita-shi, Osaka 560-0871, Japan

sakai@cs.tsukuba.ac.jp
http://www.cvs.cs.tsukuba.ac.jp/

Abstract. The visual system constructs a surface as a proto-object by segregating scenes into figure and ground (FG). We investigated what spatial structure of FG evokes neural responses that correspond to a surface or figure. We proposed a new method based on the spike-triggered average for the estimation of the optimal FG organization of neurons rather than the conventional luminance-based optimal stimulus. The estimated optimal FG organization represents the receptive-field structure in response to figure and ground (FG-RF). We applied this method to the neural responses recorded from monkey V4 while presenting a number of natural images. The computed FG-RFs exhibited an antagonistic structure with the preferred region in and around the classical receptive field (CRF) and the non-preferred region in the surround of the CRF. The normalized mean FG-RF across neurons showed a concentric antagonistic structure with the extent up to several times larger than the CRF. We showed, for the first time, that individual neurons in V4 code the local structure of FG. Our results also suggested that a few, up to several tens of, V4 neurons are capable of representing the FG structure of objects.

Keywords: visual cortex, V4, receptive field, figure-ground, perceptual organization

1 Introduction

The visual system decomposes scenes into pixels in the retina, and gradually constructs a model of the outside world in the cortex from the pixels. The formation of surfaces from pixels plays a crucial role in the construction of scenes with which perceptual organization is a set of phenomenological rules that govern the formation of surfaces. A surface has been considered to represent a proto-object that is a basis for the representation of an object [1], which could be further refined by knowledge and memory to enable object recognition. Neurons in V2 exhibit the selectivity to border ownership (BOS) which tells the direction of figure from the border. A spike-triggered stimulus averaging (STA) has estimated their spatial characteristics of the receptive fields [8]. Although BOS tells the direction of figure along a contour, the neural mechanism underlying the construction of surface and the perceptual organization of figure and ground have not been clarified. Physiological studies have suggested that the neurons in the intermediate-level visual cortical areas, such as V4, are sensitive to the organization of figure and ground (FG) [2, 3]. However, the neural mechanisms underlying the preference to FG have not been revealed. It has not been even clarified what spatial organization of images actually evokes the FG-dependent responses.

We investigated what spatial structure and extent of FG evoke the FG-dependent responses in V4 neurons. We have recently reported that neurons in monkey V4 exhibit FG-dependent responses to natural image patches [4]. Here, we analyzed these neurons with FG-dependent responses in order to estimate the spatial extent and structure of their classical receptive fields (CRFs) and surrounding regions in response to FG. An established method for estimating the structure of RF is STA which is the reverse correlation between white-noise stimuli and the evoked spikes [5]. A conventional STA has revealed the luminance-based receptive field structure of V1 simple cells [5]. However, the conventional STA estimates an optimal luminance organization, but not optimal FG organization. We proposed a new method to estimate the optimal spatial organization of FG. Specifically, we applied STA to the responses to a variety of natural images with the veridical FG-organization obtained from human psychophysics. The estimated optimal FG organization exhibited a preferred region around the CRF center with the extent of the entire CRF, and a non-preferred region that extended beyond the CRF with the extent up to the limit of the present measurement (three times larger than the CRF extent).

2 Experimental Design and Stimuli

Electrophysiological recordings from the visual area V4 of two macaque monkeys (Macaca fuscata) were conducted from the Laboratory for Cognitive Neuroscience of Osaka University [4]. During experiment sessions, the monkeys were analgesized and immobilized. Spiking activities of single neurons were isolated from signals recorded using a 32-channel
single-shank electrode. All animal experiments were performed in accordance with the guidelines of the National Institute of Health (1996) and the Japan Neuroscience Society, and were approved by the Osaka University Animal Experiment Committee (certification no: FBS-13-003).

Stimuli presented to the monkeys were comprised of two sets of small patches, which were generated from natural images included in Berkeley Segmentation Dataset (BSD) [6], as similar to our previous experiment [4]. Example stimuli are shown in Figure 1. The first set consisted of natural image patches (natural patches). We chose 105 sub-regions (69×69 pixels) from the BSD that included the contours passing through the center of the patches. Because the distribution of contour curvatures is highly non-uniform in natural scenes, we controlled the distributions of the degree of convexity, closure, and symmetry of contours [7]. The second set consisted of natural contours with one side filled with the preferred color of the cell (either white, black, red, green, yellow, or blue) and the other with the opposite color (filled patches). The color-inverted stimuli were also included in the set. We used Human Marked Contour (HMC) available in BSD that were drawn by 10 human participants. We also used mirrored images of the natural and filled patches (a total of 630 patches: 210×2+105×2). The images were mirrored with respect to the tangent of the contour at the patch center. The colors were also inverted in the mirror images so as to keep contrast polarity. To obscure the boundary of stimulus and background, we attenuated contrast towards the periphery with a Gaussian. A few examples are shown in Figure 1. The patch stimuli were scaled to cover the classical receptive fields (CRFs) of the cell under recording, more than three times larger than the rough estimate of the CRF diameter, yielding a stimulus size between 2.5 and 20 degrees. The stimuli were presented for 200ms with a blank interval of 200ms in random order with 10 repetitions.

3 Results

The recording data included the responses of 1118 neurons of which 281 neurons showed significant visual response and their CRF center fell onto the stimulus patches. We tested whether these 281 neurons exhibited significantly different responses to figures and grounds (FG significant; ANOVA p<0.05). Sixty-eight and 39 neurons showed FG significance with the filled and natural patches, respectively. Among them, 16 neurons showed FG significance to both filled and natural patches. We examined these 107 FG-significant neurons in the following analyses. The distributions of the CRF centers of the neurons were shown in Figure 2.

3.1 Estimation of Receptive-Field Structure in response to Figure-Ground Configuration by Spike-triggered Average

Spike-triggered average (STA) is an established method for estimating the spatial structure of linear receptive fields by computing the reverse correlation between a stimulus (luminance) and the number of spikes observed shortly after the stimulus presentation [5]. Specifically, the presented stimuli were averaged with weights corresponding to the number of spikes. Here, we tried to estimate the FG configurations, rather than the luminance, that evoked the strongest responses.
to single neurons. The receptive field structure corresponding to figure and ground (FG-RF) cannot be estimated by the conventional STA that average stimulus luminance. To estimate FG-RF, we propose to compute the reverse correlation between the number of spikes and the FG configuration of the stimulus. Specifically, we provided FG-labeled images where the stimulus was binarized with regard to figure and ground (+1 and -1 for figure and ground, respectively), and computed their weighted average based on the recorded spikes. FG-RF is thus defined by:

$$\text{FG-RF} = \left( \frac{\sum_{i} \# \text{spike}(i) \times \text{Stim}_{FG}(i)}{\sum_{i} \# \text{spike}(i)} - \frac{\sum_{i} \text{Stim}_{FG}(i)}{N} \right) / N.$$ 

$\# \text{spike}(i)$ and $\text{Stim}_{FG}(i)$ are the number of spikes and the FG-labeled image of the $i$-th stimulus, respectively. $N$ is a total number of stimulus. The spatial distribution of FG across FG-labeled images was almost uniform but not perfectly, thus, we compensated this non-uniformity by subtracting the simple ensemble average of FG-labeled images from the computed FG-STA divided by the total number of spikes. This subtraction will result in a zero FG-RF if a neuron exhibited the same number of spikes for each stimulus. A similar method has been applied to estimate the receptive field structure of border ownership-selective neurons in V2 [8]. Note that regions, where neither figure or ground can be assigned by the contour passing through the patch center, were labeled as zero in the FG-labeled images, and that the periphery of the images was blurred by a Gaussian. Because the location of the CRF center varies across neurons, STA for each neuron was computed with its center matched with the CRF center of the neuron. The veridical labels of FG were given by human psychophysical experiments [7].

The responses of an example neuron to a variety of stimulus patches are shown in Figure 3. This example neuron was FG significant for filled and natural patches, and preferred ground. A number of patches that evoked strong responses had a ground region around the CRF center (See the figure caption for details). This tendency is observed in the responses to both filled and natural patches. The estimated FG-RFs for this example neuron are shown also in Figure 3. The FG-RFs for filled and natural patches show similar structures, indicating that this neuron signals figure and ground independent of the image characteristics in the patches such as color and texture. The FG-RFs exhibit an antagonistic structure; a ground-responsive region (suppressive for the figure) around the CRF center and a figure-responsive region (facilitative) above the ground-responsive region. Most of FG significant neurons showed similar antagonistic structures with the region responsive to the preferred side (figure or ground) around the CRF center and that to the non-preferred side far from the center.

**Figure 2.** The distribution of the CRF centers of the neurons being analyzed, superimposed onto an example stimulus. Red and blue crosses show the CRF centers of figure- and ground-preferred neurons, respectively. The left and right panels show the distributions of the neurons that showed significantly different responses to FG in filled and natural stimuli, respectively. The CRF centers were distributed across the extent of the patches.
Figure 3. The responses and estimated FG-RF of an example neuron. Patches in the top blocks show the presented stimuli in rank order of responses. The contrast of patches is proportional to the response (spike/s). The left and right blocks include filled and natural patches. The middle block shows the FG organization of the patches with black and white being figure and ground, respectively. The bottom panels show the estimated STA (FG-RF) for the filled (left) and natural (right) patches. Bluish and reddish colors show the ground- and figure-preferred regions, respectively. Squares with dotted lines indicate the estimated extent of the CRF. We observe that this cell had an antagonistic structure with a ground-preferred region on and around the CRF and a figure-preferred region in the periphery.
3.2 Population Analysis of Figure-Ground Receptive-Field

The estimated FG-RFs showed antagonistic structures; a preferred region around the CRF center and a non-preferred region aside. However, this linear antagonistic structure might be evoked by the constraints in stimulus dimensions (small patches) and contour location (passing through the stimulus center, not the CRF center). For instance, if the patch centers were projected onto the CRF center, a circular preferred region would appear around the CRF center, and a concentric non-preferred region would appear in the periphery of the preferred region. On the other hand, if the patch centers were projected onto the bottom of the CRF center, no stimulus was projected onto to the top, and thus the non-preferred region should appear only in a lower region. Because of these constraints, our estimates of FG-RF for single neurons could not fully represent the structure. To overcome this possible distortion, we computed the mean FG-RF across neurons with normalization that compensates this distortion.

The direction of an estimated non-preferred region with respect to the CRF center would vary depending on the direction of the CRF center with respect to the patch center. The distribution of the direction of the CRF centers with respect to the patch was not uniform across the recorded neurons. Thus, we computed the ensemble average of FG-RFs across neurons with the normalization based on the distribution of the direction of the CRF center. The computed mean FG-RFs for the figure- and ground-preferred neurons are shown in Figure 4. We observe a preferred region around the CRF center, and a concentric non-preferred region in the periphery of the preferred region. The non-preferred region extending beyond the CRF indicates a crucial role of surround suppression. The extent of the non-preferred region was at least three times larger than that of the CRF. Note that the extent of the patch was set to three times larger than the CRF. It may be plausible that non-preferred regions extend beyond this range.

4 Discussions

We investigated the optimal spatial structure of figure and ground (FG-RF) that evoked the FG-dependent responses in V4 neurons. We proposed a new method for estimating the optimal spatial organization of FG by taking the reverse correlation between the number of spikes and the FG configuration of the stimulus. We showed, for the first time, that FG-RFs exhibited antagonistic structures with a preferred region in and around the CRF and a non-preferred region in the periphery of the preferred region. The normalized ensemble average of FG-RFs across neurons showed a concentric structure with a preferred region around the center and a non-preferred region in the periphery. The regions extending beyond the CRF indicate a crucial role of surround modulation [9]. These results indicate that individual neurons in V4 are capable of detecting local FG within the extent of several times larger than their CRFs. This result further suggests that a small number of V4 neurons, perhaps a few or up to several tens, are capable of representing FG even for natural objects, supporting a population coding of FG in the intermediate-level visual area.
Figure 4. The estimated mean FG-RFs across neurons with figure- (A) and ground-preference (B). (A) The top row shows the normalized mean FG-RFs estimated from the filled (left) and natural patches (right). The bottom row shows the Mexican-hat Gaussians that best fit the estimated FG-RFs. The ellipses below the 3D Gaussian plots indicate the 2SD of the excitatory (figure-preferred) Gaussian. (B) The same conventions as (A) except for the ellipses at the bottom indicating the 2SD of the inhibitory (ground-preferred) Gaussian.
Acknowledgement

This work was supported by grants-in-aid from JSPS (KAKENHI, 26280047, 17H01754) and RIEC, Tohoku University (H31/A64) awarded to K.S., (KAKENHI, 16K01962, 71J40112) awarded to Y.Y., and from MEXT, Japan (SHITSUKAN JP15H05921) awarded to H.T.

References

3. Hasuike, M., et al., *Representation of Local Figure-Ground by a Group of V4 Cells*. Lecture notes in computer science, 2016. 9947: p. 131-137.
Modelling Speaker-dependent Auditory Attention Using A Spiking Neural Network with Temporal Coding and Supervised Learning

Yating Huang$^{1,2}$, Jiaming Xu$^1$ *, and Bo Xu$^{1,2,3}$

$^1$ Institute of Automation, Chinese Academy of Sciences, Beijing, China
$^2$ University of Chinese Academy of Sciences, Beijing, China
$^3$ Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China

{huangyating2016, jiaming.xu, xubo}@ia.ac.cn

Abstract. Spiking Neural Networks (SNNs) are regarded as the third generation of neural network models, which can learn the precise spike trains of the stimuli. As speech signals exhibit strong temporal structure, SNNs are a natural choice for learning temporal dynamics of the speech. Therefore, we propose a unified biologically plausible framework using spiking neurons with temporal coding and supervised learning to solve the auditory attention problem. We further introduce momentum and Nesterov’s accelerated gradient into the Remote Supervised Method to improve the performance and speed up the spike train learning. We evaluate our model on Grid corpus and demonstrate that our model performs a precise spike train coding for auditory attention and outperforms the baseline artificial neural networks.

Keywords: Cocktail Party Problem · Temporal coding · Remote supervised method.

1 Introduction

Attending to a specific speaker in a multi-talker environment is a trivial task that humans perform routinely, but remains a fundamental challenge for the automatic speech recognition community. The general problem of separating the target auditory signal from the competing input streams is referred to as Cocktail Party Problem [2], which has attracted the interests of many researchers, as the underpinnings of which will contribute to efficient front-end processors of automatic speech recognition. However, the exact neural mechanisms underlying the process are still unclear.

Conventional artificial neural networks encode the stimuli using rate coding, which is a traditional coding scheme assuming that the intensity of the stimuli is encoded by the average number of spikes within the encoding window. However, the temporal structure is neglected in rate coding and a growing number of experimental results have suggested that straightforward mean rate codes may be too simple to describe the complex brain activity [8]. When extra information is carried by the precise temporal structure of a spike train, it’s referred to as temporal coding [14]. Recent studies suggest that spike times might turn out to be general units of the sensory representation in various systems of the brain, including auditory and visual systems [23, 6]. Spiking neural networks (SNNs) [11] consider the spike timings of spike trains and thus are a natural choice to encode and learn the temporal structures of complex sensory signals in speech perception. Nevertheless, there is little research concerning the application of SNNs to Cocktail Party problem. Prior studies of application of SNNs use unsupervised learning and have no training phase [26, 15, 25]. The core of these models are a two-layer oscillator network. Sources are separated in terms of oscillator synchrony and desynchrony. However, they only work on isolated speech or simple cases, for example, separating the mixture of two isolated human voice /di/ and /da/, which is impractical for real-world scenarios. By formulating Cocktail Party Problem as a supervised learning problem, the discriminative patterns of speech can be learned from the training corpus [27].

Besides, results of dichotic listening research support that humans are not able to listen to and remember two concurrent speech, while they selectively attend to the target speech and ignore sounds from other sources in the mixed signal [2, 13]. Obviously, this is more efficient and practical in the complex auditory scene, whereas previous approaches mentioned above attempt to separate all the sources rather than attend to the target. Therefore, we are

* Corresponding author.
inspired to use an SNN with temporal coding and supervised learning from the perspective of speaker-dependent auditory attention to solve Cocktail Party Problem.

Remote Supervised Method (ReSuMe) is a supervised learning algorithm proposed for SNNs that aims to produce the desired precise time sequences of spikes. ReSuMe is proved to be an efficient supervised learning rule for SNNs, which is greatly inspired by the biological mechanisms involved in learning and plasticity [16, 17]. ReSuMe makes no assumption of the neuron model, thus it could work independently of any neuron model. In the following work, ReSuMe is extended to apply to multi-layer SNNs from an alternative motivation when the neuron models are restricted to linear stochastic neuron models [20], which reveals the relationship between ReSuMe and Stochastic Gradient Descent (SGD). This allows us to consider introducing momentum and Nesterov’s accelerated gradient [21] into ReSuMe learning rule to improve the performance and convergence of the learning process.

In this work, we propose a unified spiking neural network framework using temporal coding and supervised learning for auditory attention. The main contributions of our work are three-fold: (1) To the best of our knowledge, this is the first time that an SNN with temporal coding and supervised learning has been applied to auditory attention. (2) We propose two novel temporal coding schemes to generate spatiotemporal patterns for auditory attention task and compare the performance of different coding schemes. (3) We introduce momentum and Nesterov’s accelerated gradient to accelerate the learning process of ReSuMe.

![Fig. 1](image.png)

**Fig. 1.** The overall structure of our proposed framework. Our model uses different temporal coding schemes to transform the stimuli into spike trains that a spiking neural network can handle. Then the spiking neural network learns the mapping of the mixture input spike trains to the target output spike trains. Finally the target signal is reconstructed according to the output spike trains.

2 Methods

In this section, we attempt to apply a spiking neural network to speaker-dependent auditory attention task. We described the model architecture, along with our proposed temporal coding schemes and the improved ReSuMe learning rule.
2.1 Model Architecture

Figure 1 shows the overall structure of the proposed framework. The framework contains three parts: temporal coding, supervised learning and attentive tuning.

The temporal coding part converts the audio signal into spike trains so that an SNN can handle it. Given a raw mixture input \( x \), the model first transforms the signal into time-frequency domain \( X_{t,f} \) using Short-Time Fourier Transformation (STFT). In order to ignore T-F bins that are not important to the source, T-F bins are maintained only when they are greater than some background noise threshold. Then we perform \( K \)-Means clustering to map the original continuous intensity information to \( D \) intensity levels. To further make the representation sparse and speed up the learning process, T-F bins with the lowest intensity level are set to be silence units. Then we map the remaining T-F bins to spike trains using the temporal coding schemes.

The supervised learning part is a feed-forward SNN. The neuron model is Leaky Integrate-and-Fire (LIF) model, whose spike response function can be simply defined as \( \alpha \)-function:

\[
\epsilon(t) = \frac{t}{\tau} \exp\left(1 - \frac{t}{\tau}\right) H(t),
\]

where the time constant \( \tau \) determines the rise and decay time of Postsynaptic Potential (PSP) and \( H(t) \) is the Heaviside function. The membrane potential of the neuron is reset to the resting potential and held for the refractory period \( \tau_{\text{ref}} \) once it emits a spike. The number of input neurons \( m \) is determined by temporal coding schemes and the frequency dimension \( F \) of \( X_{t,f} \), while the number of output neurons \( n \) is solely determined by \( F \). Note that \( m \) is the same as or \( D - 1 \) times as many as \( n \) according to different temporal coding schemes. ReSuMe is used to modify the synaptic weights.

In the attentive tuning part, the output spike trains are converted to an Ideal Binary Mask (IBM) as the attention map to tune to the target signal in the mixture speech and produce the attended spectrogram. Finally, the attended signal is reconstructed by inverse Short-Time Fourier Transformation (iSTFT).

2.2 Temporal Coding

![Fig. 2. Spikes of neuron i encoding intensity level 2 at encoding window \( t_0 \) and intensity level 3 at encoding window \( t_1 \) when \( D = 4 \). \( P_d \) denotes the \( d_{th} \) neuronal population.](image)
Besides rate coding and temporal coding, population coding is another important coding scheme found in neuroscience. Population coding assumes that the information is encoded by the joint activities of a number of imprecise neurons rather than single precise neurons [14]. Inspired by neural coding mentioned above, we propose two novel temporal coding schemes to generate spatiotemporal spikes for the auditory attention task, Temporal-Rate coding and Temporal-Population coding. Temporal-Rate coding is a coding scheme that combines the ideas of rate coding and temporal coding, while Temporal-Population coding is a coding scheme inspired by the ideas of population coding and temporal coding. The intensity information is represented by the number of spikes for Temporal-Rate coding and Temporal-Population coding. The difference between Temporal-Rate coding and Temporal-Population coding is that Temporal-Rate coding distributes the spikes over the shift time of the sliding encoding window, while Temporal-Population coding distributes the spikes over different neuronal populations. For example, suppose that the intensity level \( D \) is set to be 4, then there will be 3 neuronal populations for Temporal-Population coding because the T-F bins with the lowest intensity level are set to be silent, while there is only one neuronal population for the conventional Time-to-First-Spike [22] and our proposed Temporal-Rate coding. Now we have 3 intensity levels, namely 1, 2, 3. Assume that shift time of the sliding encoding window is 6 ms, and the onset of the current time window is \( t_0 \) ms. If a specific T-F bin corresponds to the current encoding window indexed by \( t_0 \) of neuron \( i \) in the neuronal population and its intensity level is 2, then for the Time-to-First-Spike case, neuron \( i \) will fire at \((t_0 + 3)\) ms during the current encoding window; for the Temporal-Rate coding case, neuron \( i \) will fire 2 spikes at \( t_0 \) ms and \((t_0 + 3)\) ms, respectively; for the Temporal-Population coding case, neuron \( i \) of the first two neuronal populations will fire a spike at the onset \( t_0 \) ms of the current time window respectively, while the corresponding time window of the third neuronal population will stay silent. Suppose that the following encoding window \( t_1 \) encoding a T-F bin with intensity level 3, neuron \( i \) will fire at \((t_1 + 1.5)\) ms for Time-to-First-Spike, and will fire 3 spikes at \( t_1 \) ms, \((t_1 + 2)\) ms, \((t_1 + 4)\) ms for Temporal-Rate coding; for Temporal-Population coding, neuron \( i \) of all the three population will fire at \( t_1 \) ms. See Figure 2 for an illustration.

2.3 Accelerating ReSuMe Learning

Consider a fully connected feed-forward SNN with two layers of stochastic linear neurons, identified as \( I \) (Input) and \( O \) (Output). The synaptic weight update between output neuron \( o \in O \) and input neuron \( i \in I \) according to ReSuMe learning rule is given as:

\[
\Delta w_{oi}(t) = [S^d_o(t) - S^a_o(t)](a + \int_0^\infty W(s)S_i(t - s)ds),
\]

(2)

where \( S^d_o(t), S^a_o(t) \) and \( S_i(t) \) denote the desired output spike train, actual output spike train and input spike train, respectively; \( a \) represents the non-Hebbian term; \( W(s) \) defines the learning window:

\[
W(s) = \begin{cases} 
A \cdot \exp\left(\frac{-s}{\tau_{win}}\right), & \text{if } s > 0 \\
0, & \text{if } s \leq 0 
\end{cases}
\]

(3)

where \( s \) is the time difference of the postsynaptic and presynaptic firings, \( A > 0 \) is the amplitude and \( \tau_{win} \) is the time constant of the learning window. For simplicity, the learning window is set to zero when presynaptic spikes fire after postsynaptic spikes, because the anti-causal order of spikes contributes nothing to the success of learning [16,17]. The total weight change \( \Delta w_{oi} \) is given by integrating Eq. (2) over the time course.

Inspired by the successful practice of SGD combined with momentum and Nesterov’s accelerated gradient in training artificial neural networks [21], we introduce momentum and Nesterov’s accelerated gradient to the weight update of ReSuMe learning rule in order to improve convergence and get around of local minima. Note that the addition of momentum is not a direct application of [21]. Momentum and Nesterov’s accelerated gradients are originally used in SGD to accelerate the learning process. However, the weight update in ReSuMe learning rule is not a gradient. Only when we use linear stochastic neuron models can we get the similar form between ReSuMe and SGD, and add the momentum term and Nesterov’s accelerated gradient into ReSuMe learning rule.

Stochastic Gradient Descent with Momentum (SGDM) [18] is a first-order optimization method that accelerates SGD using the combination of the current gradient update and the previous update, the momentum. Adding the momentum to the ReSuMe learning rule, the velocity vector \( v_{oi}^k \) can be updated by:

\[
v_{oi}^{k+1} = \beta v_{oi}^k + \eta \Delta w_{oi}^k,
\]

(4)
where \( k \) represents the \( k \)th iteration, \( \beta \in [0, 1] \) is the momentum coefficient and \( \eta \) is the learning rate. Nesterov’s Accelerated Gradient (NAG) [12] is also a first-order optimization method with better convergence rate under certain conditions. The velocity vector in NAG is computed by:

\[
v_{oi}^{k+1} = \beta v_{oi}^k + \eta \Delta w_{oi}^k = w_{oi}^k + \beta v_{oi}^k,
\]

where \( \Delta w_{oi}^k = w_{oi}^k + \beta v_{oi}^k \) is the gradient at \( w_{oi}^k \). By adding (4) or (5) to \( w_{oi}^k \), we introduce the momentum term or NAG to ReSuMe learning rule.

From Eqs. (4) and (5), we see that the methods differ in how the velocity vector is computed. Note that though the ReSuMe learning rule is biologically plausible, the augment of the momentum term and Nesterov’s accelerated gradient is only from the perspective of mathematics to find better solutions, thus it has little to do with biological plausibility. For convenience, hereafter we refer to the naive ReSuMe as ReSuMe, ReSuMe with Momentum as ReSuMe-M and ReSuMe with Nesterov’s accelerated gradient as ReSuMe-NAG.

3 Experimental Setup

3.1 Dataset and Task

The speaker-dependent auditory attention task is conducted on a subset of Grid corpus [4]. Grid corpus is a multi-talker audio-video sentence corpus aimed to support the research of speech perception. We randomly select audio clips of a female and a male speakers from the corpus, each with 20 utterances. The utterances per speaker is split into three parts, 10 utterances to generate the train set, 5 utterances to generate the validation set and 5 utterances to generate the test set. We mix two speakers’ utterances for each part respectively in order to get the mixture speech. Finally, there are 100 mixture speech in the train set, 25 mixture speech in the validation set and 25 mixture speech in the test set. The samples are clipped to 0.5 s for alignment. All the data are resampled to 8 kHz and normalized to reduce the computation cost. STFT with 32 ms window length, 16 ms shift time and sine window are applied to get the spectrogram \( \mathbf{X}_{t,f} \). Following [9], the background noise threshold is set to -40 dB of the source’s maximum magnitude. After clustering and setting silence units, the remaining T-F bins are mapped into spike trains according to different coding schemes. The spike trains are the input features for our spiking neural network model.

The goal of our speaker-dependent auditory attention task is to attend to the target speaker, the female speaker, from the mixture speech. We further compare the performance of our model with the two-layer baseline artificial neural network models, namely Multi-layer Perceptron (MLP), Recurrent Neural Network (RNN) and Long-Short Term Memory (LSTM) [10]. The number of input neurons and output neurons are determined by the frequency dimension \( F \) of the spectrogram \( \mathbf{X}_{t,f} \). The spectrogram \( \mathbf{X}_{t,f} \) is served as input features for the baseline models. The mask used by the baseline models is Ideal Ratio Mask (IRM), which is reported to gain better performance compared to IBM for the artificial neural networks [28]. The model architecture of our model and the baseline models are listed in Table 1.

### Table 1. Experimental configures. TF, TR and TP denote Time-to-First-Spike, Temporal-Rate coding and Temporal-Population coding, respectively. \( F \) represents the frequency dimension of the spectrogram \( \mathbf{X}_{t,f} \) and \( D \) represents the intensity level.

<table>
<thead>
<tr>
<th></th>
<th>Ours (TF,TR)</th>
<th>Ours (TP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of input neurons ( m )</td>
<td>( F )</td>
<td>( F )</td>
</tr>
<tr>
<td>Number of output neurons ( n )</td>
<td>( F )</td>
<td>( F(D - 1) )</td>
</tr>
</tbody>
</table>

3.2 Hyperparameter Settings

For simplicity, we set the resting potential to 0 and the threshold potential to 1 for LIF model. Other hyperparameters with respect to temporal coding schemes are found by validation, see Table 2 for details. The following
experiments are all based on these experimental settings. We optimize our model with initial learning rate 0.05 and decrease it by a factor of 0.95 if the distance between the desired output spike trains and the actual output spike trains on train set increases for over 5 epochs. The coefficient $\beta$ in ReSuMe-M and ReSuMe-NAG is set as 0.9. The model is evaluated on the validation set every 5 epochs and early stopping strategy due to GNSDR [24] with patience of 15. The baseline models are optimized using the same optimization methods with early stopping strategy. All the results are calculated by averaging over 5 trials with random initialization.

Table 2. Hyperparameters of our model with respect to different coding schemes.

<table>
<thead>
<tr>
<th>Coding schemes</th>
<th>$\tau$</th>
<th>$\tau_{ref}$</th>
<th>$\tau_{win}$</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-to-First-Spike</td>
<td>0.10</td>
<td>0.80</td>
<td>0.80</td>
<td>9</td>
</tr>
<tr>
<td>Temporal-Rate coding</td>
<td>0.60</td>
<td>0.80</td>
<td>0.60</td>
<td>8</td>
</tr>
<tr>
<td>Temporal-Population coding</td>
<td>0.45</td>
<td>0.80</td>
<td>0.70</td>
<td>10</td>
</tr>
</tbody>
</table>

4 Results and Analyses

4.1 Effects of Coding Schemes

To compare the performance of different coding schemes, we conduct experiments using ReSuMe and report GNSDR scores on our test set. Table 3 shows GNSDR results of different coding schemes and different optimization methods. We can see from the first row (ReSuMe) that among all the coding schemes, there are significant gaps between the performance of Time-to-First-Spike and those of Temporal-Rate coding and Temporal-Population coding. It suggests that Time-to-First-Spike is too simple to encode the complex sounds in the Cocktail Party Problem. Furthermore, Temporal-Population coding performs the best and attains a comparatively stable performance. We postulate that this is due to the fact that Temporal-Population coding uses a population of neurons to encode the intensity of the stimuli, which is more robust to noise and small weight changes compared to single-neuron coding like Temporal-Rate coding and Time-to-First-Spike. For single-neuron coding, Temporal-Rate coding represents the intensity information by the number and precise position of spikes in an encoding window, while Time-to-First-Spike represents the information only by the latency of a single spike to the onset, for which the strongest intensity may be transferred to the weakest intensity due to small disturbances and thus lead to bad performance.

Table 3. GNSDR scores (mean±stddev) of our model and baseline models on test set for speaker-dependent auditory attention.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Ours (TF)</th>
<th>Ours (TR)</th>
<th>Ours (TP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReSuMe-M</td>
<td>1.81±0.31</td>
<td>3.71±0.32</td>
<td>4.04±0.27</td>
</tr>
<tr>
<td>ReSuMe-NAG</td>
<td>2.16±0.21</td>
<td>4.03±0.29</td>
<td>4.41±0.29</td>
</tr>
<tr>
<td>Methods</td>
<td>MLP</td>
<td>RNN</td>
<td>LSTM</td>
</tr>
<tr>
<td>SGD</td>
<td>3.70±0.07</td>
<td>3.56±0.06</td>
<td>3.80±0.03</td>
</tr>
<tr>
<td>SGDM</td>
<td>3.72±0.07</td>
<td>3.58±0.05</td>
<td>3.94±0.07</td>
</tr>
<tr>
<td>NAG</td>
<td>3.74±0.06</td>
<td>3.58±0.05</td>
<td>3.94±0.06</td>
</tr>
</tbody>
</table>

4.2 Effects of Optimization Methods

In order to verify the effectiveness of momentum and Nesterov’s accelerated gradient, we conduct experiments optimized by ReSuMe-M and ReSuMe-NAG. As shown in Figure 3(a) and Table 3 on the top, ReSuMe-M and ReSuMe-NAG greatly speed up the learning speed for our model and improve the performance to some extent compared to ReSuMe. GNSDR scores suggest that ReSuMe-M and ReSuMe-NAG converge to better solutions than ReSuMe under all the coding schemes for our task, among which ReSuMe-NAG is better for Temporal-Rate
coding, ReSuMe-M is better for Temporal-Population coding and no significant difference for Time-to-First-Spike using ReSuMe-M or ReSuMe-NAG. While for Time-to-First-Spike, there is no significant improvement in learning speed, both ReSuMe-M and ReSuMe-NAG greatly speed up the learning process when the stimuli are encoded by Temporal-Rate coding or Temporal-Population coding. We observed similar results on the learning speed of the baseline models, see Figure 3(b). In comparison with the baseline models, the average of GNSDR scores under Temporal-Rate coding and Temporal-Population coding are mostly greater than the results of the two-layer baseline models. Though the variances are a bit larger comparatively, it is probably due to the characteristic of SNNs, which is sensitive to precise spike trains. Furthermore, MLP slightly performs worse than LSTM but better than RNN. Similar results can be found in the previous work [3].

4.3 Effects of Intensity Level

As shown in Table 3, our model under Temporal-Rate coding and ReSuMe-NAG gets the best performance among all the experimental settings. We further choose ReSuMe-NAG as the default optimization method and report GNSDR scores of different intensity levels encoded by Temporal-Rate coding or Temporal-Population coding to investigate the effects of intensity level during the encoding phase. We vary the intensity level $D$ around the optimal value for Temporal-Rate coding and Temporal-Population coding, respectively and fix other hyperparameters. As shown in Figure 4, neither too small nor too high values of $D$ are optimal, as they perform unstably. If the value of intensity level is too small, the number of discrete intensity levels are not sufficient to represent the complex auditory stimuli. If the value is too large, the computation cost will become relatively expensive. For the Temporal-Rate coding case, the total number of spikes will increase; for the Temporal-Population coding case, as the intensity level $D$ is one of the factors that influence the number of input neurons, bigger value will lead to more synaptic weights. When the intensity level takes a moderate value, our model achieves a stable performance with comparatively high mean GNSDR scores and small variances.

![Image](image_url)

Fig. 3. The average epochs needed for convergence of (a) our model and (b) baseline models with respect to different optimization methods.

5 Discussion

With the introduction of the momentum term and Nesterov’s accelerated gradient, our model encoded by Temporal-Rate coding or Temporal-Population coding outperforms the baseline models. It is notable that ReSuMe-M and ReSuMe-NAG improve the performance of our model quite a lot in comparison with our model optimized by naive ReSuMe. We infer that this is due to the digital implementation (computation in discrete time) of SNNs, which
leads to a rough error surface with many false local minima. These false local minima could cause poor performance of nonlinear learning methods [5]. Momentum and Nesterov’s accelerated gradient solve this problem by getting rid of the local minima to some extent and therefore improve the performance. Comparing Temporal-Rate coding and Temporal-Population coding, it is undeniable that the number of parameters of Temporal-Population coding are several times larger than that of Temporal-Rate coding and Temporal-Population coding is much time-consuming under the same configuration. We further choose ReSuMe-NAG as the default optimization method and investigate the effect of intensity level during the encoding phase under Temporal-Rate coding and Temporal-Population coding, and show that with the moderate intensity level, we are able to get a relatively high-quality stable performance.

Our proposed supervised spiking neural networks using temporal coding differs from the previous mentioned unsupervised spiking neural networks, two-layer oscillator network model [26, 15, 25] completely. The basic unit for the oscillator networks is a neural oscillator, while the basic unit for our model is a stochastic linear neuron model, for example, LIF model. The oscillator network model is CASA-based, which contains two main phases, segmentation and grouping. In an oscillator network model, the signal preprocessed by the auditory peripheral model is described by the activity of oscillators. Oscillators belonging to the same source will synchronize, while oscillators belonging to other sources will desynchronize. Whereas our work is a sequence-to-sequence model that learns a function about mapping the mixture to the attended speech. Moreover, as mentioned before, the oscillated neural networks only work in very simple cases, but our model are able to handle the multi-talker continuous speech cases.

Though there is still a great gap between the performance of our proposed model and those of conventional deep learning based models [9, 1, 29], our investigation demonstrates the potential of spiking neural networks. As the spiking neural network is a natural choice for learning the temporal dynamics of speech, it’s possible that incorporating other biologically plausible mechanisms will improve the performance of our model. Notice that auditory attention in this paper is stimulus-driven bottom-up attention, thus it’s straightforward to consider introducing task-driven top-down attention into the model to modulate the responses of the neurons or neuronal populations according to the given task. What’s more, since visual input enhances the auditory processing in the Cocktail Party Problem [7], it’s also possible to integrate the visual information of the talking faces into the recent task.

6 Conclusion and Future Work

In this work, we proposed a unified biologically plausible computational model using spiking neurons and two novel temporal coding schemes for auditory attention task. Then we introduced momentum and Nesterov’s accelerated gradient to ReSuMe learning rule to improve the performance and learning speed. We showed that our model outperformed the baseline artificial neural networks, demonstrating the potential of spiking neural networks in solving auditory attention task. Our work attempts to provide a new way to solve the Cocktail Party Problem from a more biological point of view. In future works, we hope to incorporate other brain-inspired mechanisms into our model to further improve the performance of auditory attention, for example, top-down attention and visual inputs.
Acknowledgements

This work was supported by the Major Project for New Generation of AI (Grant No.2018AAA0100400), the Beijing Brain Science Project (Grant No.Z181100001518006) and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No.XDB32070000).

Appendices

A Relationship between ReSuMe and Stochastic Gradient Descent

In order to make the relationship of ReSuMe and Stochastic Gradient Descent more intuitive, we will give the detailed derivation of ReSuMe learning rule for the two-layer spiking neural network from an alternative motivation following [20] in this section.

Consider a fully connected feed-forward spiking neural network with two layers of stochastic linear neurons, identified as I (Input) and O (Output). The instantaneous network error is defined as the least square error function:

$$E(t) = \frac{1}{2} \sum_{o \in O} [R^a_o(t) - R^d_o(t)]^2,$$  (6)

where $R^a_o(t)$ and $R^d_o(t)$ represent the actual instantaneous firing rate and the desired instantaneous firing rate at time $t$, respectively. Note that the instantaneous firing rate of a neuron can be regarded as the expectation of concrete spike trains over infinite trials:

$$R(t) = \lim_{M \to \infty} \frac{1}{M} \sum_{j=1}^{M} S_j(t),$$  (7)

where $M$ denotes the number of the trials and $S_j(t)$ is the concrete spike train in a trial. The spike train $S(t)$ has the following form:

$$S(t) = \sum_f \delta(t - t^f),$$  (8)

where $t^f$ denotes the $f$th spike in the spike train and $\delta(x)$ is the Dirac function with $\delta(t) = 0$ for $t \neq 0$ and $\int_{-\infty}^{\infty} \delta(t)dt = 1$. The instantaneous firing rate of output neuron $o \in O$, the probability density of firing at time $t$, is determined by the instantaneous firing rate of the input neurons $i \in I$:

$$R_o(t) = \frac{1}{n_i} \sum_{i \in I} w_{oi} R_i(t),$$  (9)

where $w_{oi}$ represents the weight between the output neuron $o$ and the input neuron $i$, $n_i$ represents the number of neurons of the input layer. In order to minimize the network error, we modify the weights according to gradient descent algorithm,

$$\Delta w_{oi}(t) = -\eta \frac{\partial E(R^a_o(t))}{\partial w_{oi}},$$  (10)

where $\eta$ is the learning rate. For the simplicity of the derivation, let $\eta = 1$ in the following equations. Applying the chain rule, we get:

$$\frac{\partial E(R^a_o(t))}{\partial w_{oi}} = \frac{\partial E(R^a_o(t))}{\partial R^a_o(t)} \frac{\partial R^a_o(t)}{\partial w_{oi}}.$$  (11)

According to the objective function, Eq. (6), the first term in the right hand of Eq. (11) is:

$$\frac{\partial E(R^a_o(t))}{\partial R^a_o(t)} = R^a_o(t) - R^d_o(t).$$  (12)
From expression in Eq. (9), the second term can be derived as:

$$\frac{\partial R_0^c(t)}{\partial w_{oi}} = \frac{1}{n_i} R_i(t).$$

(13)

Combining the above results, we have:

$$\Delta w_{oi} = -\frac{1}{n_i} [R_0^c(t) - R_0^c(t)] R_i(t).$$

(14)

Since we have only a single spike train \(S(t)\), we substitute the unknown instantaneous firing rate \(R(t)\) with \(S(t)\). Thus the weight change is modified according to:

$$\Delta w_{oi} = \frac{1}{n_i} [S_d^d(t) - S_0^d(t)] S_i(t).$$

(15)

The nonlinear product of \(S_i(t)S_d^d(t)\) can be interpreted as an STDP process triggered by the temporal correlation of presynaptic spike train \(S_i(t)\) and the desired postsynaptic spike train \(S_0^d(t)\). Likewise, the nonlinear product of \(-S_i(t)S_d^d(t)\) can be interpreted as an anti-STDP process over the presynaptic spike train \(S_i(t)\) and actual postsynaptic spike train \(S_0^a(t)\). Now let:

$$S_i(t)S_d^d(t) \rightarrow S_i(t)[a + \int_0^\infty W(s)S_d^d(t-s)ds]$$

(16)

$$S_i(t)S_d^d(t) \rightarrow S_i(t)[a + \int_0^\infty W(s)S_d^d(t-s)ds]$$

(17)

where the constant \(a\) is the non-Hebbian term, \(s\) is the time difference of the postsynaptic and presynaptic firings. The kernel \(W(s)\) defines the so-called learning window:

$$W(s) = \begin{cases} +A \cdot \exp\left(\frac{-s}{\tau_{\text{win}}} \right), & \text{if } s > 0 \\ 0, & \text{if } s \leq 0 \end{cases}$$

(18)

where \(A > 0\) is the amplitude and \(\tau_{\text{win}}\) is the time constant of the learning window. The learning window is set to zero when presynaptic spikes fire after postsynaptic spikes, because the anti-causal order of spikes contributes nothing to the success of learning [16, 17]. Then Eq. (15) takes the following form if we ignore the factor of \(1/n_i\):

$$\Delta w_{oi}(t) = [S_d^d(t) - S_0^a(t)][a + \int_0^\infty W(s)S_i(t-s)ds],$$

(19)

which is the original ReSuMe rule. The total weight change is given by integrating Eq. (19) over the time course:

$$\Delta w_{oi} = \int \Delta w_{oi}(t) dt.$$  

(20)

From the above derivation, it’s reasonable to introduce the momentum term and Nesterov’s gradient into ReSuMe learning rule.

**B Spike distance between two spike trains**

In our experiments, we decrease the learning rate if the distance between the desired spike trains and the actual spike trains increases for over 5 epochs. The distance is defined as the van Rossum distance [19], the Euclidean distance of two filtered spike trains:

$$f(t) = \sum_i \exp\left(\frac{-(t-t_i)}{\tau_c}\right)H(t-t_i),$$

(21)
\[ D^2(f, g) = \frac{1}{\tau_c} \int_0^T (f(t) - g(t))^2 dt, \quad (22) \]

where \( t_i \) are the spike times and \( H(t) \) is the Heaviside function; \( \tau_c \) is the time constant of the exponential function, which is fixed as 5 ms in our experiments.

References
Simulation of multi-peak discharge patterns by improved deterministic Chay model

Zhongting Jiang¹, Dong Wang¹,²*, Huijie Shang¹, and Yuehui Chen¹,²

¹School of Information Science and Engineering, University of Jinan, Jinan 250022, China
²Shandong Provincial Key Laboratory of Network Based Intelligent Computing, Jinan 250022, China
*Corresponding author. E-mail address: ise_wangd@ujn.edu.cn (Dong Wang)

Abstract. In this paper, the native characteristic of the potassium channel in the depolarization was considered and added as an improved term of the deterministic Chay model. The improved model can simulate multi-peak discharge patterns, including not only the familiar neural discharge patterns but also unusual ones, like the neighbor period alternating discharge pattern which was only simulated by the stochastic Chay model before. Thus, the chaotic and stochastic discharge rhythm were effectively integrated into the new improved model. Besides, combining with several nonlinear time series analysis methods, through qualitative analysis and quantitative calculation, the discharge characteristics of the two rhythms are explained, which provides an effective method for distinguishing and identifying the two rhythms.

Keywords: Neural discharge, Deterministic Chay model, ion channel, Chaos, Neural firing pattern, nonlinear time series analysis.

1. Introduction

As the basis of the nervous system, the neuron plays a crucial role in regulating the lives of higher animals and human beings[1-4]. The nervous system senses and transmits information through the discharge behavior of neurons. The basis of neural discharge is ion flow across membranes. In coordination with the complex and non-linear structure of neurons, the discharge activities also present abundant dynamic behaviors, such as periodic, chaotic, random discharges and so on[5-9]. Therefore, understanding the nature of neural discharge patterns from the perspective of dynamics is the basis and key link in the process of further understanding of nerve information processing.

Nervous system information is encoded in rhythm, not frequency. The rhythmic pattern of neural discharge has become a hotspot in neurodynamics research at all levels[10-11]. Because of the complex dynamical natures, the discharge rhythm of neural discharge exhibited various patterns and transitions between different patterns. [12-16]. As widely acknowledged today, charged ion currents, which were caused by the selective permeability transmembrane ion channels, are the material basis of action potential(AP), which is shown in Fig. 1[17]. In most cases, these channels, including the most important sodium channel and potassium channel, use special voltage-dependent mechanism to control ion transmembrane movement. Therefore, some behaviors of ion channels, such as noise, also have important effects on nerve discharge. In our previous studies, the interesting multi-peak discharge-periodic alternating discharge rhythm observed in real biological experiments has been successfully simulated by using the stochastic Chay model with the addition of the overall noise of the ion channel and the noise of the single potassium channel. Our research specifies the type of noise. However, those same studies have also shown that it seemed like too difficult to identify the sources and types of ion channel noise in a real neuron system, of course, same difficult to accurately describe or control the noise. So, in this paper, we changed focus to the ion channels activities at some crucial moments during the biological generation process of AP.

The nervous system senses and transmits information through the discharge behavior of neurons. Thereof, multi-peak discharge was considered as a very important mode because underlying abundant information. Many studies proved this point of view. For example, monkeys’ auditory nerves can use the multi-peak discharge to effectively strengthen the sensitivity to feel external periodic stimulation; something similar happened while observing the relationship between multi-peak discharge rhythm and temperature regulation in sharks’ thermoreceptors[18-19]. Among many characteristics of multi-peak discharge, determinacy and randomness are two important ones that should be paid more attention in the studies as its nature dynamics. Nevertheless, irregular and complex multi-peak discharge in experiments have been considered only abnormal rhythms in many previous research. By combining nonlinear theory with neuroscience, the chaotic and random neural discharges were recognized in a new way. Some chaotic rhythms mistaken as random rhythms before were corrected, while the real random neural firing activities, especially induced by noise, were gradually clearly explained from resonance mechanism[20-22].
An interesting irregular multi-peak discharge pattern observed in the biology experiment, which seemed as the rhythm with the composition of random alternation by periodic n and periodic (n+1) bursts, was only simulated by the stochastic model with noise before. Thus, to study the unification between chaotic and random discharge in the same bifurcation process, a new research viewpoint was proposed in this paper.

In this paper, a realistic nervous system model, the starting point of experimental nerve in rats, was chosen as an example to study the complex pattern of neural discharge and analyze its spontaneous discharge rhythm. The improved deterministic Chay model (DCM) simulated the chaotic discharges defined by the original deterministic model and the random discharge appearing in the previous stochastic model simulation and unified the two rhythms in the same bifurcation process. Whether for model simulation data or biological experimental data, we hoped to identify its basic properties through a series of reasonable index analysis, the basic dynamic properties, such as the identification of the discharge was deterministic or random rhythm. The discharge rhythm was analyzed by combining the non-linear time series analysis method with other traditional statistical methods [23-27]. In the relevant research process, the non-linear time series analysis method could characterize the randomness and certainty of discharge rhythm, and identified its signal sources. It became a bridge between biological experimental data, model simulation results, and intelligent algorithm analysis.

2. Models

2.1 Theoretical model

Chay model is based on the ion channel behavior derivative from Hodgkin-Huxley model, which describes the complex neural discharge rhythm [5,28]. It can simulate the multiple periodic and chaotic discharges of experimental neural starting point, the multi-peak discharge situation, and the transition mode between different rhythms, that is, bifurcation phenomenon. The basis of discharging is the charged ion flow formed by selectively allowing different ions to enter and exit the cell through various ion channels on the neuron cell membrane. Therefore, from the perspective of ion channel behavior, it is possible to grasp the mechanism of neural coding, processing and information transmission in essence. The unification of dynamic concepts and biological phenomena is also achieved through the simulation of Chay model.

In this paper, to research the principle of AP generation, we expand the behavior of potassium channels at the AP level, and make reasonable assumptions about the probability of potassium channel opening. When the voltage of a single pulse reaches the maximum, the potassium channel would immediately open. Here, the "immediately" indicated that the course occurred after the depolarization at once and finished at once before the repolarization. There was almost
no time delay, just shown as the corresponding time of the tip-top point connecting rising phase and falling phase in Fig.1. This hypothesis also corresponds to the phenomena observed in real biology. Consistent with the physiological process, it has exact biological significance. Considering this biological process, we control the realization of the process, the second formula of DCM has a value of 1 when the potassium channel opens fully at the moment of peak. At this time, \( w_K \) should be \( \tau_n / (n_\infty - n) \). This constraint only worked at the extreme point and did not work at any other points, where \( w_K = 1 \). Therefore, an improved deterministic Chay model(IDCM) based on peak constraint is formed is as follows:

\[
\begin{align*}
\frac{dV}{dt} &= g_I m^3 h(V_I - V) + g_K, n^4 (V_K - V) + g_{K,C} \frac{C}{1+C} (V_K - V) + g_L (V_L - V) \\
\frac{dn}{dt} &= w_K \frac{n_\infty - n}{\tau_n} \\
\frac{dC}{dt} &= \rho (m^3 h(V_C - V) - K_CC) 
\end{align*}
\]

(1)

(2)

(3)

\( V, n \) and \( C \) represent cell membrane potential, \( K^+ \) channel activation probability, intracellular \( Ca^{2+} \) concentration, respectively. \( \tau_n \) is relaxation time. \( V_C \) is the equilibrium potential of \( Ca^{2+} \) channel. It is noted that we did not introduce noise into the model, the improved model is still deterministic. Mannell numerical integration method is used to solve the model and the integration time step is \( 10^{-3} \) s. The setting of parameters can be seen in [29].

2.2 Data Analysis Methods

From the perspective of data analysis, many analytical methods have played active roles in the multi-peak discharge researches, especially in extracting the effective characteristics from the complex neural multi-peak discharges. This paper synthetically analyzed the randomness of multi-peak discharge produced in experiments and numerical simulation of mathematical models, summarized its occurrence mechanism, made a qualitative and quantitative analysis of the randomness of rhythm, which could calculate the randomness of discharge rhythm more effectively and understood the internal characteristics of chaotic discharge more comprehensively. This analysis point and method also provided theoretical basis for chaos and stochastic behavior research in real systems.

The whole research process adopts the research method of the combination of biological experiment, mathematical model numerical simulation and time series analysis was the main method throughout this paper. Taking the mature experimental neural starting point model as the object of biological experiment data acquisition, the rhythm generated near the bifurcation point of the same location is unified into a mathematical model for simulation. The interspike intervals(ISI) series converted from the spike trains were mainly used in the following analysis. A combination of multiple time series analysis methods, including first return map(FRM), surrogate data(SD), statistical histogram, nonlinear prediction(NPE), autocorrelation coefficient(ACC), approximate entropy (ApEn) and complexity calculation was used. All methods are the same as those described in [19].

3. Results

Both common and unusual multi-peak discharge, which was referred to in [30-31], can be simulated by the improved model. Here we focused on the discharge which can be simulated only by deterministic model or stochastic model in the previous studies. Because we found that both of them can be observed in the same one improved model. We can effectively integrate it into one model when \( \lambda n = 210 \) with \( V_C \) decreased from 300 mV to 0mV. It transitioning from period-1 bursting, to random alternating rhythms of period 1 and 2 bursts, to chaotic bursting via period-adding bifurcation, to a sharp "shrinkage", to spiking and ending with period-1 spiking, as shown in Fig. 2. The whole large background bifurcation shows the typical 'routes' that from period-adding bifurcation to chaos. However, the IDCM without any noise in this paper simulated the periodic alternation bursting and chaos discharge in the same bifurcation process, while chaos occurred at the lower periodic bifurcation points.

Fig. 2. Period-adding bifurcation in the DCM (\( \lambda n=210 \))

Here, much more attention had be paid to the discharge at the bifurcation point. In the IDCM, when \( V_C = 198mV \), the
irregular rhythms composed of random alternation composed of period 1 and period 2 bursts were successfully simulated. This discharge rhythms can be regarded as the random transition process by period 1 and period 2 bursts, as shown in Fig. 3(a). From the bifurcation behaviors and the spike trains of the results, this irregular aperiodic firing activities simulated by the IDCM was as same as the the random multi-mode firing simulated by the formerly reported stochastic Chay model.

When $V_c = 115$ mV, the chaotic discharge is simulated numerically. It is the transitional discharge rhythm between period 2 and period 3 bursting in period-adding bifurcation. As shown in Fig. 3(b), besides period 2 burst and period 3 burst, it also includes other periodic rhythms, as shown by the oblique arrow. When $V_c = 59$ mV, chaotic discharge is also simulated. It is the transition discharge rhythm between period 5 and period 6 bursting in period-adding bifurcation. As shown in Fig. 3(c), besides period 5 burst and period 6 burst, it also includes other periodic rhythms, as shown by the oblique arrow.

![Fig. 3. Spike trains of multi-peak discharges in period-adding bifurcation ($\lambda_{n}=210$).](image)

Fig. 3. Spike trains of multi-peak discharges in period-adding bifurcation ($\lambda_{n}=210$). (a) $V_c=198$, indicate random alternating discharges of period 1 and period 2 bursting. (b) $V_c=115$, indicate chaotic discharge between period 2 and period 3 bursting. (c) $V_c=59$, indicate chaotic discharge between period 5 and period 6 bursting. The oblique arrow represents other discharge rhythms except for the discharge rhythm in the transition period.

The differences between the chaotic discharge and the periodic random alternation rhythm were not only reflected in the spike trains as shown in Fig. 3, but also clearly relected in the FRM as shown in Fig. 4. The FRM of periodic random alternation rhythm is only a few scattered point clusters, while chaotic discharge has obvious clustered point clusters or specific special structures. For these two kinds of chaotic discharges, there are some differences, mainly in the regard at the degree of chaos, as shown in Fig. 4.

![Fig. 4. FRM analysis of ISI is done in the IDCM($\lambda_{n}=210$).](image)

Fig. 4. FRM analysis of ISI is done in the IDCM($\lambda_{n}=210$). (a) $V_c=198$, indicate random alternating discharges of period 1 and period 2 bursting. (b) $V_c=115$, indicate chaotic discharge between period 2 and period 3 bursting. (c) $V_c=59$, indicate chaotic discharge between period 5 and period 6 bursting.

The random alternating rhythms between period 1 and period 2 bursting when $\lambda_{n}= 210$, $V_c = 198$ mV by this new IDCM.
was taken for instance. It had 3 kinds of ISIs, sharing the smallest one by neighboring bursting, shown in Fig. 5 (a) and (b). ACC decreased from 1 to 0 oscillation with a very short delay, showing a small correlation. ACC was -0.6 when the lag is 1, and oscillates only near 0 when the lag was larger, which indicated that the rhythm was less autocorrelation, shown in Fig. 5 (c). NPE was predictable only when step size was 1, while the surrogate data of ISI was unpredictable regardless of step size, which illustrated in Fig. 5 (d). ApEn was 0.615663, complexity was 0.733148, it showed that the rhythm has slightly higher complexity and higher orderliness.

Next, we made a non-linear analysis of the chaotic discharge between period 2 and period 3 bursting when $\lambda n = 210$, $Vc = 115$mV in the IDCM. It had 4 types of ISIs bands, the smallest was the shared ISI band, shown in Fig. 6 (a) and (b). ACC was 0.8 when the lag is 1, and slowly decreased from 1 to 0 with the increase of lag, eventually only around 0, which indicated that the rhythm was strong autocorrelation, illustrated in Fig. 6 (c). The NPE of ISI was predictable when step size was small, while the surrogate data of ISI was unpredictable regardless of step size, which illustrated in Fig. 6 (d). ApEn was 0.323592, complexity was 0.372883, it showed that the rhythm has higher complexity and lower orderliness.

Fig. 5. The analysis results of ISI time series with random alternating rhythms of period 1 and period 2 in the IDCM($\lambda n=210$, $Vc=198$) (a) ISI series (b) ISI histogram (c) ACC (d) NPE (red — raw data, black — SD)

Fig. 6. The analysis results of ISI time series with chaotic discharge between period 2 and period 3 in the IDCM($\lambda n=210$, $Vc=115$). (a) ISI series (b) ISI histogram (c) ACC (d) NPE (red — raw data, black — SD)
Similar non-linear analysis of chaotic discharge between periods 5 and 6 is carried out as follows in Fig. 7. ApEn was 0.616251, complexity was 0.57694, it showed that the rhythm has slightly higher complexity and slightly higher orderliness.

![Fig. 7. The analysis results of ISI time series with chaotic discharge between period 5 and period 6 in the IDCM(λn=210, Vc=59). (a) ISI series (b) ISI histogram (c) ACC (d) NPE (red — raw data, black — SD)](image)

4. Discussion

In this paper, aiming at some problems in the study of multi-peak neural discharge, based on the physiological mechanism of AP generation, the improved deterministic Chay model was proposed and the simulated results were analyzed. By the new improved model, the random and chaotic discharges, which could be simulated only by deterministic model or stochastic model before, were numerically simulated in only one model. The random alternating discharge near the bifurcation point contains only two kinds of discharge rhythms near the bifurcation point, not other rhythms. The chaotic discharge near the bifurcation point contains not only two kinds of discharge rhythms near the bifurcation point, but also other rhythms. Their most prominent feature is that on the FRM of ISI, the rhythm of chaotic discharge has an obvious geometric shape, which is roughly "several" distribution. The degree of chaos varies, and the degree of bending of the geometric shape varies slightly. The complexity and autocorrelation of chaotic discharge rhythm is smaller than period alternating random discharge rhythm, and the certainty of chaotic discharge rhythm is stronger. Previous non-linear analysis of periodic alternating events showed that this pattern exhibit a typical integer-multiple like characteristics and stronger randomness.

The periodic bifurcation process with chaotic discharge can be simulated by the deterministic model, and the non-periodic can be simulated in the noise-added stochastic model, but the two rhythms are not unified into the same bifurcation process. In this paper, the improved model unifies the multi-peak discharge rhythm including the chaotic discharge defined by the original model and the random discharge simulated before in the same bifurcation process. Combining with various time series analysis methods, the analysis results reflected the randomness of the random discharge and the chaotic discharge. Through qualitative analysis and quantitative calculation, the discharge characteristics of the two rhythms are explained for identification. Besides, we found that the random alternating discharge is short-series alternation in the simulation. In the previous experiments, we also found the long-series alternating discharge rhythm. Whether it can be simulated in this model needs further study.

The improvement based on the generation mechanism of the AP is simple and intuitive. At the same time, this is the exact reason that the IDCM without noise here could well simulate the multi-peak discharge, which was recognized as a stochastic pattern and only simulated by the stochastic neuron model with noise before. Meanwhile, the result provides the possibility to understand the unity of the uncertainty and randomness in a neuron system in a new light and further enriched the understanding of the generation mechanism of neuron multi-peak discharge patterns.

5. Conclusion

Multi-peak discharge has been a very common phenomenon in the previous simulation, but it represents abundant nerve information transmission. Based on our previous work, we successfully simulated various spontaneous discharging rhythms and bifurcation processes of experimental nerve starting points using IDCM. The basis of discharging is the
charged ion flow formed by selectively allowing different ions to enter and leave the cell through various ion channels on the neuron cell membrane. Therefore, the phenomenon of neural discharge is analyzed from the viewpoint of ion channel behavior. Considering the effect of the opening of potassium channel on neural discharge after depolarization, the DCM is improved.

Then, using the IDCM, we simulate several kinds of multi-peak discharge, including the chaotic discharge in the DCM and the irregular discharge with random alternation between two neighboring bursts like that only simulated in the stochastic model before. The analysis results showed that the pattern exhibited both deterministic and random features. In the past, the period alternating discharge was only simulated in the stochastic Chay model but not in the DCM. The improved model in this paper just unified the different kinds of multi-peak discharge rhythms only simulated by the original or the stochastic models before into the same bifurcation process by one model. Besides, the action interval and the intensity of the noise should also affect the simulation results and was difficult to choose artificially in the previous studies, which was time-consuming, labor-consuming, and easily ignored. However, the IDCM here could simulate this multi-peak discharge without noise. This paper attempts to discuss the mechanism of multi-peak discharges from the unification of certainty and randomness. This paper also provides the possibility to understand the unity of the uncertainty and randomness in a neuron system in a new light.

Authors’ contributions

DW and YC conceived the methods. ZJ designed methods, realized algorithms and wrote the main manuscript text. HS conducted the experiments. All authors reviewed the manuscript.

Acknowledgments. This research was supported by the Shandong Provincial Natural Science Foundation, China (No. ZR2018LF005), the National Key Research and Development Program of China (No. 2016YFC0106000), the Natural Science Foundation of China (Grant No. 61302128, 61573166, 61572230), and the Youth Science and Technology Star Program of Jinan City (201406003).

References

Echo State Networks Composed of Units with Time-Varying Nonlinearity

Gouhei Tanaka*, Ryosho Nakane, Akira Hirose

Institute for Innovation in International Engineering Education, Graduate School of Engineering, The University of Tokyo, Tokyo 113-8656, Japan

gouhei@sat.t.u-tokyo.ac.jp
nakane@cryst.t.u-tokyo.ac.jp
ahirose@ee.t.u-tokyo.ac.jp

Abstract. The echo state network (ESN) is a representative model of reservoir computing. This model transforms sequential inputs into a high-dimensional feature space using a recurrent neural network called a reservoir and then performs a linear calculation to produce outputs from the high-dimensional signal in a readout. The major characteristic of ESNs is that they are trained with a linear regression algorithm and thereby capable of high-speed learning. A standard ESN uses a fixed dynamical reservoir, but this is problematic when addressing data that are generated by multiple dynamical systems switched with time. In such a case, it would be reasonable to consider a time-varying reservoir. Here we propose a method to adaptively change the nonlinearity of reservoir units depending on a statistical property of teacher data. We evaluate the efficacy of the proposed model in a prediction task with synthetic time series data generated by temporally switching between linear and nonlinear systems. The results show that the proposed method can reduce the prediction error, suggesting effectiveness of task-dependent reservoir designs.

Keywords: Reservoir computing, adaptive reservoirs, switched dynamical systems

1 Introduction

Reservoir computing is a computational framework capable of temporal/sequential pattern recognition [1–3]. One of the seminal models of reservoir computing is the echo state network (ESN) proposed by Jaeger in 2001 [4, 5]. A standard ESN model consists of a reservoir part of a fixed recurrent neural network and a readout part which linearly combines the reservoir states to produce an output. The reservoir transforms temporal inputs into a high-dimensional feature space such that the transformed data are linearly separable in the readout. Since the readout is trained with a linear regression algorithm, the training cost of ESNs is much less than that of other machine learning algorithms containing many model parameters to be adjusted iteratively. In exchange for the advantage in training cost, it is necessary to design a good reservoir in ESNs for achieving excellent computational performance. Although a guideline for designing good reservoirs in ESNs is found in Ref. [6], a good reservoir depends on the type of computational task and the property of time series data.

A reservoir should operate as a nonlinear transformation of inputs such that linearly inseparable problems are successfully solved in a readout. The nonlinearity of a reservoir is attributed to the nonlinear activation function of each reservoir unit. On the other hand, the dynamical behavior of a reservoir needs to embed a memory of past inputs for temporal pattern recognition. The length of short-term memory of a reservoir can be estimated by the memory capacity task [7]. The memory capacity is at most the number of units in the reservoir. It is known that the maximum memory capacity is achieved with a reservoir given by a delay line of linear units [8]. Hence, the nonlinearity of reservoir units is not favorable in terms of memory capacity. In this way, there is a fundamental tradeoff between nonlinearity and linear memory, both of which are required for temporal pattern recognition [9]. A simple way to make them compatible is to construct a mixture reservoir of linear and nonlinear units [10].

The performance of ESNs has been widely assessed in a prediction task with benchmark time series data. Benchmark time series data are normally generated from a single dynamical system such as a nonlinear autoregressive moving average (NARMA) model [11]. For predicting time series data generated by a NARMA-10 model, the predictive model should have a memory capacity larger than 10 steps and have at least quadratic nonlinearity. Such requirements for a reservoir depend on the property of time series data to be predicted. Therefore, it is questionable to build a predictive model under the assumption that the data is generated by a single dynamical system. There is often a case that dynamical systems behind real time series data seem to be switched with time as found in electrical signals produced by switched circuits [12], engineering data under switched controls [13], neuronal activity data resulting from a thresholding property of a neuron [14], biomarker data under intermittent medication [15, 16], epidemic outcomes under seasonal forcing [17], and those produced by other hybrid dynamical systems [18, 19]. The aim of this study is to propose a method to deal with such a case using a reservoir with time-varying characteristics.
In this study, we consider synthetic time series data which are generated by randomly switching an autoregressive moving average (ARMA) model and a NARMA model. The ARMA model is a linear process and can be well approximated with a linear model, whereas the NARMA model is a nonlinear process and can only be well approximated with a nonlinear model. Therefore, the synthetic time series is hard to approximate using a predictor represented as a single dynamical system. To deal with such a situation, we propose a method to switch the reservoir unit depending on a statistical property of the past time series data. We demonstrate that the proposed method can reduce the prediction error compared with a standard ESN model. Time-varying reservoirs could be realized with some physical reservoirs using evolving materials that change properties depending on input signals [11]. In Section 2, the proposed method is described. In Section 3, numerical results are shown to confirm the efficiency of the proposed method. In Section 4, this study is briefly summarized.

2 Methods
A standard ESN model is described as follows [4, 6]:
\[
\begin{align*}
\mathbf{x}(t + 1) &= f \left( W^{\text{in}} \mathbf{u}(t + 1) + W \mathbf{x}(t) \right), \\
\mathbf{y}(t + 1) &= W^{\text{out}} [\mathbf{u}(t + 1), \mathbf{x}(t + 1)], \quad t = 0, 1, 2, \ldots
\end{align*}
\]
where \( t \) is a discrete time, \( \mathbf{u}(t) \) is the input vector, \( \mathbf{x}(t) \) is the state vector of the reservoir, and \( \mathbf{y}(t) \) is the output vector. We denote the dimensions of the input vector, the state vector, and the output vector by \( N_u \), \( N_x \), and \( N_y \), respectively. The matrix \( W^{\text{in}} \in \mathbb{R}^{N_x \times N_u} \) is the input weight matrix, \( W \in \mathbb{R}^{N_x \times N_x} \) is the weight matrix of the recurrent neural network in the reservoir, and \( W^{\text{out}} \in \mathbb{R}^{N_y \times (N_u + N_x)} \) is the output weight matrix. The function \( f \) represents an element-wise activation function. This activation function is typically given by a sigmoid function, e.g. a hyperbolic tangent function \( f(x) = \tanh(x) \). The nonlinearity of the sigmoid function is responsible for the nonlinear transformation by the reservoir. In most cases, the nonlinearity of the activation function is fixed irrespective of input signals.

When a statistical property of time series data appears to be varied with time, the characteristics of the reservoir units can also be adaptively changed with time to enhance the approximation ability. We denote the parameter controlling the nonlinearity of an activation function \( f \) by \( a \) and assume that this parameter is a function of time as follows:
\[
\begin{align*}
\mathbf{x}(t + 1) &= f_{a(t)} \left( W^{\text{in}} \mathbf{u}(t + 1) + W \mathbf{x}(t) \right), \\
a(t) &= g \left( \mathbf{y}^{\text{target}}(t - 1), \ldots, \mathbf{y}^{\text{target}}(t - \tau) \right), \quad t = 0, 1, 2, \ldots,
\end{align*}
\]
where the time-varying nonlinearity parameter \( a(t) \) changes depending on the past target outputs \( \mathbf{y}^{\text{target}}(t - 1), \ldots, \mathbf{y}^{\text{target}}(t - \tau) \). The parameter \( \tau \) is the length of near past data for which a statistic (denoted by \( g(\cdot) \)) is evaluated.

In this study, we specifically assume that
\[
\begin{align*}
f_{a(t)}(x) &= \tanh(a(t)x) \\
g(x_1, \ldots, x_\tau) &= \frac{1}{1 + \exp(-\epsilon \text{Var}(x_1, \ldots, x_\tau))}
\end{align*}
\]
where \( \text{Var} \) represents the variance of \( x_1, \ldots, x_\tau \) normalized to \([0, 1]\). The idea behind this assumption is that the system should be nonlinear when the variation of the teacher data is large and it should be linear when the variation of the teacher data is small. The reservoir states computed from Eqs. (3)-(4) are used to provide the output in Eq. (2).

In the readout part, the optimal \( W^{\text{out}} \) is obtained using a linear regression as follows:
\[
W^{\text{out}} = \arg\min_{W^{\text{out}}} \frac{1}{T} \sum_{t=1}^{T} ||\mathbf{y}(t) - \mathbf{y}^{\text{target}}(t)||^2
\]
(7)
The network output generated by the trained model is represented as follows:
\[
\hat{\mathbf{y}}(t) = W^{\text{out}} [\mathbf{u}(t), \mathbf{x}(t)].
\]

3 Results
3.1 Time series data
The NARMA 10th-order system (NARMA-10) has been widely used for assessing the generalization ability of ESNs [20,8]. The system is described as follows:
\[
y(t + 1) = 0.3y(t) + 0.05y(t) \sum_{i=0}^{9} y(t - i) + 1.5u(t - 9)u(t) + 0.1,
\]
(9)
where $u(t)$ is the input sequence which is randomly sampled from the uniform distribution in the range $[0, 0.5]$. The righthand side contains a term depending on the past 10 steps and a nonlinear quadratic term of the input. Therefore, 10-step memory and nonlinearity are required for a machine learning model to successfully approximate this system. An example of time series generated by Eq. (9) is shown in Fig. 1.

In this study, our target is a machine learning prediction for time series data that are generated by switched dynamical systems. To change the order of the NARMA model with time, we consider a NARMA $m$th-order system (NARMA-$m$) following Eq. (9), which is described as follows:

$$y(t + 1) = 0.3y(t) + 0.05y(t) \sum_{i=0}^{m-1} y(t - i) + 1.5u(t - 9)u(t) + 0.1,$$

(10)

On the other hand, we consider an example model of ARMA with order 1 given by

$$y(t + 1) = 0.3y(t) + u(t) + 0.1,$$

(11)

which is a linear process and qualitatively different from the general NARMA-$m$ model.

We generate synthetic time series data by switching between the NARMA-$m$ model in Eq. (10) and the ARMA model in Eq. (11). The input $u(t)$ common to the two models is shown in Fig. 2(a). An example of the synthetic data is shown in Fig. 2(b). The total time length is divided into multiple time intervals with random lengths as shown in Fig. 2(c) where the intervals for the two models are alternately assigned. The order $m$ is a randomly chosen value from $[1,10]$ for the NARMA-$m$ model and $m = 1$ for the ARMA model as shown in Fig. 2(d). We can see from a comparison between Fig. 1(b) and Fig. 2(b) that the time series produced by the mixture model seems to contain alternate patterns due to the model switching unlike that produced by the NARMA-10 model.

### 3.2 Prediction performance

We performed time series prediction for the synthetic time series data shown in Fig. 2(b) using the proposed method. The input data was normalized to $[0,1]$ and the teacher output data was normalized to $[-0.5, 0.5]$. The parameter values of the reservoir in the ESN model were set as listed in Table 1. We generated synthetic data with time length 1500 and removed initial 100 steps to avoid the influence of transient states. Among the remaining 1400 steps, 1000 steps were used for training and subsequent 400 steps were used for testing. The
Fig. 2. An example of time series generated by the mixed model of NARMA-$m$ and ARMA models. (a) The input $u(t)$. (b) The target output $y_{target}(t)$. (c) The intervals for the ARMA model indicated by “0” and those for the NARMA-$m$ models indicated by “1”. (d) The order $m$ in the intervals.

testing error is evaluated with the following normalized root mean squared error (NRMSE):

$$NRMSE = \sqrt{\frac{\langle ||\hat{y}(t) - y_{target}(t)||^2 \rangle}{\langle y_{target}(t)^2 \rangle}},$$

where $\langle \cdot \rangle$ indicates the time average on the time period for testing.

Figure 3 illustrates the results of the prediction task for the synthetic time series data. Figure 3(a) shows the performance of the standard ESN model. The network output does not well capture the alternate patterns in the synthetic time series. This implies that the standard ESN model represented as a single dynamical system fixed with time is not suited for prediction of switched dynamical systems. Figure 3(b) shows the performance of the proposed method. The behavior of the network output seems to better fit the synthetic data compared with Fig. 3(a).

Figure 4 shows the NRMSE between the network output and the target output for different $\tau$ values of the proposed method. For each value of $\tau$, 50 trials were performed using different seeds for the random number which directly affects the initial reservoir condition, the network connectivity, and the input data $u(t)$. The blue circles indicate the average values of NRMSE and the error bars indicate the standard deviations. The dashed line indicates the NRMSE for the standard ESN model with a single reservoir fixed with time. The result shows that the proposed model yields better performance than the standard ESN model when $\tau$ is set at a small value below around 10. This suggests that a reservoir with time-varying nonlinearity is effective for prediction of a
Fig. 3. The target output (blue) and the network output (orange) after model training. (a) A standard ESN model. (b) The proposed method with $\tau = 3$.

Fig. 4. The prediction performance of the proposed model with adaptive nonlinearity, which is evaluated by the NRMSE in Eq. (12). The blue circle indicates the average over 50 different trials with different seeds for random number generation. The error bar indicates the standard deviation for the trials. The black dashed line indicates the prediction error for the normal ESN model with sigmoid units.

time series data where a short-term statistical property seems to vary with time. It is significant to find out a good statistic to well represent qualitative changes in given time series data.

4 Summary

We have proposed an ESN-based method to deal with a time series data having time-varying trends. In our method, the nonlinearity parameter of the reservoir unit is changed adaptively depending on the variance of the near past sequence in the teacher data. We have shown that the proposed method outperforms the standard ESN model with fixed nonlinearity in prediction of the synthetic data generated by switching between the ARMA and NARMA-$m$ models. The result suggests that adaptive reservoirs can be useful for some real data that are produced by switched dynamical systems. We have not compared the proposed model with other models trained with gradient methods, because the training time for them is much larger than the ESN model. A future
work is to apply the proposed method to other real time series data as found in phenomena related to switched dynamical systems. Adaptive reservoirs could be realized using physical substrates with plasticity in physical reservoir computing.

**Acknowledgments.** This work was based on results obtained from a project (No. 18102285-0) subsidized by the New Energy and Industrial Technology Development Organization (NEDO).

**References**


Multilayer LSTM with Global Access Gate for Predicting Students Performance Using Online Behaviors

Shaojie Qu [0800-0003-4237-1961], Kan Li* [0800-0003-3528-4739], Bo Wu [0800-0002-9440-1088], Kai Tang [0800-0002-0867-3707] and Weimin Liu [0800-0002-7520-0360]

Beijing Institute of Technology, Haidian District, Beijing 100081, China
* Corresponding author
qushaojie,likan,wubo,tangk,iwing@bit.edu.cn

Abstract. The Internet makes the everyday lives of users more convenient, but it also affects their work and learning capacity. This study evaluated the influences of the online behaviors of > 3500 college students on their academic performances. Although the university network log is tremendously large, only a few of its features affect data classification and prediction. In this study, we propose a Monte-Carlo-based feature selection algorithm to select the best feature set for representing student behaviors. Accordingly, we propose a student performance prediction algorithm based on long short-term memory that incorporates global features and considers the temporal behavior of students. Experiments show that our feature selection algorithm exhibits an improved capacity to select the optimal feature set and achieves 89.76% accuracy in student performance prediction. This study evaluated the influences of online behavior and specific websites on student performance.

Keywords: Data Mining, Feature selection, Long Short Term Memory, Monte Carlo, performance prediction.

1 Introduction

People use the Internet to learn, entertain, and search for answers. Although it offers significant convenience, its excessive use is likely responsible for the loss of interest of its users in learning and working, shifting their interests towards entertainment projects, which will likely shorten their attention spans [1–3]. This study attempts to a) examine the impact of the online behaviors of college students on their performances, and b) use the behaviors to predict their performance.

Given that only a few features are effective in the available network log data, the amount of low information features will affect data classification and prediction [4]. Presently, there are numerous feature selection or noise removal methods [5], including principal component analysis [6], random forest [7], and others. However, many algorithms ignore the relationships among these features, whereas other algorithms encounter non-deterministic polynomial (NP) problems when they select the optimal feature set from the massive datasets [8]. After the selection of the optimal feature set, the appropriate algorithm can achieve better prediction and classification results. In recent years, neural networks have been extensively used for prediction and classification. However, ordinary neural networks ignore temporal factors, whereas recurrent neural networks (RNN) [9] consider that temporal factors will ignore the far-reaching features. Accordingly, these networks are prone to the phenomenon of gradient disappearance and gradient explosion. The development of long short-term memory (LSTM) [10] neural networks solved the long-term memory loss and gradient problems of RNN but ignored the global features.

In this paper, the main contribution are as follows:

Firstly, we propose a Monte Carlo feature selection algorithm based on double sliding windows to select the optimal feature set that can represent student behaviors.

Secondly, we propose a LSTM algorithm with global Access gate to predict student performances and improve the attention to global features.

Thirdly, the experimental results show that the feature selection method could select the optimal feature set that better represent students’ behaviors and the improved LSTM algorithm gain the accuracy of 89.76% in predicting students’ performance.

2 Related Work

2.1 Feature Selection

A feature is an individual measurable property of a process that is being observed. The objective of feature selection is to select a subset of variables that most efficiently describe the input data, reduce the effects of noise or irrelevant variables, and still provide good prediction outcomes [11].

2.2 Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS) was used to solve planning and sequential decision-making problems and few researchers have studied Monte-Carlo tree in features selection.

Monte-Carlo trees have been combined with neural networks. Xiaoxiao Guo [19] use Monte Carlo tree search method to generate training data for real-time play in Atari Game. Silver, D [20] combines Monte Carlo simulation and policy networks, and their program achieved a 99.8% winning rate against other Go programs.

2.3 Neural Network

LSTM was designed to resolve the problem of prolonged time requirements for learning and information storage by using recurrent back propagation [21]. Gers et al. [10] incorporated the forget gate into LSTM, which enabled cells to reset themselves at appropriate times, and they found that LSTM with forget gates could easily solve the benchmark problems that other algorithms, including RNN and LSTM alone, failed to solve. LSTM is extensively used currently in activity recognition [22, 23], human trajectory prediction [24], and so on.

3 Double Sliding Window Monte Carlo Tree Search

At present, there are many feature selection algorithms including filter methods, wrapper methods, embedded methods, and some other methods [11–13]. The goal of feature selection algorithm is to represent the characteristics of samples with a small number of features [14–16]. We propose a double-sliding-window Monte Carlo tree search (DSWMCT) algorithm to select features and construct the optimal feature set. Our algorithm includes four phases, the expand policy, select policy, simulate policy and backup policy. Meanwhile, we added double sliding windows: the best window is used to identify good performance features, and the explorer window is intended to identify potential features to improve the performance. The algorithm is revealed as follows:

In this part, we introduce the feature selection method using the Monte Carlo tree. In the next part, we introduce the use of the LSTM neural network to predict students’ performance.
Fig. 1. The proposed Monte Carlo Tree-Based Multi-Task neural network for feature selection

Our method constitutes a Monte-Carlo Tree Search (MCTS)-based feature selection method to select the optimal features for predicting students’ performance. The method incorporates the Feature Upper Confidence Bounds Applied to Trees (FUCT) algorithm to select features and construct the optimal feature set. Our proposed FUCT algorithm, like the Upper Confidence Bounds for Trees (UCT) algorithm [17-19], also has four phases including EXPAND_POLICY, SELECT_POLICY, SIMULATE_POLICY and BACKUP_POLICY. However, there are some differences. Expanding nodes in the FUCT algorithm mean adding features to the optimal feature subset of the last iteration. The expanding nodes in the UCT algorithm are not fully leaf nodes, whereas for the FUCT algorithm, all expanding nodes are leaf nodes.

We initialized the Monte Carlo tree with the null feature set as the root $S_0$, the best set as null, the best award as 0, and the evaluation of feature $k$ as $award[k]$ using an MLP. We used the value of $award[]$ to evaluate expanded nodes and to select the top $m$ nodes. We used the value of $sim[]$ to evaluate the simulated result and selected the best feature in this iteration. Finally, we added the best node to the optimal feature subset and updated the award of each feature.

EXPAND_POLICY expands the leaf nodes for all features that are not in the optimal feature set. SELECT_POLICY finds the features to be simulated, as not all leaf nodes will be simulated. We selected the top $m$ and bottom $n$ features sorted by the award.

SIMULATE_POLICY updates the $Set_b$ (best set) to find more features and gain better performance. We simulated the result using each feature and the $Set_b$ obtained in the last iteration and then found the best simulation result $sim[b]$ for node $b$. If we could obtain better simulation results than the best reward in the last iteration, we would add the best feature $b$ to $Set_b$ and update the best reward.

We gain another array $Δaward[Vi]$ to evaluate the features. In Formula (1), $ρ$ is the best reward in the last iteration, and $θ$ is the best reward in this iteration. $SIM(V_i, Set_b) − ρ$ reflects the comparison between the performance of the new
feature set and the optimal feature set, and $(SIM(V_i, Set_b) - \theta)$ reflects the comparison between the feature and the best feature of this iteration. By adjusting the value of $\alpha$ and $\beta$, we can control the efficiency of searching for new features.

$$\Delta\text{award}[V_i] = \alpha(SIM(V_i, Set_b) - \rho) + \beta(SIM(V_i, Set_b) - \theta)$$  \hspace{1cm} (1)

BACKUP_POLICY updates the award of the other $m-1$ features using Formula (2) without the best feature in the iteration. With this strategy, bad features will gain negative $\Delta\text{award}$ and their ranking as awards will drop. Then, more features will be searchable in the next iteration.

$$\text{reward}[V_i] = \text{reward}[V_i] + \Delta\text{award}[V_i]$$  \hspace{1cm} (2)

4 LSTM with Global Access Gate

LSTM can solve the problem of gradient disappearance and explosion in RNN. The algorithm relies on information in the learning sequence. Basically, the LSTM unit is composed of three gates, namely a forget, an input, and an output gate [20-22]. The LSTM uses two gates to control the value of the unit state $c$. The first is the forget gate, which determines how much of the unit state from the last moment has been retained at the current moment. The second is the input gate, which determines how much of the network’s input is saved to the unit state at the current time. LSTM uses the output gate to control how much of the unit state is output to the current output value of the LSTM. A gate is essentially a fully connected layer whose input is a vector and whose output is a real vector between 0 and 1. However, LSTM only considers the information in sequence and ignores the influence of the global features on the results [23-24].

In this study, we propose a deep LSTM network with additional global attention. We add a global access gate in LSTM unit as shown in Fig 2, called GLSTM, to mark the global features that is used to calculate $g_t$ and $C_t$. Because an LSTM pays more attention to the timing of the sequences, and because its global evaluation concerning the sequences is poor, we added gate $g$ to store the global features and achieve better results.

![Fig. 2. Long short-term memory (LSTM) unit with global access gate.](image)

The LSTM neural network comprises three layers: an input, a hidden, and an output layer. When more than one hidden layer exists in a cyclic neural network, the output of each layer serves as the input to the next layer. As shown in Fig. 3, in this study, we construct a multi-layer LSTM network, where each hidden layer consists of 15 GLSTM units.
5 Results

In this section, we describe the data and experiment, and show the results of feature selection and prediction. In the first part, we reveal the data source. In the second part, different feature selection methods are compared. In the third part, we compare naïve Bayes (NB), logic regression (LR), multi-layer perceptron (MLP), LSTM, and GLSTM methods for predicting students’ performance.

Based on the average grade point of students, we assume that the last quarter of the students ranked are students with academic difficulties, and we find that 75 points is the demarcation line from the statistics. In this experiment, we use 75 points as a dividing line for predicting students’ performance.

5.1 Data

We have captured online data from > 3500 students in the period of one month. The number of logs was 737 million. These included 580,000 domain names, 34,000 websites that were visited > 100 times, and 10900 websites that were visited > 1000 times. We counted the number of times each student visited a website domain name and selected and analyzed each website domain name as a feature.

In the process of data processing, we can guarantee the student privacy from two aspects. First, before we analyzed the data, we will remove all personally identifiable information of students. Second, all the network logs only show the domain names of the websites visited by students. The details of the network logs are hidden. Although this will affect the results to a certain extent, it can guarantee the privacy of student information.

5.2 Feature Selection

For the selected features, we used the random forest method to evaluate the importance of the features (20000 most visited website) firstly. The results of the evaluation are shown in Fig. 4. Then we used the Monte-Carlo-based feature selection method for feature evaluation and selected the following features:
We used a random forest algorithm to evaluate the importance of all the features. The most distinguished features and their importance are listed in Table 1. Table 1 indicates that the most influential features are the takeout, games, and chat. We conducted further analyses and statistics on these characteristics. The experiment proved that the students who ordered takeaway often and did not go to the canteen to eat performed poorly, and the students who chatted and played more games performed worse on average.

![Fig. 4. Feature importance for all domain names using random forest method.](image)

Because students have visited more than 580,000 different domain names, it is very difficult to select the best feature set from a single domain name. We classify 26 different types of domain names, including games, chats, videos, learning, shopping, and takeaway. We counted the number of times each student visited each type of website and constructed a vector to describe the students' behavior. We use random forest, RFE, and DSWMCT to select the best feature set and then use a simple neural network to evaluate the selected feature set. Experiments indicate that the DSWMCT method is better than the random forest and RFE methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Feature Number</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>8</td>
<td>76.24%</td>
</tr>
<tr>
<td>RFE</td>
<td>8</td>
<td>76.32%</td>
</tr>
<tr>
<td>DSWMCT</td>
<td>10</td>
<td>76.96%</td>
</tr>
</tbody>
</table>

Random forest selected the best feature set to predict students' performance, which contains eight types of features. The domain name types represented by these features are games, videos, social, shopping, learning, news, applications, and cloud. The accuracy rate of the random forest in predicting students' performance is 76.24% based on the simple neural network. The best feature set selected by DSWMCT contains 10 types of features. These domain features are games, videos, learning, search, radio, payment, notes, live broadcasting, tourism, and blogging. The accuracy of predicting students' performance with the feature set is 76.96% based on the simple neural network. DSWMCT pays more attention to the correlation between features, and can select the best feature set, while the algorithms that only relies on feature ranking for feature selection will ignore the correlation between features.
5.3 GLSTM

As few people use online behaviors to predict students' performance, we just conduct experiments with the NB, MLP, LR, LSTM, and GLSTM methods using the feature set selected by DSWMCT. We use a three-layer GLSTM to predict student performance. We used 80% and 20% of the data for training and testing, respectively, and we iterated 300,000 times. The iteration process is shown in Fig. 5.

![Graph of Variations of loss and accuracy of GLSTM as a function of the number of iterations.](image)

**Fig. 5.** Variations of loss and accuracy of GLSTM as a function of the number of iterations.

The results are listed in Table 3. The experiments demonstrate that the LSTM algorithm performs better than the NB, MLP, LR, and traditional methods. It reveals that students' temporal behavior can better reflect students' performance. For the GLSTM algorithm, it achieves the accuracy of 89.76% and the recall rate of 96.5%. It reveals that GLSTM algorithm considers global features performs better than the LSTM algorithm.
6 Conclusions

This study evaluated the influence of the online behaviors of college students on their academic performance. We proposed a feature selection algorithm based on Monte Carlo tree search to select the best feature set that can represent student behaviors. We then proposed an LSTM-based student performance prediction algorithm that considered the temporal behavior of students and added global features. Experiments demonstrated that our feature selection algorithm could select the optimal feature set in a better manner than existing methods. Our prediction algorithm achieved 89.76% accuracy in the prediction of student performances. Based on the findings of this study, it is inferred that different websites accessed by the students had positive and negative effects on their performances.

References

A Hybrid Time Series Model based on Dilated Conv1D and LSTM with Applications to PM2.5 Forecasting

Liqing Zhao, Bo Cheng, and Junliang Chen

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China {mrdqing, chengbo}@bupt.edu.cn

Abstract. PM2.5 forecasting is crucial because it affects people’s physical health, but also provides guidance for pollution control. Since the prediction of PM2.5 involves various meteorological factors as well as the influence of historical data, in this paper, we take the prediction of PM2.5 as a multivariable time series prediction problem, and propose a hybrid time series model based on LSTM and dilated convolution networks-DConvLSTM. The model can capture the internal long-term dependence and external impact factors from a large amount of historical data, and make a prediction of PM2.5 concentration in the future period. DConvLSTM can identify the simple time pattern from the sequence data through the Dilated CNN in one dimensional, where the added cavity is used to filter the sequence signal and increase the receptive field. LSTM is a variant of RNN, which naturally has the characteristics of learning long-term dependence from time series data. The DConvLSTM is composed of a one-dimensional Dilated CNN, LSTM and a completely connected layer. In this paper, DConvLSTM model is applied to the prediction of PM2.5 concentration in Beijing. The experiment shows that the proposed model could well predict the PM2.5 concentration in the future period. Comparison with other models, DConvLSTM has a higher fit indicators and computational efficiency.

Keywords: PM2.5 · Time series · Conv1D · LSTM · Dilated Convolution.

1 Introduction

In recent years, the problem of air pollution brought about by the development of society and industry has become more and more serious. The haze formed by PM2.5 as the main component has received great attention because of its harmfulness to people’s body [1]. PM2.5 concentration have been included in new air quality standards by many countries. The prediction of PM2.5 concentration is crucial to provide guidance for people’s life and help for air pollution control. How to establish a reliable prediction model of PM2.5 and further improve the prediction accuracy is our main research work in this paper.

At present, many scientific research institutions are actively involved in the establishment of the PM2.5 prediction model. Benas used multiple linear regression models to predict the concentration of PM2.5 by combining more meteorological elements, but also did not consider time dependencies [2]. Model research based on time dependence and multivariate variables is a category of time series analysis in machine learning. The current best method for time series analysis is RNN(Recurrent Neural Network) in deep learning. Min Han Kim et al. proposed a RNN model combined with multivariate nonlinear regression algorithm for air quality prediction in 2010 [3]. In an article published in 2018, Lei used LSTM(Long Short-Term Memory), a RNN variant, to solve the problem of long-term dependence [4]. Compared with the Random Forest and Encoder-Decoder model, it achieved better performance and improved the accuracy of the prediction. Bai et al. have used a one-dimensional convolution network united with deep residual network to solve the sequence classification problem [5].

In this paper, our primary goal is to establish a PM2.5 prediction model based on one-dimensional dilated convolution network and LSTM to predict the concentration of PM2.5 better, which can include both multivariate
meteorological factors and time dependence. Another objective is to comprehensively evaluate and analyze the performance of our model. The final goal is to compare the performance with LSTM, GRU, and Conv1D to verify the superiority of DConvLSTM.

2 Related Work

In this chapter, we will introduce the principles of one-dimensional convolutional neural networks and LSTM, as well as the current state of research on the combination of convolutional neural networks and LSTM.

2.1 Conv1D(One Dimensional Convolution) and Dilated Convolution

Since the AlexNet designed by Krizhevsky et al. [6] won the championship with a huge margin in the image classification competition of ImageNet in 2012, CNN (Convolutional Neural Network) has become the focus of academic research. CNN can automatically extract partial features of data by convolutional calculation on original information [7]. Similarly, Conv1D can convolve one-dimensional signals to identify the underlying rules in sequence data [8]. Conv1D can not only extract more important features from the sequence data, but also reduce the dimension of the sequence. When reduced to length 1, the model can be used for regression modeling. Likewise, the length can be the number of categories, and the model can model the classification problem. Formula (1) is the convolution calculation formula.

\[ F(x) = (f * g)(x) = \sum_{\tau = -\infty}^{\infty} f(\tau)g(x-\tau) \] (1)

Where \( F(x) \) represents the function for the convolutional calculation, and \( f(x) \) and \( g(x) \) represents the input and convolutional kernel respectively. \( x \) is kernel size, and \( \tau \) represents the index of data in the sequence.

Dilated Convolutions are mainly used for semantic cutting [9]. Compared with the ordinary convolution, dilated convolutions adds a dilation rate parameter to indicate the size of the expansion. Dilated Convolutions differs from ordinary convolutions in that although the convolution kernel is the same size, it has gained a greater receptive field by the existence of the dilation rate [10]. The receptive field is the range in which the convolution kernel can act on the original information. Equation (2) is the calculation formula for the convolution kernel size of Dilated Convolutions.

\[ k_{size} = r_{dilation} \times (o_{size} - 1) + 1 \] (2)

Where \( k_{size} \) is the convolutional kernel size after expansion, \( r_{dilation} \) is the expansion rate, and \( o_{size} \) is the original size of the defined convolution kernel.

2.2 LSTM: Long Short-Term Memory

LSTM is a variant of RNN. The original purpose of LSTM is to learn the long-term dependencies between sequences and determine the optimal time lag for time series problems [11, 12]. The Schematic of LSTM is shown in Figure 1, which is composed of an input gate, a forget gate, an output gate, and a memory cell. Among them, three gates can solve the long-term dependence problem faced by ordinary RNN. Cells are a memory component that has a self-joining state that holds the temporal state [13]. The greatest strength of the memory unit is to preserve error constant of network and solve the problem that the gradient vanishes faced by the deep neural network as the number of layers increases.

The specific derivation formula of LSTM is as follows:

\[ f_t = \sigma(W_{zf} \ast x_t + W_{hf} \ast h_{t-1} + b_f) \] (3)
\[
\begin{align*}
i_t &= \sigma(W_{xi} * x_t + W_{hi} * h_{t-1} + b_i) \\
\hat{c}_t &= \tanh(W_{he} * h_{t-1} + W_{xc} * x_t + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \hat{c}_t \\
o_t &= \sigma(W_{xo} * x_t + W_{ho} * h_{t-1} + b_o) \\
h_t &= o_t \odot \tanh(c_t) \\
\sigma(x) &= \text{sigmoid}(x) = \frac{1}{1 + e^{-x}}
\end{align*}
\]

Where \( f_t, i_t, o_t \) are forget gate, input gate and output gate respectively. \( h_t \) represents the output of hidden layer in time \( t \). \( h_{t-1} \) and \( c_{t-1} \) represent the hidden output and state of cell memory in previous step \( t-1 \). \( W \) and \( b \) are weight matrix and bias matrix in different hidden layers. \( \sigma \) and \( \tanh \) are activation functions. The symbol \( \odot \) represents the scalar product of two vectors.

Since our model is composed of convolutional neural network and LSTM, we have done some research on the joint model of the two models. Shi et al. [14] proposed a ConvLSTM model which uses a convolutional network instead of a fully connected layer to model timely rainfall. But the model is a many-to-many model, and our prediction of PM2.5 is a many-to-one model. Gope et al. [15] proposed a hybrid model combining CNN and LSTM. The output of CNN in the model is used as input to LSTM, but CNN is mainly used to model spatial data. Zhan et al. [16] merge Conv1D and LSTM to model stock sequence data, but the data is univariate, and we are modeling a multivariate time series.

3 Model

To develop a deeper model in the time series and build an excellent model for PM2.5 prediction, we propose a hybrid model based on LSTM and Dilated Conv1D. The model can predict future information based on historical data more accurately and accelerate the efficiency of computing performance if the network is very deep. In part 3.1, we propose the model structure of DConvLSTM and elaborate on the design principle of the model. In Part 3.2, we describe the construction of the PM2.5 predictive model.
3.1 DConvLSTM: The model based on Dilated Convolution in One Dimension and Long Short-Term Memory.

Figure 2 shows the structure of our proposed model, DConvLSTM. The model combines Dilated Conv1D with LSTM. The model structure consists of input layer, extended convolutional layer, LSTM layer, fully connected layer, and output layer. The input layer can receive multivariate sequence data. The convolutional layer consists of one or more layers of dilated convolutional neural networks in one dimension. The Conv1D can identify the local features of sequences by convolution calculation of sequence data and the convolution kernel, and also shortens the length of the sequence and enhances the dependence between data. The dilated convolution network increases the receptive field of the convolution kernel of the same size, and brings faster mining of the sequence pattern which can balance the correlation between different data. Each layer of convolution has multiple filters, each of which learns a feature from the sequence, so the model can learn more features from the sequence. The LSTM layer consists of an LSTM network that learns the long and short term dependencies in the sequence. LSTM has two temporal relations: short-term memory and long-term memory. Short-term memory is realized by the interconnection of hidden layers. Long-term memory is realized by memory cells which run through the entire timing chain. The fully connected layer consists of the dense network and is responsible for mapping features into the sample space. The output layer outputs sequence data of the same latitude as the sample space.

Next, we will elaborate on the computational logic and construction principle of the model.

We assume that we have a set of time series data \( \{x_t^e|t = 1, 2...n \quad e = 1, 2...i \} \) is input, where \( t \) represents time lag and \( e \) represents feature number. Input features can be converted into the following matrix:

\[
X_{input} = \begin{bmatrix}
x_1^1 & \cdots & x_1^i \\
\vdots & \ddots & \vdots \\
x_n^1 & \cdots & x_n^i 
\end{bmatrix}
\]  \hspace{1cm} (10)

The data first enters the convolutional layer for local feature extraction. Figure 3 shows the diagram of one dimensional convolution computing.
We define the filter kernel matrix as follows:

\[ f_{\text{kernel}} = \begin{bmatrix} w_1^1 & \cdots & w_i^1 \\ \vdots & \ddots & \vdots \\ w_1^k & \cdots & w_i^k \end{bmatrix}, k \leq n - 1 \] (11)

Where \( i \) is feature number of input matrix \( X_{\text{input}} \), which is the channel number in the usual convolution kernel shape, and \( k \) is the size of convolutional kernel size, which is the filter width in the usual convolution kernel shape.

When we apply the convolution kernel to \( X_{\text{input}} \) for convolutional calculation, we first intercept the data matrix of length \( k \) and width \( i \) from the \( X_{\text{input}} \) matrix. We set it as \( X_p \):

\[ X_p = \begin{bmatrix} x_p^1 & \cdots & x_p^i \\ \vdots & \ddots & \vdots \\ x_{p+k-1}^1 & \cdots & x_{p+k-1}^i \end{bmatrix}, p+k-1 \leq n \] (12)

\( X_p \) is a matrix of the same size as the kernel intercepted from the \( p \)th row of the \( X \) matrix.

Based on the convolution formula (1), we perform convolution calculations for \( X_p \) and \( f_{\text{kernel}} \)

\[ F_p = w_1^1 \ast x_p^1 + w_1^2 \ast x_p^2 + \ldots + w_k \ast x_{p+k-1} = \sum_{j=1}^{k} \sum_{l=1}^{i} w_{j}^l x_{j+p-1}^l \] (13)

\( F_p \) represents the value of \( p \)th cell in the output series as Figure 3.

The above formulas give is the normal process of one-dimensional convolution calculation. When a hole is added to the convolution kernel, the dilated convolutional calculation of \( F_p \) also changes according to formula (2). The kernel with the original size \( k \) will change as follows:

\[ f_{\text{kernel}} = \begin{bmatrix} w_1^1 & \cdots & w_i^1 \\ \vdots & \ddots & \vdots \\ w_r^1 & \cdots & w_r^i \end{bmatrix}, r \ast (k - 1) + 1 \leq n \] (14)

\[ X_p = \begin{bmatrix} x_p^1 & \cdots & x_p^i \\ \vdots & \ddots & \vdots \\ x_{p+r*(k-1)}^1 & \cdots & x_{p+r*(k-1)}^i \end{bmatrix}, p+r \ast (k - 1) \leq n \] (15)

**Fig. 3.** Schematic of the One Dimensional Convolution Calculation.
The meanings of $f_{\text{kernel}}, X_p$ and $F_p$ are the same as in the above formula. $r$ is the dilated ration, which is same as $r_{\text{dilation}}$ in formula (2).

Then, we can calculate every value in the output matrix based on the formula (16). Outputs of each column comes from the same filter. So, M filters have M columns output. Just as shown in figure 3, we can get a matrix as the output of the convolution layer which has M columns and K rows, where M is the filters numbers and K is the output size of sequence after the convolutional calculation by input and kernel.

In the model, we continue to operate on the output of the convolutional layer as input to the LSTM layer. The calculation formula of LSTM layer is shown in equations (3) to (9). Here we do not specify the operation process. Let’s set $F_{\text{lstm}}$ to the output of the LSTM layer:

$$F_{\text{lstm}} = W_{\text{lstm}} * \text{LSTM}(F_{\text{conv}}) + b_{\text{lstm}}$$

(17)

Where LSTM(,) represents the calculation procedure based on formula (3) to (8). $W_{\text{lstm}}$ is the weight from hidden layer to output, and $b_{\text{lstm}}$ is the bias. $F_{\text{conv}}$ denotes the output of Convolutional layers.

The last layer that requires data operations is the full connectivity layer, which maps the output of LSTM layer to the sample space. Just like formula (18):

$$F_{\text{fc}} = \sigma(W_{\text{fc}} * F_{\text{lstm}} + b_{\text{fc}})$$

(18)

Where $\sigma(\cdot)$ is the activation function in full connection layer. $W_{\text{fc}}$ and $b_{\text{fc}}$ are weight and bias respectively. $F_{\text{fc}}$ is the output of full connection layer.

To sum up, this is the formula and principle of our prediction using DConvLSTM. Next, we will use examples to analyze the use of the model in more detail.

### 3.2 Prediction for PM2.5.

The concentration of PM2.5 will be affected by current meteorological factors such as rainfall, high winds, and temperature. At the same time, the PM2.5 concentration at the previous moment will remain after a period of time and will have an effect on the PM2.5 concentration at the current moment. Therefore, the prediction of PM2.5 is a multivariate time series regression problem.

$$p_{t+n} = f([p_t, m^1_t, m^2_t, \ldots, m^k_t], [p_{t+1}, m^1_{t+1}, \ldots, m^k_{t+1}], \ldots, [p_{t+n-1}, m^1_{t+n-1}, \ldots, m^k_{t+n-1}])$$

(19)

Where $p_t$ and $m_t$ denote the pm2.5 concentration meteorological factor respectively at time t. $k$ means the number of meteorological factors is k. $f$ denotes the model mapping from the meteorological factors and history PM2.5 to the PM2.5 concentration of n-step ahead. The construction of the function $f$ has been elaborated in Section 3.1, and is formed by nesting three functions $F_{\text{conv}}, F_{\text{lstm}}, F_{\text{fc}}$ and activation functions, just as shown in formula(20):

$$y_{\text{pre}} = f = \text{Sigmoid} \left( F_{\text{fc}} \left( \text{Tanh} \left( F_{\text{lstm}} \left( \text{Relu} \left( F_{\text{conv}}(X_{\text{input}}) \right) \right) \right) \right) \right)$$

(20)
A Hybrid Time Series Model based on Dilated Conv1D and LSTM with Applications to PM2.5 Forecasting

Where Sigmoid, Tanh, ReLu are activation functions. ReLU (Rectified Linear Unit) can overcome the problem of gradient vanishing and speed up the training. Tanh can take into account the differences in features in long-term changes after LSTM layer. Sigmoid can realize nonlinear mapping from features to sample space. $y_{pre}$ is the prediction result of the model.

Figure 4 is the structure diagram of the PM2.5 prediction model. On the left side of the figure, the input points to the format conversion of raw data, which is preparing for the input of convolution layer. The middle flow chart is the construction diagram of the PM2.5 prediction model. Among them we add super-parameters and activation function for every layer. We use two convolutional layers to handle larger time step values in case the neural network gets deeper. We also mark the output shape of every layer on the right of the figure.

There will be errors between predicted values by the model and real values. Since we solved the regression problem, we finally chose MAE (Mean Absolute Error) as the loss function, as shown in equation (21). MAE is more robust to outliers in the data.

$$loss = \frac{1}{m} \sum_{i=1}^{m} |y_{truth}^i - y_{pre}^i|$$  

(21)

Where $y_{truth}^i$ is the true value, and $y_{pre}^i$ is the predicted value by our model.

In order to minimize the loss, we choose Adam (Adaptive Moment Estimation) [17], which has a good effect and exceeds other adaptive algorithm, to optimize the weight in each layer. During the training process, the update rules for each parameter are as follows:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}} \hat{m}_t$$  

(22)

Where $\theta_{t+1}$ is the weight value after update. $\theta_t$ is the weight value before update. $\alpha$ is learning rate. $\hat{m}_t$ is exponential moving mean. $\hat{v}_t$ is square gradient.

In conclusion, by using DConvLSTM to predict PM2.5, we can learn the dependence of PM2.5 value with the past period of meteorological data and PM2.5 data in the training process of minimizing loss function. Then we can establish the model for PM2.5 forecasting.
4 Experiments

The dataset we used in the experiment is the PM2.5 dataset collected by the US Embassy in Beijing for each hour from January 1, 2010 to December 31, 2014. It contains the average PM2.5 concentration and average meteorological data in each hour.

We will have 2 experiments: (1) We will use different time steps and use different models to compare the performance with DConvLSTM. We will use LSTM, Conv1D and GRU to compare with our proposed model. (2) We will compare the computational performance of the four models proposed in (1). We use RMSE (Root Mean Square Error), MAE (Mean Absolute Error), $R^2$ (coefficient of determination) as the indicator of predictive performance. The formulas are as follows:

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (y_{pre}^i - y_{truth}^i)^2}$$

$$MAE = \frac{1}{m} \sum_{i=1}^{m} |y_{pre}^i - y_{truth}^i|$$

$$R^2 = 1 - \frac{\sum_{i=1}^{m} (y_{pre}^i - y_{truth}^i)^2}{\sum_{i=1}^{m} (\bar{y} - y_{truth}^i)^2}$$

(23)

Where $m$ is number of the test samples. $y_{pre}^i$ and $y_{truth}^i$ are predicted value and true value respectively. $\bar{y}$ is the average of the truth data.

We use the time cost in each epoch as the evaluating indicator of calculation performance.

4.1 Prediction Performance Comparison with other models

Based on the smooth nature of the PM2.5 data, we used 3h, 6h, 12h, 24h, and 36h as the time to predict the value of the next hour. Longer time steps will not only deepen the structure of the network, but also generate more computing consumption, so we will not use it.

Since there is a slight error in each training result, we use the mean of five results as the final indicator value. We used 3h, 6h, 12h, 24h and 36h as time step to test the predictive performance of models.

<table>
<thead>
<tr>
<th></th>
<th>LSTM</th>
<th>GRU</th>
<th>Conv1D</th>
<th>DConvLSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 hours</td>
<td>26.008</td>
<td>26.697</td>
<td>43.722</td>
<td>21.454</td>
</tr>
<tr>
<td>6 hours</td>
<td>26.514</td>
<td>28.883</td>
<td>38.956</td>
<td>21.996</td>
</tr>
<tr>
<td>12 hours</td>
<td>26.39</td>
<td>29.362</td>
<td>30.064</td>
<td>22.976</td>
</tr>
<tr>
<td>24 hours</td>
<td>26.417</td>
<td>26.86</td>
<td>29.092</td>
<td>20.151</td>
</tr>
</tbody>
</table>

We used 3h, 6h, 12h, 24h and 36h as time step to test the predictive performance of models. As can be seen from Table 1, four models can achieve PM2.5 prediction, but DConvLSTM is better on the RMSE using the test dataset than other models. As shown in figure 5, the prediction performance of DConvLSTM is the best at 24h. So the experiment proves that using the historical data and meteorological factors of one day to predict PM2.5 will
A Hybrid Time Series Model based on Dilated Conv1D and LSTM with Applications to PM2.5 Forecasting

be more accurate. But we also found that both LSTM and GRU also performed well. Among them, Conv1D has become better and better with the increase of time step. But the increasing steps also lead to more complex network structures. So, the experimental results prove that DConvLSTM is the best choice of PM prediction model.

We use 24h as the timestep to test the predictive performance of four models by using different evaluation metrics, such as RMSE, MAE, and R2-Score. Table 2 shows the values of different metrics. DConvLSTM performed best in the three baselines. From figure 6, we can see that among the predicted error of all models, the error metric of DConvLSTM is the minimum, but the coefficient of determination is the maximum.

Figure 7 shows a comparison between the actual value and the predicted value of PM2.5 by four models. The results show that DConvLSTM is the best matching of the real value trend among the four models.

### 4.2 Computing Performance Comparison with other models

We take 24 hours as the time step and analyze the time consumption of every model in training process. In the experiment, we use 50 epochs for training. As can be seen from the table 3, when the time step is 24, the Conv1D model consumes the shortest time and the LSTM consumes the longest time. But we know from section 4.1 that Conv1D has the largest error and DConvLSTM has the smallest error. Therefore, by comprehensive consideration, the DConvLSTM model is the best model for us to predict PM2.5.

![Fig. 5. The performance of predictions at different timesteps.](image.png)
Fig. 6. The performance of predictions at different metrics: (a) is based RMSE and MAE; (b) is the coefficient of determination.

Fig. 7. (a): The prediction of PM2.5 based on LSTM (timestep = 24). (b): The pm2.5 prediction based on GRU (timestep = 24). (c): The pm2.5 prediction based on Conv1D (timestep = 24). (d): The pm2.5 prediction based on DConvLSTM (timestep = 24).

Table 3. The statistics of training time.

<table>
<thead>
<tr>
<th></th>
<th>LSTM(s)</th>
<th>GRU(s)</th>
<th>Conv1D(s)</th>
<th>DConvLSTM(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total time</td>
<td>542</td>
<td>306</td>
<td>57</td>
<td>198</td>
</tr>
<tr>
<td>Average time</td>
<td>10.84</td>
<td>6.12</td>
<td>1.14</td>
<td>3.96</td>
</tr>
</tbody>
</table>
5 Conclusions

In order to solve the prediction problem of PM2.5, we propose a hybrid model based on LSTM and one-dimensional expansion convolution network. The model is mainly for time series prediction problems. ConvLSTM has three advantages: first, using the one-dimensional expansion convolution to filter the original sequence data, the time feature can be further extracted; second, the expanded convolution network can also reduce the dimension and speed up the calculation speed of the network; third, we can learn long-term and short-term dependencies in time series and can effectively build predictive models. Through our validation in the experiment, we found that the DConvLSTM we built can not only effectively model the sequence problem, but also greatly improve the computational performance of the model training.

Acknowledgment This work was supported in part by the National Key Research and Development Program of China (No. 2018YFB1003804).

References

Dissect Sliced-RNN in Multi-Attention View

Zeping Yu, Linqing Shi, and Gongshen Liu

Shanghai Jiao Tong University, Shanghai Minhang 200240, China
zepingyu@foxmail.com, {shilinqing,lgshen}@sjtu.edu.cn
https://www.sjtu.edu.cn/

Abstract. Sliced recurrent neural network (sliced-RNN) has achieved much faster speed and better performance than standard RNN. Training RNN used to be very slow, the slice structure successfully deals with this problem. The slice structure is a major breakthrough, however, the explanation of sliced-RNN is weak. It is difficult to understand why sliced-RNN can get performance improvement, because intuitively the slice structure destroys the relation of several adjacent words. To tackle this problem, we propose the multi-attention mechanism to explain why the slice structure could achieve good performance, and the visualization of the attention weights shows that the slice structure could obtain more information from the sequences. The slice structure is able to capture information on different levels. Additionally, we propose a shortcut connection over the multi-attention mechanism, which could help transfer important information effectively. We propose experiments on six text classification datasets, and the results show that the multi-attention mechanism and the shortcut connection are both useful.

Keywords: sliced-RNN · multi-attention · shortcut connection.

1 Introduction

Recurrent neural network (RNN) is useful on many NLP tasks. However, it is difficult to be parallelized because of the recurrent structure, so the speed of RNN is very slow. Sliced-RNN improves the structure by slicing the sequence into many subsequences and transmitting the information through multiple layers. Because of the improved slice structure, sliced-RNN has much faster speed and higher accuracy on the document classification task. The model structure of sliced-RNN is shown in Figure 1.

Although the experimental results showed that sliced-RNN has better performance than RNN, the reason why the slice structure is good has not been explained convincingly. The speed advantage is easy to understand, but the performance improvement is not. Intuitively, the slice structure may destroy the connection between some adjacent words and phrases. A possible reason is that the slice structure could obtain high-level information rather than only word-level information through the multi-layer structure. But this explanation has not been verified. To explore this problem, we implement the multi-attention mechanism on each layer of sliced-RNN. The attention mechanism is a widely used function to focus on the most important part of the sequences because not all parts of a sequence are equally relevant. The attention mechanism is widely used on a large amount of NLP tasks, such as machine translation, image caption, and text classification.

In this study, we propose the multi-attention mechanism, which could obtain information on different levels. We slice the sequence into many subsequences and use bidirectional RNN on the subsequences. And then we use the attention mechanism on each layer to capture the most important information of the subsequences. The representation of each subsequence is used as the input of the next layer, and this operation is repeated until we get the final representation of the sequence. In addition, we propose a shortcut connection over the multi-attention mechanism by paying attention to the input representations and the hidden states simultaneously on each layer. Moreover, we implement the multi-layer shortcut connection, which could be added across many layers. This method can help the information to be transmitted more efficiently.

The multi-attention mechanism could pay attention to the important information on different levels, and the information could be transmitted layer by layer. The model structure is similar to hierarchical attention network (HAN). Our model is more flexible than HAN, because the documents do not have to be split into sentences. Also, the attention mechanism could be added on more than 2 layers, so it could be much faster and capture high-level information from the sequence.

We compare our model with attention-based bidirectional RNN (att-BRNN). Figure 2 is a review of Yelp 2013, and the task is predicting sentiment labels from 1-5. The key words in this sentence are great and like. Att-BRNN pays too much attention to the word great and predicts the wrong label 5. Our model captures both words and predicts the correct label 4, and it also finds the "!" in the word 'vegus'. In this example, we could find the advantage of the multi-attention sliced-RNN. The slice structure could capture more comprehensive information from the sequence, so it could achieve better performance.

The main contribution of this paper is using the multi-attention mechanism to explain why the slice structure achieves better performance than the standard recurrent structure and propose a shortcut connection for multi-attention. We visualize the multi-attention weights and compare them with att-BRNN, and the results show that the slice structure could capture more information and transmit it through the layers. Our study clearly explains why the slice structure achieves good results.
2 Model Structure

In this section, we introduce the model structure of multi-attention sliced-RNN. As shown in Figure 1, our model has multiple layers. On the 0th layer, the input is the word embeddings of the sequence, and they are sliced into many subsequences. On each layer, a bidirectional RNN is added on the subsequences. Then the multi-attention mechanism pays attention to the hidden states of the bidirectional RNN, the important information is obtained in this way. Especially, if the shortcut connection is used, the multi-attention mechanism pays attention to both the bidirectional RNN and the input of this layer. Then the representation of each subsequence is used as the next layers input. This operation is repeated until the final representation $F$ is obtained. Similar to sliced-RNN, the number of the layers and the length of each subsequence are hyper-parameters which need to be chosen. The details of the model will be described in this section.

2.1 Bidirectional RNN

Gated recurrent unit (GRU) [5] is a simplified version of long short-term memory (LSTM) [7]. Both of the two recurrent units have a gating mechanism, which could solve the vanishing gradient problem. GRU has a reset gate $r$ and an update gate $z$, these gates could decide whether to keep the previous memory or accept the current input. The equations of GRU are as follows:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$  (1)
$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$  (2)
$$c_t = \tanh(W_c x_t + U_c (r_t \cdot h_t))$$  (3)
$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot c_t$$  (4)

The reset gate controls the candidate hidden state, and the hidden state is controlled by the update gate. The standard GRU only processes the previous information and ignores the future information. Bidirectional GRU could get both the future context and the previous context. The input sequence is processed by two GRUs from both directions. The two hidden states from both directions are concentrated.

2.2 Multi-Attention

The attention mechanism could focus on the important parts of the sequences. Because the slice structure has many layers, the attention mechanism has different weights on different layers. In this way, the multi-attention mechanism could get multi-level information.
Fig. 2. An example from Yelp 2013 dataset. The word "great" and "like" has the largest contribution. The label of this document is 4 (1-5). The prediction of multi-attention sliced-RNN (above) is the correct label 4, and the prediction of attention-based bidirectional RNN (below) is 5.

Fig. 3. Model structure of multi-attention sliced-RNN. The attention mechanism on multiple layers could capture more information on different levels. Here ATT means the attention mechanism, \( r \) is the representation of each subsequence.

Information from the sequence. The important information could be transmitted through the layers. The attention mechanism on each layer is:

\[
\begin{align*}
\mathbf{u} &= \tanh(W_h \mathbf{h} + b_h) \\
\alpha &= \text{softmax}(\mathbf{v}^T \mathbf{u}) \\
\mathbf{s} &= \alpha^T \mathbf{h}
\end{align*}
\]  \hspace{1cm} (5) \hspace{1cm} (6) \hspace{1cm} (7)

where \( \mathbf{h} \) is the hidden states produced by the bidirectional GRU. The representation \( \mathbf{s} \) is the average of \( \mathbf{h} \). The attention mechanism could learn the weights.

**shortcut connection** The shortcut connection pays attention to both the input representations and the hidden states. If shortcut connection is adopted, the input of the multi-attention mechanism is the concatenation of the input representations and the
Table 1. Dataset information.

<table>
<thead>
<tr>
<th></th>
<th>Yelp 13</th>
<th>Yelp 14</th>
<th>Yelp 15</th>
<th>Yelp P</th>
<th>Amz.F</th>
<th>Amz.P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documents</td>
<td>469k</td>
<td>670k</td>
<td>898k</td>
<td>598k</td>
<td>3.65M</td>
<td>4M</td>
</tr>
<tr>
<td>Average Lengths</td>
<td>129</td>
<td>116</td>
<td>108</td>
<td>153</td>
<td>93</td>
<td>91</td>
</tr>
<tr>
<td>Max Lengths</td>
<td>1060</td>
<td>1053</td>
<td>1092</td>
<td>1073</td>
<td>441</td>
<td>257</td>
</tr>
<tr>
<td>Classes Number</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

hidden states on each layer rather than the hidden states only. The important information could be transmitted easier through multiple layers. This idea is similar to the shortcut connection of Resnet [6], which is a commonly used method in CNN. With shortcut connection, this equation should be added before equation (5)-(7):

\[
h = \text{Concat}(x, h) \tag{8}
\]

where \(x\) is the input of this layer, and \(h\) is the hidden states of the bidirectional RNN.

Multi-layer Shortcut Connection The shortcut connection could help transmit important information directly across adjacent layers. When having many layers, an intuitive idea is to transmit information directly across many layers. Multi-layer shortcut connection could achieve this by paying attention to all the previous layers, which is similar to Densenet [8]. When using multi-layer shortcut connection, the mechanism should pay attention to:

\[
h = \text{Concat}(x, x_p, h, h_p) \tag{9}
\]

where \(x_p\) and \(h_p\) are the inputs and hidden states on previous layers.

2.3 Classification

In order to predict the labels, a softmax layer is added after the final representation \(F\):

\[
p = \text{softmax}(W_F F + b_F) \tag{10}
\]

and the loss function is negative log-likelihood with document \(d\):

\[
L = -\sum \log p_d \tag{11}
\]

3 Experiments

3.1 Datasets

Yelp reviews The Yelp reviews datasets are obtained from Yu and Liu [26]. They separate the dataset into Yelp 2013, Yelp 2014 and Yelp 2015. There are five sentiment labels from 1 to 5. Yelp P is a polarity dataset with sentiment labels 1 and 2, which are proposed by Zhang et al. [27].

Amazon reviews The Amazon reviews datasets are also obtained from Zhang et al. [27]. It also has a polarity dataset. The document length of Amazon datasets is less than that of Yelp datasets.

3.2 Baselines

We compare our model with sliced-RNN, att-BRNN, HAN and standard RNN. We use GRU as the recurrent unit.

GRU The whole document is the input, and we take the last hidden state as the representation for classification.
**HAN** HAN was proposed by Yang et al. [25]. The hierarchical structure has word-level and sentence-level, which could be seen as a special slice structure. Intuitively, HAN could achieve better results because it has semantic features of the documents.

**Attention-based bidirectional-RNN** Attention-based bidirectional RNN was proposed by Zhou et al. [28]. They use the attention mechanism after the bidirectional RNN layer. We compare our method with this model because we want to explain why the slice structure could achieve better accuracy than the standard recurrent structure.

### 3.3 Model Training

In order to compare with the baseline models, the training method is similar to Yu and Liu [26]. We pad the sequences into a proper length. We pad zero on sequences shorter than that length, and sequences longer than that length are truncated. To compare with HAN, we set the number of the layers to be 2. The word vocabulary is set to be 30000. We use the Glove embeddings [17] to initialize the embeddings.

We tune the hyper parameters on the validation set. On all the datasets, the embedding dimension is 200 and the GRU dimension is 50, so the bidirectional GRU has 100 dimensions on each layer. The final representation also has 100 dimensions. The mini-batch size is set to be 100, and we use the Adam optimizer [10] having $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \text{and} \epsilon = 10\epsilon(-8)$.

### 3.4 Experimental Results

The results are shown in Table 2. The first block is some deep CNN models with many layers. The second block contains some RNN-based models. The third block is our models with multi-attention and shortcut connection. Here MA means multi-attention, SC means shortcut connection, MLSC means multi-layer shortcut connection.

When we look at the second block and the third block, models with the attention mechanism achieve better performance than the non-attention models. Take Yelp 2013 dataset as an example. SRNN+MA has 1.46% higher accuracy than SRNN. AttBRNN has 1.8% higher accuracy than GRU. So we could see that the attention mechanism is very useful, it could pay attention to the most important parts of the sequences and extract the important information. When exploring the slice structure, we could find that the slice structure achieves higher accuracy than the standard recurrent structure. The accuracy improvement is not easy to understand. Intuitively, the connection between the words may be destroyed by the slice structure. However, the experimental results have shown that the slice structure achieves very good performance. An explanation is that the slice structure could get high-level information from the sequence, instead of only extracting the word-level information. In order to validate whether this explanation is credible, we will visualize the attention weights of multi-attention without shortcut connection and explore what the slice structure could learn. When we explore the shortcut connection and multi-layer shortcut connection, we could find that they are very useful and greatly improve the performance. On all the datasets sliced-RNN with one-layer or multi-layer shortcut connection outperforms the other models. The shortcut connection over the multi-attention mechanism is helpful for the information to be transmitted through multiple layers. The shortcut connection could choose to extract information from the hidden states or just the input representations on each layer, so the model is stronger and more flexible. On all datasets, HAN has a little higher accuracy than sliced-RNN. This is because of the semantic knowledge of the documents. The slice structure is more flexible because it does not need to be split into sentences. Moreover, the hierarchical structure of HAN sometimes does not work because the input sequence does not have any semantic features. For example, when we use RNN on user modeling and recommender system, the input sequences are the web pages that the users clicked before. The sequences cannot be split into sentences and words, while the slice structure is still useful in this case.

### 3.5 Visualization and Analysis

In order to verify whether our model could extract the important information from the input sequence, we visualize the attention weights and compare it with att-BRNN. We use multi-attention without short connection in order to explore the slice structure. We choose several examples from the Yelp 2013 dataset. Because the model has multiple levels on different layers, we multiply the attention weights together to compare with att-BRNN, which only has the word-level attention. The results are shown in Figure 4, Figure 5, and Figure 6. The larger the weight is, the darker the color is shown. The weight of each word is printed under the words.

The visualization of multi-attention is shown on the left, and that of att-BRNN is shown on the right. In order to compare with att-BRNN, we only show the word-level weights instead of showing the high-level weights. But when we look at this visualization, we could find that even if one word has large weight, the words near it have very small weights. It means that the slice structure has the ability to transmit important information through multiple layers. The weights of the multi-level
Table 2. Accuracy on document classification datasets, in percentage.

<table>
<thead>
<tr>
<th></th>
<th>Yelp 13</th>
<th>Yelp 14</th>
<th>Yelp 15</th>
<th>Yelp P</th>
<th>Amz.F</th>
<th>Amz.P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char CNN(9 layer)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>94.46</td>
<td>57.41</td>
<td>94.15</td>
</tr>
<tr>
<td>WOrd CNN(9 layer)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>95.40</td>
<td>55.60</td>
<td>94.16</td>
</tr>
<tr>
<td>Char CNN(9 layer)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>95.72</td>
<td>63.00</td>
<td>95.59</td>
</tr>
<tr>
<td>GRU</td>
<td>66.12</td>
<td>70.63</td>
<td>72.89</td>
<td>95.96</td>
<td>61.36</td>
<td>95.22</td>
</tr>
<tr>
<td>SRNN</td>
<td>66.80</td>
<td>70.76</td>
<td>73.30</td>
<td>95.99</td>
<td>61.41</td>
<td>95.18</td>
</tr>
<tr>
<td>HAN</td>
<td>68.34</td>
<td>71.46</td>
<td>74.03</td>
<td>96.43</td>
<td>63.49</td>
<td>95.98</td>
</tr>
<tr>
<td>Att-BRNN</td>
<td>67.92</td>
<td>71.14</td>
<td>73.62</td>
<td>96.17</td>
<td>62.73</td>
<td>95.46</td>
</tr>
<tr>
<td>SRNN+MA</td>
<td>68.26</td>
<td>71.32</td>
<td>73.84</td>
<td>96.26</td>
<td>63.24</td>
<td>95.53</td>
</tr>
<tr>
<td>SRNN+MA+SC</td>
<td>68.44</td>
<td>71.82</td>
<td>74.19</td>
<td>96.46</td>
<td>63.58</td>
<td>96.16</td>
</tr>
<tr>
<td>SRNN+MA+MLSC</td>
<td>68.56</td>
<td>71.64</td>
<td>74.47</td>
<td>96.48</td>
<td>63.44</td>
<td>96.28</td>
</tr>
</tbody>
</table>

attention could validate this. Take Figure 2 as an example. It is a 3-layer sliced-RNN model with multi-attention. The input length is 512, the sequence length of Figure 2 is 27, and the length of the subsequences on each layer is 8. The words great and like have the largest weight. The word great is the 6th word of the 5th slice on the 2nd layer, and like is the 2nd word of the 7th slice on the 2nd layer. We extract the attention weights on the 2nd layer: [0.00, 0.00, 0.00, 0.00, 0.30, 0.18, 0.35, 0.17], which means that the 5th and the 7th slice are the most 2 important slices. And the attention weights of the 5th slice are: [0.00, 0.00, 0.00, 0.00, 0.97, 0.02, 0.01], and the 7th slice: [0.02, 0.83, 0.02, 0.01, 0.01, 0.09, 0.01]. These two large weights correspond to the words great and like. This means our model could indeed find the most important information from the bottom layer and pass them layer by layer.

Figure 4 is an example that both multi-attention sliced-RNN and att-BRNN predict the correct label. We could find that our model finds more information than att-BRNN. Our model not only finds the words good and fair but also captures the words properly, little rich and still, which are also important. Our model could capture more information because it has more layers. The receptive field on each layer is small, so it is easier to capture the information and transmit it layer by layer. In contrast, the receptive field of att-BRNN is too large, it is hard to capture all the important information, so it just extracts some information and sets the weights very large. We could look at the weights on Figure 4, Figure 5, and Figure 6. In our model, the weights are smooth and distributed, the largest weight is less than 0.15. But in att-BRNN, the weights are large on very few words, such as good and not. The largest weight is more than 0.2. Once their model captures the wrong word, it will predict the wrong model, just like the mistake made in Figure 2.

Figure 5 shows an example with label 1. Our model predicts the correct label, and att-BRNN predicts label 2. We could find an amazing result from the visualization: Even if the word love has the largest weight, our model still predicts the correct label. This means our model definitely understands the word love is not to praise, but to criticize. Although att-BRNN captured the word not and set the weight very large, it regards the word love as a positive word, so att-BRNN thinks the review is an implied criticism and predicts the wrong label 2.

Figure 6 is an example with label 1. Att-BRNN predicts the correct label, and our model predicts label 2. However, when we look at the review, we may have the same choice as multi-attention sliced-RNN. Our model captures the words on a good note and great, which talk something positive, that is, the onion rings. Our model recognizes them, while Att-BRNN does not. Amazingly, although the words on a good note and rings where great are actually in two subsequences, the slice structure captures this information. This means the slice structure does not break the word connection and could transfer the information through layers.

In addition to the examples already shown, there are many other examples that demonstrate the validity of the slice structure. We will open source our data and code later.

4 Related Work

Deep learning methods have great results on text classification. Kim uses convolutional neural network (CNN) for text classification [9]. Zhang et al. build a character-level CNN for text classification [27]. RNN is the most widely used model for sequences because it could capture the order information, so it is used on many NLP tasks, including machine translation [5], image caption [24], and question answering [4,23]. Several works combine CNN and RNN for text classification [12,27].
The attention mechanism is used in many NLP tasks including machine translation [5], image caption [24], and text classification [28]. Another attention mechanism called self-attention could capture the relationship of different positions in a sequence [22], and it has been used successfully in a variety of tasks including semantic role labeling [19], abstractive summarization [16], textual entailment [15] and learning task-independent sentence representations [14].

Several works try to improve the RNN speed by improving the recurrent units [1,2,11,13]. Some architectures are proposed to improve the recurrent structure of RNN [3,11,18]. The slice structure proposed by Yu and Liu 2018 [26] could greatly improve the speed of RNN. The structure is similar to wavenet [21], and the difference is that sliced-RNN uses recurrent units. The slice structure is also related to the hierarchical structure [20,25]. The slice structure is more flexible because it does not need to split the documents into many sentences, and it is also much faster because it could have more than 2 layers and shorter subsequence length.

5 Conclusions

In this paper, we present the multi-attention mechanism for sliced-RNN, which explains why sliced-RNN has better performance than RNN. Also, we propose the shortcut connection and multi-layer shortcut connection, which could transmit the information effectively and improve the performance. By visualizing the attention weights and comparing with attention-based bi-RNN, we could find that the slice structure could capture more information, and the information could be transmitted
layer by layer. Experimental results on six document classification datasets have shown that sliced-RNN with multi-attention mechanism achieves the best performance.

6 Acknowledgements

This research work has been funded by the National Natural Science Foundation of China (Grant No.61772337, U1736207), and the National Key Research and Development Program of China NO. 2016QY03D0604 and 2018YFC0830703.

References

Discovering Sequences in Systems Logs by Neural Networks

Jakub Nowak¹[0000 – 0002 – 1572 – 3426], Marcin Korytkowski¹[0000 – 0002 – 6002 – 2733], Marcin
Wozniak²[0000 – 0002 – 9073 – 5347], Rafał Scherer¹[0000 – 0001 – 9592 – 262X]

¹ Częstochowa University of Technology
Al. Armii Krajowej 36, 42-200 Częstochowa, Poland
{jakub.nowak, marcin.korytkowski, rafal.scherer}@iisi.pcz.pl
http://iisi.pcz.pl

² Institute of Mathematics, Faculty of Applied Mathematics, Silesian University of Technology, 44100 Gliwice, Poland
marcin.wozniak@polsl.pl

Abstract. Computer and network system logs consist of an enormous number of various events. By discovering
sequences of events related to a system failure, we can identify its causes. It is usually a difficult task, as many
various factors can contribute to the failure. In this paper, we use a deep neural network to discover unknown
sequences in system logs as an alternative to sequential pattern mining algorithms. We train the neural network
to recognize repetitive sequences and test it on unseen real sequences of events from system logs. The logs are
one-hot, event-level encoded. The network achieves very high accuracy and is faster than traditional algorithms
for sequence discovery.

Keywords: computer system logs, security, sequential pattern mining

1 Introduction

Detecting recurring events is a very important issue. Based on the collected previous knowledge, it is possible to
determine if after the occurrence of a sequence of events other previously observed events may be repeated. This
knowledge can be used to analyse sales data to define associations between products better, to observe natural phe-
nomena, including determining the probabilities of cataclysms or analysing stock market trends, bank transactions,
software events, web page requesting sequences, customer behaviour. In the paper we discover unknown sequences
in data preceding failure by a neural network. Sequential data can be used also to detect intrusions [9].

One of the first works formalising issues related to sequences was frequent itemset mining (FIM) by Agrawal
and Srikant in [1]. It is a slightly different task as it concerns basket-item data and associations. They introduced
three algorithms from the Apriori family to solve the problem. In their research, the most important emphasis was
put on the performance of algorithms using their sequence generator (now unavailable). They compared the impact
of the minimal sequence support in the database on the calculation time. The Apriori methods require that the
searched sequence is closely related in each example (there is no ABC sequence in the ABDC sequence).

The paper was followed by [11], where the Generalized Sequential Patterns algorithm (GSP) was presented.
GSP was improved in three ways. There were possible gaps between consecutive elements of the sequence. It was
possible to define time constraints. The concept of taxonomy was introduced. Another development was the SPADE
algorithm [4]. SPADE searched the database in a vertical way, which means that it looked for associations between
individual elements and then created a longer sequence of them, which translated into better performance. A more
detailed comparison of SPADE and GSP can be found in [13]. The next was the BIDE algorithm [15][14], which
was designed for closed sequences for improving efficiency.

In the paper, we train a neural network to recognize unknown sequences in system logs. We are interested in
sequences preceding a failure, i.e. those possibly contributing to the failure. We train the network with the synthetic
input data containing sequences mixed with random numbers. The desired output is the clean sequence without
the added noise. We test the network (sequence discovery stage) on system logs unseen during training. We convert
the real-system logs to one-hot event-level encoding. It needs to be emphasized that the approach presented in the
paper is not classification or pattern matching what a typical application of neural networks is. It is also not a
regular expression matching (such as the task of the GREP utility).

The advantages of the proposed approach are speed in the case of long sequences and a possibility to discover
transposed sequences. We use the U-Net architecture, described in Section 2, with one-hot event-level encoding. We
generated sequential data for the training stage of the experiments. To train the network, we generated a data set
where the input data are sequences mixed with random numbers and the output data are sequences to be found (Section 3).
Through this research, we highlight the following features and contributions of the proposed model.
We present a novel, neural network-based method for discovering unknown sequences. Our work provides new insights, showing that neural networks can detect unseen before sequences with nearly 100% accuracy. In the case of long sequences, our approach works faster as the neural network analyses a long window at once.

The remainder of the paper is organised as follows. In Section 2, we described the neural network architecture. Section 3 described data preparation. Section 4 describes experiments comparing the method with some standard sequential mining algorithms. Finally, conclusions and discussions of the paper are presented in Section 5.

2 U-Net Convolutional Neural Network

The U-Net network [10] is a special architecture of convolutional networks pioneered in [8]. It was designed for semantic segmentation of medical images to detect tumours or changes in X-ray images for relatively small sets of training data. This was the main rationale behind this choice. It is not possible to generate all possible sequences for training due to a combinatorial explosion. Moreover, for the nature of the problem, there is no point in performing data augmentation. Earlier, we performed experiments with various standard convolutional networks without success.

The U-Net network has an encoder-decoder structure; the encoder that processes the image in the first part of the network is designed to extract the most important elements. A max-pooling operation with size $2 \times 2$ is used to reduce the feature map by selecting the elements with the highest value. The reverse process is up-pooling (up-conv) [16]. This operation can be performed in many ways. By creating an additional filter, where the layer is calculated or by diluting the image, describing each value from the input map feature with zeros. In our case, each value is duplicated (reproduced) from the size $1 \times 1$ to the size $2 \times 2$. The maps from the encoder are additionally copied to the decoding part, creating a single 3D matrix where later convolution is carried out simultaneously on them and the layers after up-pooling.

Fig. 1. Architecture of the U-Net network used in the paper.

We have to specify the size of the input and output two-dimensional matrices when designing the U-Net architecture. In our architecture, the input and output have the same size $n \times m$, where $n$ is the number of encoded
characters (the dictionary size), $m$ is the maximal number of elements in the analysed data (the window that searches for sequences). As with black-and-white images, we only use one input channel. Our goal is to obtain only numbers that form a repetitive sequence at the U-Net output. We applied zero padding operation in the network so that the size of the network was not modified after the convolution [6]. In the experiments we use the following U-Net structure:

1. Conv. padding(3x3), in(1x256x256), out(32x256x256), ReLu activ.
2. Conv. padding(3x3), in(1x256x256), out(32x256x256), ReLu activ.
3. MaxPooling (2x2)
4. Conv. padding(3x3), in(32x128x128), out(64x128x128), ReLu activ.
5. Conv. padding(3x3), in(64x128x128), out(64x128x128), ReLu activ.
6. MaxPooling (2x2)
7. Conv. padding(3x3), in(64x64x64), out(128x64x64), ReLu activ.
8. Conv. padding(3x3), in(128x64x64), out(128x64x64), ReLu activ.
9. MaxPooling (2x2)
10. Conv. padding(3x3), in(128x32x32), out(256x32x32), ReLu activ.
11. Conv. padding(3x3), in(256x32x32), out(256x32x32), ReLu activ.
12. MaxPooling (2x2)
13. Conv. padding(3x3), in(256x16x16), out(512x16x16), ReLu activ.
14. Conv. padding(3x3), in(512x16x16), out(512x16x16), ReLu activ.
15. UpSampling (2x2)
17. Conv. padding(3x3), in(256x32x32), out(256x32x32), ReLu activ.
18. UpSampling (2x2)
19. Conv. padding(3x3), in(256x64x64[18]+128x64x64[8]), out(128x64x64), ReLu activ.
20. Conv. padding(3x3), in(128x64x64), out(128x64x64), ReLu activ.
21. UpSampling (2x2)
22. Conv. padding(3x3), in(128x128x128[20] + 64x128x128[5]), out(64x128x128), ReLu activ.
23. Conv. padding(3x3), in(64x128x128), out(64x128x128), ReLu activ.
24. UpSampling (2x2)
25. Conv. padding(3x3), in(128x256x256[20] + 32x256x256[2]), out(64x128x256), ReLu activ.
26. Conv. padding(3x3), in(64x256x256), out(64x256x256), ReLu activ.
27. Conv. padding(1x1), in(64x256x256), out(1x256x256), ReLu activ.

3 Frequent Sequences Data

The data set for supervised neural networks must consist of training and testing data. In the case of standard sequential data mining, the datasets focus primarily on checking the efficiency of the algorithm, and it is difficult to find a set that could be used for neural network purposes. Our aim was that the neural network recognises sequences that have not participated in the training process. In other words, it must grasp a general idea of repetitive sequences instead of memorising sequences existing in the data set. Furthermore, every type of sequential data can be transformed into numbers from a dictionary. Thus, we decided to train the network with synthetic random sequences.

We developed a generator for sequence datasets. Each item was a number from 1 to $n$, where $n$ is the maximum value we can encode in a given U-Net network. In other words, $n$ is the aforementioned dictionary size. One training example consists of smaller transactions in which a repetitive sequence of characters (green colour in Fig. 2) is hidden between them, and other numbers were drawn randomly from numbers not used to create the sequence. After each transaction, there is a break created from an additional number of random characters that also did not participate in the creation of the sequence. An example of the sequence is presented in Figure 2. The parameters for creating samples are:

![Sequence Example](image)

Fig. 2. Data sequence example generated by the generator.
The size of the convolutional network input \( n \) (dictionary size),
- The maximal space between sequence elements (blue parts in Fig. 2),
- The maximal and minimal length of the sequence itself (numbers in green in Fig. 2),
- The maximal interval between sequences (yellow fields).

Using this type of generator has several key advantages in the case of neural networks. During training, we want to know what sequence the network should return and to compute the training error. In the paper, we use one-hot number-level signal encoding. The input signal to the convolutional network (and the output) is a two-dimensional matrix. In algorithms such as GSP or SPADE, we have no limit to the number of elements that a sequence can consist of. In the case of the proposed method, we have to assume it earlier.

The proposed method discovers sequential patterns. Standard algorithms, such as BIDE, have a user-specified threshold named minimum support (a value in \([0,1]\)). In the proposed method, we cannot set this parameter, and the only possibility is to prepare suitable training data.

4 Experiments

In this section, we present experiments showing the performance of the method and the ability to recognize transposed sequences. We had to prepare datasets with input data and the desired output with the sequence separated from the surrounding noise. We present an example of such a training pair in Fig. 3. The neural network CNTK script and the data generator is available at [https://github.com/DiscoveringSequentialPatterns/](https://github.com/DiscoveringSequentialPatterns/).

4.1 Training

We trained U-Net with the backpropagation algorithm with the Adam optimizer [7], which adaptively sets the learning rate. That helps to leave training error local minima. Generally, the Adam optimizer is a combination of several earlier deep learning optimizers and allows achieving excellent results in a broad area of tasks. We set the momentum term to 0.9 [12], and the learning rate to \(1 \times 10^{-4}\) during the first 2-4 epochs. Then, it remained constant at \(1 \times 10^{-5}\). Establishing the loss function was an extremely important element in the case of the U-Net network. We applied the Dice-Sørensen coefficient (DSC) [2],[17], which compares pixel-wise the desired and the actual output image of the sequence (1 means a perfect match, 0 vice-versa). It defines the similarity of two samples

\[
DSC = \frac{2 \sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij}Y_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{m} Y_{ij}},
\]

where \(X\) is a training pattern, \(Y\) is the output from the U-Net network. Of curse, unlike in the case of real-life images, sound, etc., in the paper, during the training, we did not use any data augmentation, the input data will always have the same configuration. This is obvious that we have to provide the sequence examples as they are, without any modification. Moreover, as the sequential mining data are very peculiar compared to images, sound, etc., we checked the impact of the minibatch value to the training. We found out that the best results were obtained with minibatch = 1, see Figure 4.
4.2 Choosing the right number of feature maps

To pick the right number of feature maps, we checked the impact of the network size on the performance and the training time. We present the results in Table 1. In the configuration shown, the number of all feature maps in the entire network has been multiplied. It shows that we chose the optimal size of the U-Net network for the presented task.

<table>
<thead>
<tr>
<th>FM number multiplier</th>
<th>Accuracy [%]</th>
<th>Training epoch time [seconds]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>98.310</td>
<td>272</td>
</tr>
<tr>
<td>0.5</td>
<td>98.880</td>
<td>362</td>
</tr>
<tr>
<td>1</td>
<td>99.326</td>
<td>697</td>
</tr>
<tr>
<td>2</td>
<td>99.325</td>
<td>1690</td>
</tr>
<tr>
<td>4</td>
<td>99.325</td>
<td>3785</td>
</tr>
</tbody>
</table>

4.3 Comparison to standard sequence mining algorithms

Here, we compare the U-Net network to standard sequence mining algorithms (2). In this experiment, we used the U-Net network with $128 \times 256$ character input size ($n = 128$ and $m = 256$). We generated $196 \times 10^4$ different random training sequences with injected random numbers (not from a sequence itself) and a dictionary of 128. The size of the search sequences was in the range of $[5,10]$ characters, and there were several sequences in each training input-output matrices (black and white images). For the analysis of the selected algorithms, we used the library (5); it requires that each transaction is separated by number -2. This element has been added after each last element belonging to the sequence. In the case of the U-Net network used in this experiment, we could encode up to 256 characters in one input-output image (as described earlier, the applied encoding result is similar to two-dimensional images). The first step was to check whether each of the sequences was correctly found by the algorithms and the U-Net network. In the experiment, all the algorithms returned the intended sequences. Then, we could proceed to check the neural network performance against the standard algorithms. We studied the impact of the minSupport parameter on the performance of the algorithms on the tested hardware. In the case of the neural network, it is not possible to provide this parameter; therefore, the processing time is constant for each case. We can observe that the network detects the sequence in each case faster than the GSP and SPADE algorithms. However, in the case of the BIDE+ algorithm (15), after changing the minSupport parameter to 0.003 (discovering longer sequences) it works slower than the U-Net network, what means that it is faster only for very short sequences. For longer sequences, the U-Net sequence discovery is faster, and the speed gain increases with the sequence length.
Table 2. Speed comparison of the proposed method to the standard sequence mining algorithms. SPADE was implemented in the parallelized version. In the case of U-Net, min_support was not taken into account.

<table>
<thead>
<tr>
<th>min support</th>
<th>GSP time[ms]</th>
<th>SPADE time[ms]</th>
<th>BIDE+ time[ms]</th>
<th>U-Net time[ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>9712</td>
<td>1118</td>
<td>7</td>
<td>364.8</td>
</tr>
<tr>
<td>0.3</td>
<td>9802</td>
<td>1119</td>
<td>8</td>
<td>364.8</td>
</tr>
<tr>
<td>0.2</td>
<td>10465</td>
<td>1169</td>
<td>8</td>
<td>364.8</td>
</tr>
<tr>
<td>0.1</td>
<td>9817</td>
<td>1169</td>
<td>9</td>
<td>364.8</td>
</tr>
<tr>
<td>0.05</td>
<td>9730</td>
<td>1272</td>
<td>23</td>
<td>364.8</td>
</tr>
<tr>
<td>0.01</td>
<td>10967</td>
<td>1299</td>
<td>204</td>
<td>364.8</td>
</tr>
<tr>
<td>0.005</td>
<td>13023</td>
<td>1431</td>
<td>238</td>
<td>364.8</td>
</tr>
<tr>
<td>0.003</td>
<td>17866</td>
<td>1159</td>
<td>365</td>
<td>364.8</td>
</tr>
<tr>
<td>0.002</td>
<td>2126</td>
<td>787</td>
<td>364.8</td>
<td></td>
</tr>
<tr>
<td>0.0005</td>
<td>3093</td>
<td>901</td>
<td>364.8</td>
<td></td>
</tr>
</tbody>
</table>

4.4 Server system logs analysis

We tested the proposed idea in practice to analyze the HDFS server logs from [3]. In this dataset, we wanted to detect a repetitive sequence of events that occurred before an error occurs. Below we present example entries:

```
081109 211204 1518 INFO dfs.DataNode \textdollar DataXceiver: Receiving block
081109 211204 1518 INFO dfs.DataNode \textdollar PacketResponder: PacketResponder 1 for block blk-3679882444868340280 terminating
081109 211204 1518 INFO dfs.DataNode \textdollar PacketResponder: Received block blk-3679882444868340280 of size 67108864 from /10.251.197.161
081109 211204 1520 ERROR dfs.DataNode \textdollar DataXceiver: 10.251.214.112:50010:DataXceiver: java.io.IOException: Block blk992101295951175683 is valid, and cannot be written to.
```

Fig. 5. Proposed idea of system log preprocessing to obtain the input for the neural network. We collect up to 20 entries before every error in logs and concatenate them in 256-element input vectors.

In the experiment, we took into account the last up to 20 entries before the entry with the "ERROR" label. We present the log preprocessing for the U-Net network in Figure 5. There were 258 entries with this label in our data set. We determined the number of 20 entries empirically. However, we assume that the most recent entries have the most significant contribution to failures. Analysis of what entries may indicate a failure is a much more complex problem requiring separate research. Then, selected entries in the log were grouped into 256-element input vectors.
so that the entries before one ERROR event were not divided into separate groups, i.e. if the group of 20 events exceeded 256 characters, the events were moved to the next testing vector. We encoded the events in a bitmap in the form of a one-hot vector for the U-Net network based on a different dictionary for each package. Thanks to this, the number of events in one input vector never exceeded 256 what kept the size of the U-Net network relatively small.

![Fig. 6. Proposed idea of the U-Net network for the detection of sequences before the failure on the server logs.](image)

5 Conclusion

We presented a method for discovering frequent sequences in computer system logs. The method can replace traditional sequence mining algorithms such as BIDE. We used the U-Net architecture trained with the desired sequences for a given input data. Beforehand we tried various types of convolutional neural networks, and only U-Net was able to detect sequences. We used one-hot number-level encoding for input data and the output (found) sequences. We also tried to use one-dimensional encoding without success. The method can analyse efficiently large datasets. Moreover, the workload can be distributed into many GPU devices. The method is less efficient than standard algorithms (e.g. GSP, SPADE) only in the case of very short sequences, usually, impractical in real-life applications. The most important comparison is with the BIDE algorithm as this is the fastest standard algorithm. Its speed comes from its design for closed sequences. Generally, the proposed method is more efficient for real-life usage as it analyses a long window at once. It should be mentioned that our method presented in the paper concerns finding any sequence, not matching patterns from the training data sets. The neural network finds new sequences, unseen before in the training data. In the current version of the system, we cannot determine the minimum support or time constraints.

Acknowledgement

The project financed under the program of the Polish Minister of Science and Higher Education under the name “Regional Initiative of Excellence” in the years 2019–2022 project number 020/RID/2018/19, the amount of financing 12,000,000.00 PLN.

References


Information for Authors

The Australian Journal of Intelligent Information Systems (AJIIPS) is published regularly. The aim of AJIIPS is to publish papers describing theory, methods, and techniques, applications, and tutorial presentations in a range of areas relating to Computer Science and Computer Engineering. AJIIPS publishes full papers, short notes, and survey articles. Specific areas include but are not limited to:

- Artificial Neural Networks
- Deep Learning
- Fuzzy Systems
- Evolutionary Computing
- Cluster Analysis
- Virtual Reality
- Extended Reality
- Ontological Studies
- Information Retrieval
- Data Mining and Knowledge Discovery
- Human Centred Computing

All papers will be reviewed by at least two reviewers and one of the editors. Decisions as to the suitability of the papers will be made by the Editor in Chief.

Papers should be clearly presented, consistent with giving proper description of primary contribution. Theory papers should be based on clear, formal foundations; methods and techniques should indicate the novelty and advantage of the technique; applications papers should present appropriate results and evaluation of performance; tutorial papers should be clearly presented to a reader with a general background in areas of interest to the journal but no prior background in the specific topic.

Submission and publication format

All papers should follow the Springer conference paper format, with the exception that 2 cm margins are used on all 4 sides of a A4 sheet.

Communication

Please use papers@ajiips.com.au to communicate with the journal, allowing 5 working days for reply. In exceptional circu