Research Article

A Temporal Pool Learning Algorithm Based on Location Awareness

Lei Li , Yuquan Zhu , Tao Cai , Dejiao Niu , Huaji Shi , and Tingting Zou

Department of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China

Correspondence should be addressed to Lei Li; lilei@ujs.edu.cn, Tao Cai; caitao@ujs.edu.cn, and Dejiao Niu; djniu@ujs.edu.cn

Received 22 March 2021; Revised 29 April 2021; Accepted 2 June 2021; Published 11 June 2021

Academic Editor: Chenxi Huang

Copyright © 2021 Lei Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Hierarchical Temporal Memory is a new type of artificial neural network model, which imitates the structure and information processing flow of the human brain. Hierarchical Temporal Memory has strong adaptability and fast learning ability and becomes a hot spot in current research. Hierarchical Temporal Memory obtains and saves the temporal characteristics of input sequences by the temporal pool learning algorithm. However, the current algorithm has some problems such as low learning efficiency and poor learning effect when learning time series data. In this paper, a temporal pool learning algorithm based on location awareness is proposed. The cell selection rules based on location awareness and the dendritic updating rules based on adjacent inputs are designed to improve the learning efficiency and effect of the algorithm. Through the algorithm prototype, three different datasets are used to test and analyze the algorithm performance. The experimental results verify that the algorithm can quickly obtain the complete characteristics of the input sequence. No matter whether there are similar segments in the sequence, the proposed algorithm has higher prediction recall and precision than the existing algorithms.

1. Introduction

Hierarchical Temporal Memory (HTM) is a new neural network model that simulates the organization and structure of cortical cells and the way the human brain processes information. HTM uses a large amount of sequence data for training [1]. By extracting and storing the patterns, HTM can predict the subsequent occurrence of the current input or detect whether the current input is abnormal. Therefore, HTM is widely applied in the analysis and processing of time series data. The basic unit of HTM is the cell. A number of cells constitute the minicolumn, and a large number of minicolumns form the network space of HTM. Spatial pool learning algorithm (SPL) and temporal pool learning algorithm (TPL) are two important learning steps of HTM. SPL establishes the mapping relationship between the input and the minicolumn set to form the sparse distributed representation (SDR) of inputs. TPL constructs or adjusts the corresponding dendrite branches among the cells to form a temporal correlation of inputs [2].

In the process of human learning, the human brain remembers many things. When given clues, the human brain can retrieve relevant knowledge with high efficiency. HTM works by imitating the way the human brain processes information. Using the memorized patterns, HTM can also efficiently obtain the complete pattern information by the clues. With SPL, the input can be quickly located and represented in HTM. TPL uses different combinations of active cells to express the context of input and constructs the temporal relationship between inputs in different environments. When an input enters HTM, HTM can quickly retrieve all follow-ups by activating all cells on the active minicolumn. With the continuous confirmation of the following inputs, HTM can identify and confirm the patterns containing sequential inputs. The retrieval efficiency of this method is much higher than that of the traditional comparison algorithm. Based on this, the trained HTM can be regarded as a warehouse of rules, which can detect whether the test sequence is abnormal. In this model, if the learning sequences are the rules, it is

worth discussing how to get all features of the rules quickly and ensure that the trained HTM has high prediction recall and precision.

First, TPL uses Hebbian rules to adjust the dendritic branches of cells to establish the temporal association between adjacent inputs. When the number of connected synapses in the dendrite branch of the cell accumulates to a certain threshold, TPL completes the extraction and memory of sequence features. TPL needs multiple iterations to memorize the sequence features, which affects the learning efficiency of TPL. Secondly, TPL gives priority to enhance the sequence features that have been learned rather than the current position sequence features, which affects the learning effect of TPL. In particular, when there are many continuous same inputs in the sequence, TPL is easy to fall into the same input cycle training and unable to obtain all features of the pattern, which affects the learning effect of TPL. Therefore, it is necessary to design a new algorithm to improve the learning efficiency and effect of TPL.

In this paper, a temporal pool learning algorithm based on location awareness (TPL_LA) is proposed. The generation rules of learning cells and active cells based on location awareness and the adjustment rules of dendrite branches associated with adjacent inputs are designed. Three datasets are used to test the performance of TPL_LA and TPL. The main innovations of this paper are as follows.

- (1) Using the multiple cells contained in the minicolumn, the generated learning cells and active cells can accurately express the position information of the input. In this way, the algorithm can give priority to learning the current sequence features, reduce the possibility of self-cycle prediction when learning continuous same inputs, and improve the learning effect of TPL.
- (2) According to the generated learning cell set and active cell set, the algorithm adjusts the updating rules of dendritic branches associated with adjacent inputs, and a new setting strategy of synaptic value is designed to improve the learning efficiency of TPL.
- (3) According to the generated learning cell set, the algorithm reduces the scope of active cells to predict the follow-up more accurately, which reduces the impact of historical memory on learning
- (4) In the open-source framework of HTM, the prototype of TPL_LA is implemented, and different types of data sets are used to test and analyze it. The results verify that TPL_LA could effectively improve the learning efficiency and effect compared with TPL.

The remainder of this paper is structured as follows: the related works for HTM are discussed in Section 2. Then, in Section 3, the efficient temporal pool learning algorithm based on location awareness is discussed. Finally, the experiments and analysis are carried out in Section 4. And the last section concludes our study.

2. Related Works

HTM is a new artificial neural network model based on Jeff Hawkins' memory prediction framework [3], and its main workflow is shown in Figure 1. The basic network space of HTM can be described as a region composed of multiple minicolumns. Regions and regions can have hierarchical relationships. There are many cells on a minicolumn. The cells on each minicolumn share a proximal dendrite to receive input stimuli. Every cell may have many branches of distal dendrites, which can build relationships with other cells [1]. The encoder converts the input into a binary vector of equal length, and the number of components with 1 in the vector is equal.

The vector can activate a small number of minicolumns to express the input content through SPL. TPL selects some cells on these activated minicolumns to express the input location information and establishes dendrite branches on these cells to construct the correlation between inputs. The sequence can be learned through the SPL and TPL to form memory in the network space of HTM. When the input sequence is consistent with the memorized pattern, the HTM can predict the next content based on the memory [4]. Two core algorithms, SPL and TPL, are used to construct the space-invariant and time-invariant features of input sequences, which are the key technologies for HTM to distinguish similar patterns.

For SPL, Ahmad and Hawkins give the characteristic description of sparse matrix and the expression form of information in the spatial pool [5], which provides theoretical guidance for the following research. Lattner Stefan uses the vector to study the spatial pool and proposes a formula for calculating the overlapping values and the corresponding learning rules [6]. Byrne describes the structure of HTM formally by using matrixes [7], which provides a mathematical basis for the derivation of rules. Leake et al. study and discuss the influence of initialization parameters on SPL [8]. Mnatzaganian et al. give a comprehensive mathematical framework of the spatial pool [9] so that the follow-up scholars can understand the core characteristics of SPL and improve the algorithm from a mathematical point of view. Li et al. propose an improved algorithm for spatial pool learning, which can make the training process of the spatial pool more efficient and stable [10]. In recent years, increasing scholars have begun to study the hardware implementation of the spatial pool to give full play to the advantages of HTM architecture [11]. Deliang Fan et al. propose that the computing blocks for HTM can be mapped using low-voltage, magneto-metallic spin-neurons combined with an emerging resistive crossbar network [12]. Krestinskaya et al. develop circuits and systems to achieve the optimized design of an HTM SP, an HTM TM, and a memristive analog pattern matcher for pattern recognition applications [13]. SPL abstracts the input features through a hierarchical structure [14], which makes HTM have wide-ranging applications in recognition and classification, such as data classification [15], face recognition [16], speech recognition [17], biometric recognition [18], detection of multiple objects located in clutter color images [19],

FIGURE 1: HTM workflow.

handwriting recognition [20], action recognition [21], gait recognition and understanding [22], and natural language processing [23].

For TPL, Hawkins and Ahmad give the formal expression of learning rules by matrixes [2], which gives a mathematical description of the rules related to active cells, learning cells, predicting cells, and synaptic renewal. This provides a theoretical basis for the subsequent study of the temporal pool learning algorithm. Younes Bouhadjar et al. reformulate the model in terms of a network of spiking neurons with continuous-time dynamics [24], in which they discuss how neuronal parameters such as cell-intrinsic time constants and synaptic weights constrain the sequence-processing speed. El-Ganainy et al. propose an improved Hierarchical Temporal Memory that can consider long-term dependence [25]. Suzugamine et al. propose a self-structured cortical learning algorithm that dynamically adjusts the numbers of columns and cells according to the input data [26]. In this model, the backtracking algorithm proposed by the author makes the prediction of the temporal pool more stable. In recent years, some scholars start to implement TPL on hardware [27]. Many scholars combine SPL with TPL and apply HTM to many fields with time series data, such as anomaly detection of time series data [28], heart attack detection [29], medical data flow prediction [25], hydrological intelligent monitoring [30], abnormal ECG detection [31], and abnormal detection in crowd management [32]. However, when the sequence data is learned as rules, TPL in the above research still has some limitations. For example, it needs multiple iterations to memorize the sequence features, it gives priority to enhancing the memory of a historical pattern when learning a sequence, and it is easy to fall into the same input cycle training when learning the sequence with many continuous same inputs. When TPL obtains all features of sequences, these limitations will lead to low learning efficiency and poor learning effect.

3. The Temporal Pool Learning Algorithm Based on Location Awareness

In TPL_LA, the learning process of TPL is reconstructed. The active cell and learning cell selection rules based on location awareness are proposed to make TPL_LA more focused on the current sequence features, which improves the learning effect of the algorithm. The dendrite updating rules associated with adjacent inputs are proposed to improve the learning efficiency of TPL_LA. The algorithm reduces the scope of active cells to predict the follow-up more accurately, which reduces the impact of historical memory on learning. The learning process of TPL_LA is shown in Figure 2.

3.1. Notation. The following notations are given for the HTM consisting of N minicolumns and M cells in each minicolumn:

- (1) W^t : the set of active minicolumns in HTM at time t . The set includes the identifiers of the activated minicolumns at time t . W^t is the output of SPL.
- (2) $A^t = \{a_{ij}^t\}$: the matrix of active cells at time t . A^t is a binary matrix of $M \times N$. a_{ij}^t indicates whether the i -th cell on the j -th minicolumn is active at the time of t . A value of 1 indicates an active state and a value of 0 indicates an inactive state.
- (3) $\Pi^t = \{\pi_{ij}^t\}$: the matrix of predictive cells at time t . Π^t is a binary matrix of $M \times N$. π_{ij}^t indicates whether the i -th cell on the j -th minicolumn is predictive at the time of t . A value of 1 indicates a predicted state and a value of 0 indicates an unpredicted state.
- (4) D_{ij}^k : the matrix of dendrite branch synaptic. D_{ij}^k is a binary matrix of $M \times N$. j is the minicolumn's number. i is the cell number on the minicolumn. k is the dendrite branch number of the cell. The matrix is

FIGURE 2: The learning process of TPL_LA.

used to store the synaptic permanence in the k -th dendrite branch between i -th cell of j -th minicolumn and other $m \times n^{-1}$ cells. Permanence is a scalar value, and its range is $[0.0, 1.0]$.

- (5) $\tilde{D}_{ij}^k$: the matrix of dendrite branch connectivity. $\tilde{D}_{ij}^k$ is a binary matrix of $M \times N$ constructed by the following equation. C_P is the threshold of connectivity.

$$\tilde{d}_{ij}^k = \begin{cases} 1, & d_{ij}^k \geq C_P, \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

- (6) $\dot{D}_{ij}^k$: the matrix of positive terms of dendrite branch synapses. $\dot{D}_{ij}^k$ is a binary matrix of $M \times N$ constructed by the following equation:

$$\dot{d}_{ij}^k = \begin{cases} 1, & d_{ij}^k > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

- (7) $\text{Cell}^t = \{\text{cell}_{ij}^t\}$: the matrix of learning cells at time t . Cell^t is a binary matrix of $M \times N$ which is a subset of A^t . Cell^t contains the cells whose dendrite branches need to be added or modified in the training process at time t . When $\text{cell}_{ij}^t = 1$, it means that the cell participates in the learning process of the current input.

- (8) I_P : initial synaptic permanence. I_P is a scalar with a value range of $[0.0, 1.0]$, indicating the permanence of the first connection between two cells.

3.2. Analysis of the Temporal Pool Learning Algorithm. Each input in the sequence continuously activates the microcolumns to express itself through SPL. TPL activates cells in these minicolumns to express the context of current input and uses active cells to predict the follow-up content. TPL selects a small number of cells from active cells as learning cells to reduce the number of dendritic branches on cells. By adjusting the permanence of dendritic branches owned by these learning cells, TPL constructs the association between the input before and after and learns the sequence features. Active cell generation rule, learning cell generation

rule, and dendritic branch synaptic adjustment method are important factors affecting the performance of TPL. Next, from the two aspects of learning efficiency and learning effect, this paper analyzes the implementation details existing in TPL, such as the selection rules of learning cells and active cells and the updating rules of dendritic branches.

3.2.1. Learning Efficiency. In HTM, cells maintain a large number of dendritic branches, which record the relationship between cells. With these dendrite branches, HTM can save the sequence relationship between inputs so that the TPL can predict the future according to the current input. When a cell creates a new dendritic branch, its synapses are usually not connected. After TPL learns the input many times, the Hebbian rule makes the permanence of the dendrite branch increase or decrease. The cell can become the prediction cell of the associated active cell, and TPL completes the memory of the sequence only when the connected synapses in the dendrite branch reach the threshold. Therefore, the sequence needs to be learned repeatedly before being memorized by HTM.

Secondly, in the minicolumn activated at time t , TPL uses the following equation to generate active cells:

$$a_{ij}^t = \begin{cases} 1, & \text{if } j \in W^t \text{ and } \pi_{ij}^{t-1} = 1, \\ 1, & \text{if } j \in W^t \text{ and } \sum_i \pi_{ij}^{t-1} = 0, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

If there are no predictive cells on an activated minicolumn at time t , TPL selects all cells on the minicolumn as active cells. TPL selects one of the cells as the learning cell to avoid creating too many dendritic branches in the learning process. TPL uses the following equation to select the learning cell on this minicolumn, which is called the best matching cell.

$$\begin{aligned} \text{cell}_{ij}^t = 1 \text{ where } & j \in W^t \text{ and } \sum_i \pi_{ij}^{t-1} \\ & = 0 \text{ and } \exists k \left(\left\| \dot{D}_{ij}^k \circ A_1^{t-1} \right\| > \text{min Threshold} \right) \\ & \text{and } \max_match \left(\left\| \dot{D}_{ij}^k \circ A_1^{t-1} \right\|_1 \right), \end{aligned} \quad (4)$$

where $\max_match(\|\dot{D}_{ij}^k \circ A_1^{t-1}\|_1)$ determines whether $\text{cell}_{i,j}$ contains a dendritic branch that overlaps with the last active cells the most among all cells on j minicolumn. The min-Threshold is the minimum threshold of overlap values of dendrite branches, which is used to avoid some invalid dendritic branches from participating in the calculation.

TPL selects a dendrite branch on the learning cell and enhances its synaptic value to strengthen its association with active cells at time $t-1$. The selected dendrite branch $\dot{D}_{ij}^k$ should satisfy the following formula:

$$\|\dot{D}_{ij}^k \circ A_1^{t-1}\|_1 = \max\left(\|\dot{D}_{ij}^k \circ A_1^{t-1}\|_1\right). \quad (5)$$

In this case, with the unpredicted activated minicolumn, the adjusted dendrite branches have the following two situations:

- (1) The dendrite branch constructs the association between cell_i^t and Cell^{t-1} . TPL adjusts the synaptic permanence to enhance the association between the current input and the previous input.
- (2) The dendrite branch does not construct the association between cell_i^t and Cell^{t-1} . TPL adjusts the synaptic permanence to enhance the characteristics of a historical subsequence.

The use of equation (4) increases the probability of Case 2, which makes TPL give priority to learning the features of repeated subsequences. When TPL cannot find the learning cell with equation (4), TPL will randomly select one with the least utilization rate as the learning cell on the unpredicted activated minicolumn. TPL constructs a new dendrite branch on the cell and establishes the association with the learning cells at time $t-1$. In this way, TPL can learn and memorize the current new sequence feature and add it to the last learning content. Therefore, TPL needs to learn a sequence repeatedly before it can memorize and store all characteristics, even if TPL directly sets the initial synapse permanence of the newly created dendritic branch to the connected state.

3.2.2. Learning Effect. TPL gives priority to learning the features of repeated subsequences rather than those of the current input sequence. Therefore, after a round of training of the input sequence, TPL can only memorize and store the repeated segments. These memories are fragmentary. Compared with the complete sequence, there are many breakpoints in memory. Although more iterative training can repair these memory breakpoints, it also needs more time. By repairing memory breakpoints, fragment memories can string into a longer memory, which can be called the memory axis. However, some sequence fragments will be memorized again in the new context and form memory branches on the memory axis. The following experimental results can verify this conclusion.

Besides, there may be continuous same inputs in the sequence. Although the contents of these inputs are the same, their meanings are different due to their different positions. TPL selects different learning cells from the

activated minicolumns to represent the different contexts of input. The cells associated with learning cells also come from these minicolumns. When TPL learns the continuous same inputs, it is easy to make the input cycle predict itself. In this case, TPL will no longer learn the same new input, which will affect the training effect.

As shown in Figure 3, HTM has learned b for P times, and each b has different learning cells due to its different positions. When TPL learns the subsequent b , the active cells of the input may be a subset of the predicted cells, i.e., $A^t = A^{t-1} \cap A^t$. This leads to the fact that the modified dendrite branches only strengthen the memory of history, and TPL no longer learns new repetitive content. Therefore, TPL cannot effectively learn repeated inputs, and HTM may only form one or more memory segments of the sequence.

When TPL learns a sequence, the memory generated by HTM usually has memory breakpoints and memory branches, which will affect the prediction effect. Memory breakpoints indicate that TPL cannot form a complete memory of a sequence, and they will reduce the recall of predictions. Memory branches will reduce the precision of prediction.

3.3. Cell Selection Rules Based on Location Awareness. In TPL_LA, the cell selection rules based on location awareness are designed. The rules reduce the range of active cells and no longer use the best matching cells. In this way, TPL_LA can distinguish the same input in different positions and give priority to learning the features of the current sequence.

When there are predictive cells on the activated minicolumns at time t , these cells contain active dendritic branches. Adjusting these dendrite branches will learn the sequence characteristics of the current position. TPL_LA uses the following equation to set the cell as a learning cell.

$$\text{cell}_{ij}^t = 1 \quad \text{where } j \in W^t \text{ and } \pi_{ij}^{t-1} = 1. \quad (6)$$

When there are not predictive cells on the activated minicolumns at time t , these cells do not contain active dendritic branches. TPL_LA uses the following equation to set the cell as a learning cell, where $\min_used(\text{cell}_{i,j})$ determines whether $\text{cell}_{i,j}$ is the least used cell on j minicolumn.

$$\text{cell}_{ij}^t = 1 \quad \text{where } \min_used(\text{cell}_{i,j}) \text{ and } (j \in W^t) \text{ and } \left(\sum_i \pi_{ij}^{t-1} = 0\right). \quad (7)$$

TPL_LA adds dendrite branches to the learning cell to construct the association with Cell^{t-1} to learn the sequence characteristics of the current position.

Finally, based on the selected learning cells, TPL_LA uses the following equation to create active cells.

$$A^t = \text{Cell}^t. \quad (8)$$

Different from the way TPL generates the learning cells and active cells, the algorithm no longer selects the best matching cells but randomly selects the cells with the least frequency as the learning cells with the unpredicted activated

FIGURE 3: An example of repeated inputs in a sequence.

minicolumns. Therefore, TPL_LA can accurately express the context of the current input and learn the sequence features of the current position. Active cells usually express multiple contexts of input. The algorithm generates the active cells according to the learning cells, so it can accurately express the current context of the input. Such active cells reduce the prediction of historical content, which makes TPL learn the sequence characteristics of the current position and reduces the occurrence of memory breakpoints and memory branches. Besides, when TPL_LA learns the same continuous inputs, such active cells can also reduce the cycle prediction of the input and improve the learning effect of TPL_LA. The purpose of the proposed algorithm is to obtain the complete features of the rule. Every input owns different cell combinations to express the context through location awareness. To some extent, this increases the number of dendritic branches on the cell. This strategy will take up more storage space but improve the accuracy of prediction.

3.4. Dendrite Updating Rules Associated with Adjacent Inputs. In TPL_LA, the updating rules of dendritic branches associated with adjacent inputs are designed, which makes TPL_LA quickly complete the learning of sequence features.

By adjusting the active dendrite branches of learning cells generated by equation (6), TPL_LA can learn the sequence features of the current position. TPL_LA uses the following equation to select the dendrite branches to be adjusted. The a_T is the activation threshold of dendritic branches.

$$\forall_{j \in W^i} (\pi_{ij}^{t-1} > 0) \text{ and } \tilde{D}_{ij}^k \circ A_1^{t-1} > a_T. \quad (9)$$

Reinforcing the above segments is to decrease all permanence values by a small value p^- and increase the permanence values corresponding to active presynaptic cells by a larger value p^+ [2].

$$D_{ij}^k = D_{ij}^k + p^+ (\tilde{D}_{ij}^k \circ A^{t-1}) - p^- \tilde{D}_{ij}^k. \quad (10)$$

By adding the active dendrite branches to the learning cells generated by equation (7), TPL_LA can establish the association between the current learning cell and $Cell^t$ and learn the sequence features of the current position. The algorithm adds a new dendrite branch D_{ij}^{new} according to the following equation. I_P is usually greater than the connectivity threshold of synapses.

$$D_{ij}^{new} = Cell^{t-1} \times I_P. \quad (11)$$

According to the above rules, either the selected active dendritic branches or the newly added dendritic branches construct the association between learning cells at the time

before and after. All of these make learning aim at the current sequence and reduce the interference of historical memory on current learning. When the synapse permanence is greater than or equal to the connectivity threshold, the synapse is connected. Therefore, using a bigger I_P , TPL_LA can quickly construct the temporal association of the sequence, extract the features of the memory sequence, and improve the learning efficiency.

3.5. The Pseudocode of the Proposed Algorithm. Based on Sections 3.3 and 3.4, the pseudocode of TPL_LA is shown as follows. (Algorithm 1).

TPL_LA modifies the rules of TPL. By sensing the position information of the input in the sequence, TPL_LA selects different learning cells to distinguish the same input. By reducing the range of active cells and no longer using the best matching cells as learning cells, TPL_LA only learns the sequence features at the current position. In this way, the memory of HTM rarely produces memory breakpoints and memory branches, and the algorithm can memorize all features of the sequence. Therefore, the learning effect of TPL_LA is highly improved. By increasing the synaptic permanence in the new dendrite branch of the learning cell, TPL_LA can learn the features of the sequences faster. Therefore, the algorithm not only guarantees the learning effect but also has high learning efficiency.

4. Experiments and Analysis

In the open-source framework of HTM, the prototype of TPL_LA is implemented. Different types of datasets are used to test and analyze it compared to TPL.

4.1. Datasets and Evaluation Indicators. In this paper, three datasets are used to test and analyze TPL_LA and TPL. Fifty digits after the π decimal point are used to construct the first dataset representing a random sequence. A sequence such as “21111..... 11113” is used as the second dataset to represent a sequence with continuously repeated inputs, whose length is set to 50. The third dataset is the statistical data of New York taxi passengers. From July 1, 2014, to January 31, 2015, the number of passengers per 30 minutes of New York taxi is used to construct the input sequence, with a total of 10320 data.

These three datasets are used to train the two algorithms, respectively, and the corresponding datasets are used for prediction tests. The test results are used to evaluate the learning effect of whether the two algorithms can obtain all features of the sequence. Three evaluation indicators, i.e., mean recall (MR), mean precision (MP), and F1, are used to evaluate the two algorithms. The $recall^t$ describes whether the input is within the prediction range at time t . The $precision^t$ describes the proportion of the input in the prediction range at time t . The evaluation indicators are calculated using the following equations:


```

(1) Get  $W^t$  of current input from spatial pool:
(2) Get  $A^{t-1} \Gamma^{t-1} Cell^{t-1}$  from last time
(3) For  $j = 0$  to  $N - 1$  do:
(4)   if  $j \in W^t$  then
(5)     for  $i = 0$  to  $M - 1$  do
(6)       if  $\pi_{ij}^{t-1} == 1$  then
(7)         set the current cell to  $cell_{i,j}^t$ 
(8)         if (learn) then
(9)           reinforce the segment of the cell;
(10)        endif
(11)      else
(12)        select the less used cell to  $cell_{i,j}^t$ 
(13)        if (learn)
(14)          create connected segment between  $cell_{i,j}^t$  and  $cell^{t-1}$ 
(15)        endif
(16)      endif
(17)    endfor
(18)  else
(19)    if  $\pi_{ij}^{t-1} == 1$  then
(20)      if (learn) punish the segment of the cell endif
(21)    endif
(22)  endif
(23) endfor
(24) if (learn)  $A^t = Cell^t$  endif
(25) Compute the  $\Gamma^t$ 

```

ALGORITHM 1: TPL_LA.

$$\begin{aligned}
recall^t &= \begin{cases} 1, & \text{if } A^t \circ \Gamma^{t-1} > a_T, \\ 0, & \text{otherwise,} \end{cases} \\
precision^t &= \begin{cases} \frac{\|A^t\|}{\|\Gamma^{t-1}\|}, & \text{if } recall^t = 1, \\ 0, & \text{otherwise,} \end{cases}
\end{aligned} \tag{12}$$

$$MR = \frac{1}{L} \sum recall^t,$$

$$MP = \frac{1}{L} \sum precision^t,$$

$$F1 = \frac{1}{1/MR + 1/MP} = \frac{2 \times MR \times MP}{MR + MP}.$$

L is the length of a sequence. When the values of MR and MP are both large, it shows that the algorithm has a high degree of restoration for the input sequence. The larger the value of F1, the better the learning effect of the algorithm.

4.2. The Testing with Random Inputs. The sequence with random inputs is constructed using dataset 1. Figure 4 is a graphical representation of each input in dataset 1.

Dataset 1 has a small input range, so the prototype system uses Table 1 to set the relevant parameters.

FIGURE 4: A sequence with random inputs.

TABLE 1: Parameters for a random sequence test.

Parameter	Parameter value	Meaning
C_D	128	Column dimensions
C_P_C	16	Cells per column
I_P	Pending	Initial permanence
C_P	0.8	Connected permanence
P_I	0.1	Permanence increment
P_D	0.05	Permanence decrement

4.2.1. The Comparison of Testing Results. When learning a sequence, TPL_LA needs larger initial value synapses. Therefore, I_P is set to the connectivity threshold of 0.8 and TPL_LA trains the sequence only once. TPL is tested in two cases. (1) TPL sets I_P to the connective threshold of 0.8 and trains the sequence once. (2) TPL sets I_P to the disconnected value of 0.7 and trains the sequence one and more times. Table 2 shows the testing results.

TABLE 2: Test results with random input.

Algorithm	I_P	Iterations	MR	MP	F1
TPL	0.7	1	0.32	0.25	0.28
TPL	0.7	10	0.90	0.74	0.81
TPL	0.7	50	0.96	0.85	0.90
TPL	0.7	100	0.98	0.89	0.93
TPL	0.8	1	0.82	0.59	0.69
TPL_LA	0.8	1	0.98	0.96	0.97

Through F1, it can be seen that TPL_LA is better than TPL. From the perspective of MR or MP, the results of the TPL_LA training sequence once are better than those of TPL in various situations. TPL_LA gives a distinct context to the input in different positions, so the recall and precision of the prediction are both the highest. In the learning process of TPL with connected I_P, repeated content is easy to form memory breakpoints. In addition, without repeated iterative learning, these breakpoints are difficult to repair, so the recall and precision of the prediction cannot reach the high point, which reduces the value of F1. In the learning process of TPL with unconnected I_P, nonrepetitive content is easy to form memory breakpoints. Repeated iterative learning can repair these memory breakpoints and produce some memory branches. Therefore, the learning result makes the recall of prediction very high but reduces the precision of prediction, which slightly reduces the value of F1. Although the TPL can also achieve a high approval of recall or precision through iterative learning, it needs more learning time. Therefore, TPL_LA has better learning efficiency and learning effect when learning a random sequence.

4.2.2. The Prediction State of Every Step. To better analyze the results in Table 2, the prediction states of each step when the two algorithms process dataset 1 are listed in Figure 5.

As can be seen from Figure 5(a), when TPL sets I_P disconnected and learns the sequence once, HTM only remembers the repeat patterns. Some patterns do not repeat in the dataset, which lets the synaptic value between the inputs of these patterns not reach the connectivity state, so HTM does not memorize these patterns. Compared with the complete sequence, there are many breakpoints in memory, so the recall and precision of prediction are very low. It can be seen from Figure 5(b) that when TPL sets I_P disconnected and learns the sequence 100 times, the memory in HTM can restore the complete sequence. Through iterative learning, TPL strengthens the disconnected synapses between inputs to the connected state, so these nonrepetitive patterns are also memorized. At the same time, the repeated patterns in different positions are given different contexts, so some memory branches are formed in HTM. Therefore, the recall of prediction is very high, and the precision of prediction is not high. It can be seen from Figure 5(c) that when TPL sets I_P connected and learns the sequence once, HTM memorizes long sequence fragments. However, memory breakpoints are formed at the position of each repetitive subsequence. The memory breakpoint will make the subsequent input have no context, which causes multiple

following predictions. As a result, the recall and precision of prediction are not very high. It can be seen from Figure 5(d) that when TPL_LA learns the sequence once, the memory in HTM can restore the complete sequence. The algorithm gives a distinct context to the input in different positions, so the memory in HTM has few memory breakpoints and memory branches. Therefore, the recall and precision of the prediction are both the highest.

4.3. The Testing with Repeated Inputs. The sequence with continuously repeated inputs is constructed using dataset 2. There are only three input types in the sequence, so the algorithms still use the parameters in Table 1.

4.3.1. The Comparison of Testing Results. For the sequence with continuously repeated inputs, the test is still carried out with the plan designed in Section 4.2.1. Table 3 shows the testing results.

Through F1, it can be seen that TPL_LA is better than TPL. From the perspective of MR or MP, the results of the TPL_LA training sequence once are better than those of TPL in various situations. TPL_LA also gives a distinct context to the input in different positions, so the recall and precision of the prediction are both the highest. In the learning process of TPL with connected I_P, continuous repeated content can be seen as the repeated iterative learning, which can repair the memory breakpoints. However, the continuous repeated content makes the active cell set the same as the predicted cell set, which prevents the algorithm from memorizing longer sequences. Therefore, the recall and precision of the prediction cannot reach a high point, which reduces the value of F1. In the learning process of TPL with unconnected I_P, repeated iterative learning can repair more memory breakpoints. However, because the active cell set is the same as the predicted cell set, the algorithm cannot memorize longer sequences. Therefore, the learning result makes the recall and precision of the prediction cannot reach the high point, which reduces the value of F1. Therefore, TPL_LA has better learning efficiency and learning effect when learning a sequence with continuously repeated inputs.

4.3.2. The Prediction State of Every Step. To better analyze the results in Table 3, the prediction states of each step when the two algorithms process dataset 2 are shown in Figure 5.

As can be seen from Figure 6(a), when TPL sets I_P disconnected and learns the sequence once, HTM only remembers a segment of the sequence. The existing algorithms will give priority to strengthening the repeated patterns. When the previous patterns are memorized, the algorithm will extend the memory content and increase the length of the memory axis. Since HTM only learns part of the sequence, the recall and precision of prediction are not high. As can be seen from Figure 6(b), when TPL sets I_P disconnected and learns the sequence 100 times, the memory in HTM still cannot restore the complete sequence. Through iterative learning, when the memorized sequence reaches a certain length, the active cells and prediction cells will be the

FIGURE 5: The prediction processes with a random sequence.

TABLE 3: Test results with repeated inputs.

Algorithm	I_P	Iterations	MR	MP	F1
TPL	0.7	1	0.82	0.82	0.82
TPL	0.7	10	0.88	0.82	0.85
TPL	0.7	20	0.90	0.80	0.85
TPL	0.7	50	0.90	0.80	0.85
TPL	0.8	1	0.88	0.86	0.87
TPL_LA	0.8	1	0.98	0.98	0.98

same, which leads to the algorithm not learning new repeated content. At the same time, the repeated pattern, especially the “13,” owns different contexts, so some memory branches are formed in HTM. Therefore, the recall of prediction is not improved much, and the precision of prediction is decreased. As can be seen from Figure 6(c), when TPL sets I_P connected and learns the sequence once, HTM still only remembers a segment of the sequence. The reason is also that when the repeat sequence is memorized to a certain length, the active cells and the prediction cells will be the same, resulting in the fact that the algorithm could not

learn new content. The algorithm learns the sequence “13” only once, so the pattern has only one context. Compared with the results of Figure 6(c), the recall and the precision of the prediction are not much different. As can be seen from Figure 6(d), when TPL_LA learns the sequence once, the memory in HTM can restore the complete sequence. The algorithm gives different contexts to the input in distinct positions, so the memory in HTM has few memory breakpoints and memory branches. Therefore, the recall and precision of the prediction are both the highest.

4.4. The Testing of the Taxi Passenger Flow of New York. The real data sequence is constructed by dataset 3. Because the dataset is much larger than the first two datasets, TPL_LA and TPL use Table 4 to set the parameters.

In this testing, TPL_LA set I_P to 0.8. TPL is tested in two cases. (1) TPL sets the I_P to 0.8. (2) TPL sets I_P to 0.7. Each algorithm only learns the data set once under the corresponding configuration.

The dataset has 10320 data and 8090 different numbers. This dataset may be seen as the first level dataset, which

TABLE 5: The test results of the taxi passenger flow (first level dataset).

Algorithm	INITIAL_PERMANENCE	Iterations	MR	MP	F1
TPL	0.7	1	0.0	0.0	0.0
TPL	0.8	1	0.95697	0.95697	0.95697
TPL_LA	0.8	1	0.9999	0.9999	0.9999

TABLE 6: The test results of the taxi passenger flow (second level dataset).

Algorithm	INITIAL_PERMANENCE	Iterations	MR	MP	F1
TPL	0.7	1	0.01337	0.01166	0.01246
TPL	0.8	1	0.67955	0.67142	0.6755
TPL_LA	0.8	1	0.9999	0.9999	0.9999

TABLE 7: The test results of the taxi passenger flow (third level dataset).

Algorithm	INITIAL_PERMANENCE	Iterations	MR	MP	F1
TPL	0.7	1	0.35193	0.0802	0.13063
TPL	0.8	1	0.4718	0.19436	0.2753
TPL_LA	0.8	1	0.9999	0.9999	0.9999

when there are repeated contents in the sequence or HTM learns the sequence more times can the algorithm extract and memorize the features of the sequence.

5. Conclusions

Through the analysis of TPL, the authors explain the reasons for the low learning efficiency and poor learning effect of TPL and propose an efficient temporal pool learning algorithm based on location awareness. After one training of TPL_LA, the sequence can be effectively memorized by HTM, which shows that the algorithm has good learning efficiency and learning effect.

In the following research, the authors intend to add more input feature information to the TPL_LA and modify the learning mechanism of the proposed algorithm, which could enhance the smoothness of the memory, make the predictive generalization higher, and improve the utilization rate of the network space of the HTM. In this way, the proposed algorithm can be better used in the intelligent network in the future.

Data Availability

The artificial data used to support the findings of this study are included within the article. The time-period statistics of New York taxi passengers can be found at <https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research work was supported in part by the National Natural Science Foundation of China (grant no. 61806086), in part by the Project of National Key R&D Program of

China (2018YFB0804204), and in part by Jiangsu Provincial doctoral Innovation Fund (CXZZ13_0689).

References

- [1] J. Hawkins, S. Ahmad, and D. Dubinsky, Hierarchical temporal memory including htm cortical learning algorithms," http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf, 2011.
- [2] J. Hawkins and S. Ahmad, "Why neurons have thousands of synapses, a theory of sequence memory in neocortex," *Frontiers in Neural Circuits*, vol. 10, pp. 1–13, 2016.
- [3] D. George, "How the brain might work: a hierarchical and temporal model for learning and recognition," Ph.D Thesis, Stanford University, Kunnamangalam, Kerala, 2008.
- [4] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin, "Biological and machine intelligence (Bami)," <http://numenta.com/biological-and-machine-intelligence/>, 2016.
- [5] S. Ahmad and J. Hawkins, "Properties of sparse distributed representations and their application to hierarchical temporal memory," 2015, <https://arxiv.org/abs/1503.07469>.
- [6] S. Lattner, "Hierarchical temporal memory-investigations, ideas, and experiments," M.Sc. thesis, Linz: Johannes Kepler Universität, Linz, Austria, 2014.
- [7] F. Byrne, "Encoding reality: prediction-assisted cortical learning algorithm in hierarchical temporal memory," 2015, <https://arxiv.org/abs/1509.08255>.
- [8] M. Leake, L. Xia, K. Rocki, and W. Imaino, "A probabilistic view of the spatial pooler in hierarchical temporal memory," *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, pp. 1111–1118, 2015.
- [9] J. Mnatzaganian, E. Fokoué, and D. Kudithipudi, "A mathematical formalization of hierarchical temporal memory's spatial pooler," *Frontiers in Robotics and AI*, vol. 3, p. 81, 2017.
- [10] L. Li, T. Zou, T. Cai, D. Niu, and Y. Zhu, "A fast spatial pool learning algorithm of hierarchical temporal memory based on minicolumn's self-nomination," *Computational Intelligence*

- and Neuroscience, vol. 2021, Article ID 6680833, 13 pages, 2021.
- [11] D. Fan, M. Sharad, A. Sengupta, and K. Roy, "Hierarchical temporal memory based on spin-neurons and resistive memory for energy-efficient brain-inspired computing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 9, pp. 1907–1919, 2016.
 - [12] O. Krestinskaya, I. Dolzhikova, and A. P. James, "Hierarchical temporal memory using memristor networks: a survey," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 5, pp. 380–395, 2018.
 - [13] O. Krestinskaya, T. Ibrayev, and A. P. James, "Hierarchical temporal memory features with memristor Logic circuits for pattern recognition," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 6, pp. 1143–1156, 2018.
 - [14] D. Maltoni, "Pattern recognition by hierarchical temporal memory," *SSRN Electronic Journal*, 2011.
 - [15] A. Irmanova, T. Ibrayev, and A. P. James, "Discrete-level memristive circuits for HTM-based spatiotemporal data classification system," *IET Cyber-Physical Systems: Theory & Applications*, vol. 3, no. 1, pp. 34–43, 2018.
 - [16] T. Ibrayev, U. Myrzakhan, O. Krestinskaya, A. Irmanova, and A. P. James, "On-chip face recognition system design with memristive Hierarchical Temporal Memory," *Journal of Intelligent & Fuzzy Systems*, vol. 34, no. 3, pp. 1393–1402, 2018.
 - [17] M. Wielgosz and M. Pietroń, "Using spatial pooler of hierarchical temporal memory to classify noisy videos with predefined complexity," *Neurocomputing*, vol. 240, pp. 84–97, 2017.
 - [18] A. P. James, I. Fedorova, T. Ibrayev, and D. Kudithipudi, "HTM spatial pooler with memristor crossbar circuits for sparse biometric recognition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 3, pp. 640–651, 2017.
 - [19] R. Skoviera, I. Bajla, and J. Skovierová, "Object recognition in clutter color images using Hierarchical Temporal Memory combined with salient-region detection," *Neurocomputing*, vol. 307, pp. 172–183, 2018.
 - [20] F. Asgari and A. Salehi, "The biologically inspired hierarchical temporal memory model for farsi handwritten digit and Letter recognition," *International Journal of Computer Applications*, vol. 129, no. 16, pp. 6–11, 2015.
 - [21] Y. Zou, Y. Shi, Y. Wang, Y. Shu, Q. Yuan, and Y. Tian, "Hierarchical temporal memory enhanced one-shot distance learning for action recognition," in *Proceedings of the 2018 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, San Diego, CA, USA, July 2018.
 - [22] J. Luo and T. Tjahjadi, "Gait recognition and understanding based on hierarchical temporal memory using 3D gait semantic folding," *Sensors (Basel, Switzerland)*, vol. 20, 2020.
 - [23] A. Irmanova, "Image based HTM word recognizer for language processing," in *Proceedings of the 2018 IEEE International Conference on Consumer Electronics—Asia (ICCE-Asia)*, pp. 206–212, JeJu, South Korea, June 2018.
 - [24] Y. Bouhadjar, M. Diesmann, R. Waser, D. J. Wouters, and T. Tetzlaff, "Constraints on sequence processing speed in biological neuronal networks," in *Proceedings of the International Conference on Neuromorphic Systems (ICONS '19)*, Knoxville, TN, USA, 2019.
 - [25] N. O. El-Ganainy, I. Balasingham, P. S. Halvorsen, and L. Arne Rosseland, "On the performance of hierarchical temporal memory predictions of medical streams in real time," in *Proceedings of the 2019 13th International Symposium on Medical Information and Communication Technology (ISMICT)*, pp. 1–6, Oslo, Norway, May 2019.
 - [26] S. Suzugamine, T. Aoki, K. Takadama, and H. Sato, "Self-structured cortical learning algorithm by dynamically adjusting columns and cells," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 24, no. 2, pp. 185–198, 2020.
 - [27] A. M. Ziyarah and D. Kudithipudi, "Neuromorphic architecture for the hierarchical temporal memory," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 3, no. 1, pp. 4–14, 2019.
 - [28] A. Anandharaj and P. B. Sivakumar, "Anomaly detection in time series data using hierarchical temporal memory model," in *Proceedings of the 2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1287–1292, Coimbatore, India, June 2019.
 - [29] T. Charrad, K. Nouira, and A. Ferchichi, "Deconstructing RPCs," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6S2, pp. 312–315, 2019.
 - [30] H. Luo, S. Jia, and W. Zhang, "Hierarchical temporal memory based anomaly detection for hydrological monitoring of unmanned surface vehicle," in *Proceedings of the 2019 IEEE 2nd International Conference on Information Communication and Signal Processing (ICICSP)*, pp. 420–424, Weihai, China, September 2019.
 - [31] W. Midani, Z. Fki, and M. BenAyed, "Online anomaly detection in ECG signal using hierarchical temporal memory," in *Proceedings of the 2019 fifth international Conference on Advances in Biomedical engineering (ICABME)*, pp. 1–4, Weihai, China, September 2019.
 - [32] A. Bamaqa, M. Sedky, T. Bosakowski, and B. B. Bastaki, "Anomaly detection using hierarchical temporal memory (HTM) in crowd management," in *Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing*, London, UK, August 2020.