ISSN 1060-992X, Optical Memory and Neural Networks, 2023, Vol. 32, Suppl. 2, pp. S284–S292. © The Author(s), 2023. This article is an open access publication.

Attractor Properties of Spatiotemporal Memory in Effective Sequence Processing Task

P. Kuderov[a, b, c, *], **E. Dzhivelikian**[a], **and A. I. Panov**[b, c]

[a] *Moscow Institute of Physics and Technology, Dolgoprudny, Moscow Oblast, 141701 Russia*

[b] *Artificial Intelligence Research Institute, Moscow, Russia*

[c] *Federal Research Center "Computer Science and Control", Russian Academy of Sciences, Moscow, 119333 Russia*

**e-mail: kuderov.pv@phystech.edu*

Received August 15, 2023; revised September 1, 2023; accepted September 5, 2023

Abstract—For autonomous AI systems, it is important to process spatiotemporal information to encode and memorize it and extract and reuse abstractions effectively. What is natural for natural intelligence is still a challenge for AI systems. In this paper, we propose a biologically plausible model of spatiotemporal memory with an attractor module and study its ability to encode sequences and efficiently extract and reuse repetitive patterns. The results of experiments on synthetic and textual data and data from DVS cameras demonstrate a qualitative improvement in the properties of the model when using the attractor module.

Keywords: sequential memory, spatiotemporal abstractions, spatial pooler, temporal memory, spatial attractor

DOI: 10.3103/S1060992X23060097

1. INTRODUCTION

Modern artificial intelligence systems are often interactive and therefore rely on the ability to process and remember sequentially incoming information [1, 2]. Unlike the human brain, for which the continuous processing of spatiotemporal information is natural, this is still a challenging task for AI systems [3, 4].

In this paper, we propose a biologically plausible model of spatiotemporal memory with an attractor module. We study its ability to efficiently encode sequences and extract and reuse repetitive patterns in experiments on synthetic and textual data, and data from DVS cameras. We demonstrate qualitative and quantitative improvements of the sequence memory when using the attractor module and compare its performance to the LSTM model [5]. Experiments show that the attractor module significantly increases the temporal memory's capacity and its convergence speed. We also investigate properties of the attractor module itself and test significance of its different components.

The remainder of the paper is structured as follows: The biologically plausible paradigm for the neocortex modeling that serves as the foundation of our study is introduced in Section 2. Our temporal memory model is described in Section 3. Section 4 describes the experimental setup that was used as well as the outcomes. The results are discussed in Section 5, which also offers suggestions for future research.

2. BACKGROUND

Our spatiotemporal memory model is inspired by HTM framework, which describes a discrete-time multi-compartment computational model of a pyramidal neuron with active dendrites of the cerebral cortex and hippocampus [6, 7]. Such neuron has its dendritic synapses grouped into segments and then into compartments. The proximal basal compartment has a single segment and is responsible for the feedforward input from other layers, while the distal basal compartment unites segments with synapses to neurons of the same layer. A single segment defines the unit of neuron activation—segments become active based on the activity of their receptive field, regardless of the activity of the others.

There are two sub-algorithms that define important building blocks in our model: a spatial pooler and a temporal memory.

Spatial Pooler (SP) is a neural network algorithm capable of encoding input stimuli patterns using the local Hebbian rule [8]. The main function of SP is the spatial specialization of neurons' receptive fields through k-WTA (k winners take all) inhibition [9]. SP output activity is a sparse distributed representation (SDR), that is only a small percentage of neurons are active at each moment of time and the semantics is spread across them. There are evidences that mammal brains might employ an analogous strategy for coding, which is efficient and noise tolerant [10]. One of the key properties of our model is that all its parts operate and communicate with SDRs.

Temporal Memory (TM) is a recurrent neural network algorithm that implements the sequential pattern memory [6, 11]. TM neurons are organized into ensembles called minicolumns. The neurons of the minicolumn share proximal basal segment's receptive field and therefore recognize the same feedforward input pattern. Activity of TM neurons is determined on the basis of two parallel factors: the relative ability of minicolumns to recognize an incoming feedforward spatial pattern and the threshold ability of neurons to recognize spatiotemporal patterns of the context arriving at the distal compartment from neurons of the same layer [12–14]. As the result, active minicolumns encode feedforward input, while active neurons represent TM's hidden state.

3. METHODS

3.1. Sequence Memory Model

In this work we propose a sequence memory model that consists of three modules: (1) the spatial encoder for the input sequence elements, (2) the sequence memory itself, and (3) the memory hidden state attractor. An illustration of the model is shown in Fig. 1.

3.1.1. Spatial encoder. The spatial encoder (SE) is a module that accepts current observation o_t and transforms it to the latent state SDR z_t. Its main role is to prepare input for easier processing while retaining input patterns pairwise similarity to keep as much information about sequence elements as possible. We expect it help clusterize/separate entire sequences resulting in the easier memorization task for the sequence memory. SE is based on SP algorithm and is tuned for these properties. If an input stream has already been encoded appropriately in terms of sparsity and topology semantics, using SE is optional. Likewise, since SE learns online, the output representation can be highly unstable during the early phases, which could bring performance drop.

3.1.2. Temporal memory. TM module represents sequence memory itself and is implemented with a high-order temporal memory (TM). Given feedforward SDR z_t, TM updates its hidden state h_t, which is then used to predict the next hidden state $\tilde{h}_{t+1}$. The high efficiency of temporal memory is largely due to its ability to represent the context of the observed element of a sequence, which is the sequence's prefix, in a unique way. While the TM algorithm efficiently separates different contexts, it is much less efficient at representing similar contexts with similar latent states. For the task of perfect memorization of near-symbolic sequences and anomaly detection, these properties may be desirable. But for the tasks that require sequence abstraction and generalization, it is a major drawback.

3.1.3. Attractor. We propose the attractor module to help TM consolidate similar contexts. It is an attractor network operating in the TM latent state space with the purpose of correcting TM predictions. The attractor module is expected to push predictions toward the strongest nearest neighbour memories. The module has two parts: spatial attractor and associative attractor.

3.1.4. Spatial attractor. A spatial attractor (SA) is a neural network implemented with the Spatial Pooler algorithm. It gets the flattened vector of the predicted TM hidden state SDR $\tilde{h}_{t+1}$ as input and transforms it recurrently several times—that is, the output from the spatial attractor w_t^i is fed to itself as an input, and so on for k iterations.

The role of the spatial attractor in the learning process is to form stable representations for the recurring TM hidden states, which represent sequences (or their prefixes). We expect that the spatial attractor trained with the Hebbian rule forms an attractor network in which representations corresponding to the frequently observed subsequences act as attractors. This could allow us to integrate contexts corresponding to similar [sub-]sequences.

Note that the SDR w_t^k, which is meant to correct the temporal memory prediction $\tilde{h}_{t+1}$, no longer corresponds to the columnar structure of the temporal memory, because SA learns without supervision. As the result, the attractor module needs another part—the associative attractor.

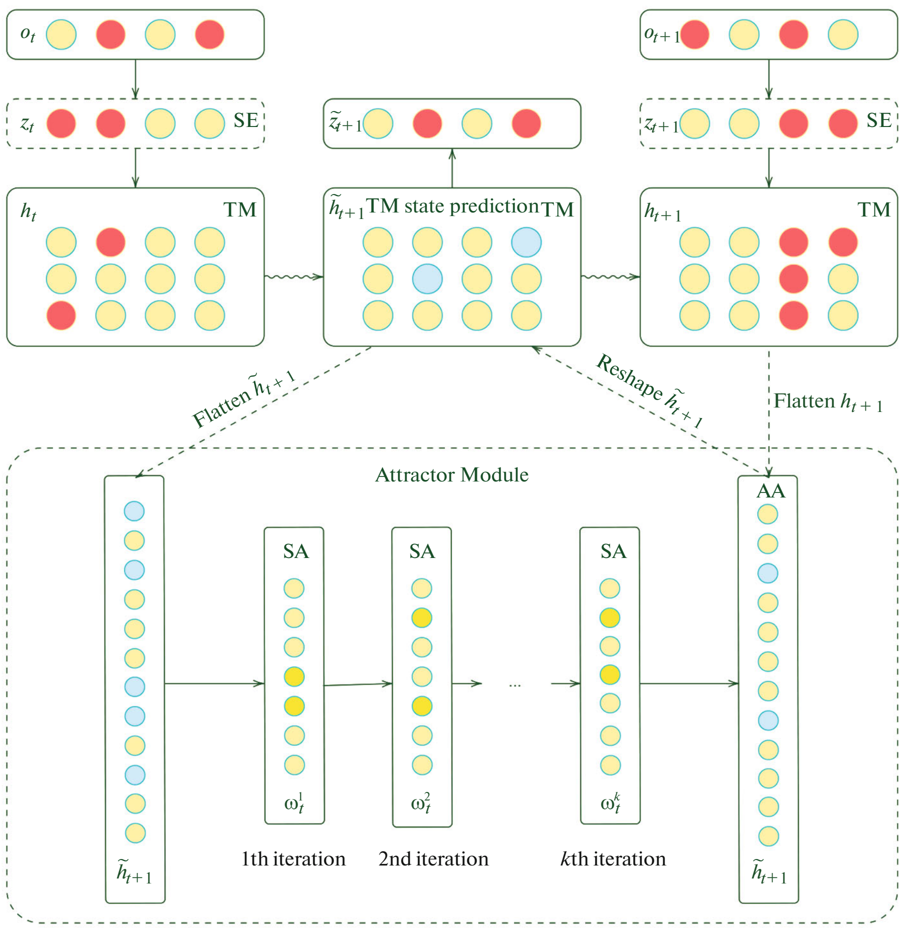


Fig. 1. Sequence memory model with attractor module. An input SDR o_t is encoded with spatial encoder (SE) to a latent vector z_t. It is forwarded to the temporal memory (TM), which updates its hidden state h_t and predicts the next-time state $\tilde{h}_{t+1}$. Attractor module corrects this prediction. First, flatten SDR $\tilde{h}_{t+1}$ is passed to the recurrent spatial attractor (SA) for k recurrent iterations. The result w_t^k is then passed to the associative attractor (AA) to form the corrected prediction $\tilde{h}_{t+1}$. It is returned back and overwrites TM prediction state. On the next timestep, the mismatch between the feedforward z_{t+1} and the TM prediction $\tilde{h}_{t+1}$ forms a feedback signal for TM and AA learning.

3.1.5. Associative attractor. The associative attractor (AA) is an associative memory, which is implemented with the first-order Temporal Memory algorithm. AA learns the inverse mapping from the SA representations space back to the TM latent state space—it learns to predict the TM's next hidden state, same as TM itself, but given the SA output SDR w_t^k as a feedforward input. The AA prediction is used to correct the TM original prediction $\tilde{h}_{t+1}$ by overwriting it. The AA learns with the prediction error signal from the TM.

It is worth noticing that the resulting attractor module has an autoencoder-like architecture with the SA playing the role of an encoder and AA playing a role of a decoder. It is also worth pointing out that the SA in the attractor block is optional, that is k could be set to zero. For both cases $k = 0$ and $k > 0$, we expect the associative attractor to also have properties of an attractor network (hence the name) because due to the lag between TM and AA learning (note that the attractor module learns over what TM has already learned), it will act as an additional regularizing filter. For the AA's segments we also set receptive fields larger than those of the TM to reinforce context consolidation effects.

Previous work [15] showed that the attractor properties of the attractor module are mostly due to the AA block alone, while the include of the SA does not bring any visible benefits. In this work we provide more thorough experimental analysis of the SA attractor properties.

In the process of testing the proposed model, we also end up with an alternative version of the attractor module (it is marked with `|W` on figures), in which it is applied to the current TM's winner cells instead of the predictions. Another difference is that the corrected prediction does not overwrite the existing TM's prediction, but is added to it with the union operation. It means that the attractor module is regularized itself and learns only the strongest relations between two timesteps. The idea behind this version becomes more clear, given the results of the main attractor module and, therefore, we discuss it in Section 4.1.

4. EXPERIMENTS

In this work we set two separate experiments. With the first we examine the ability of the model to memorize sequences and also extract and reuse abstractions, that is repeating subsequences. In this experiment, besides using single TM we also use LSTM (400 hidden state size) as the second baseline model. With the other experiment we study attractor properties of the spatial attractor.

4.1. Sequential Data Memorization Task

For this experiment, we used three different datasets: synthetic symbolic sequences, program code text, and event-based camera video. A synthetic dataset does not have a regular structure, and therefore only tests the memorization speed and memory capacity. The program code, on the contrary, has a fairly clear structure due to the syntax of the language and contains a large number of repetitive abstractions: variables, conditions and loops, functions and classes. Therefore, with this dataset, we test the model ability highlight and reuse abstractions to a greater extent. We chose programming language over natural language because of the more pronounced structure. The DVS-camera video dataset was chosen as a balanced option, in which regularities exist, but are noticeably less pronounced. Moreover, it shows practical applicability of the model to real data of this type.

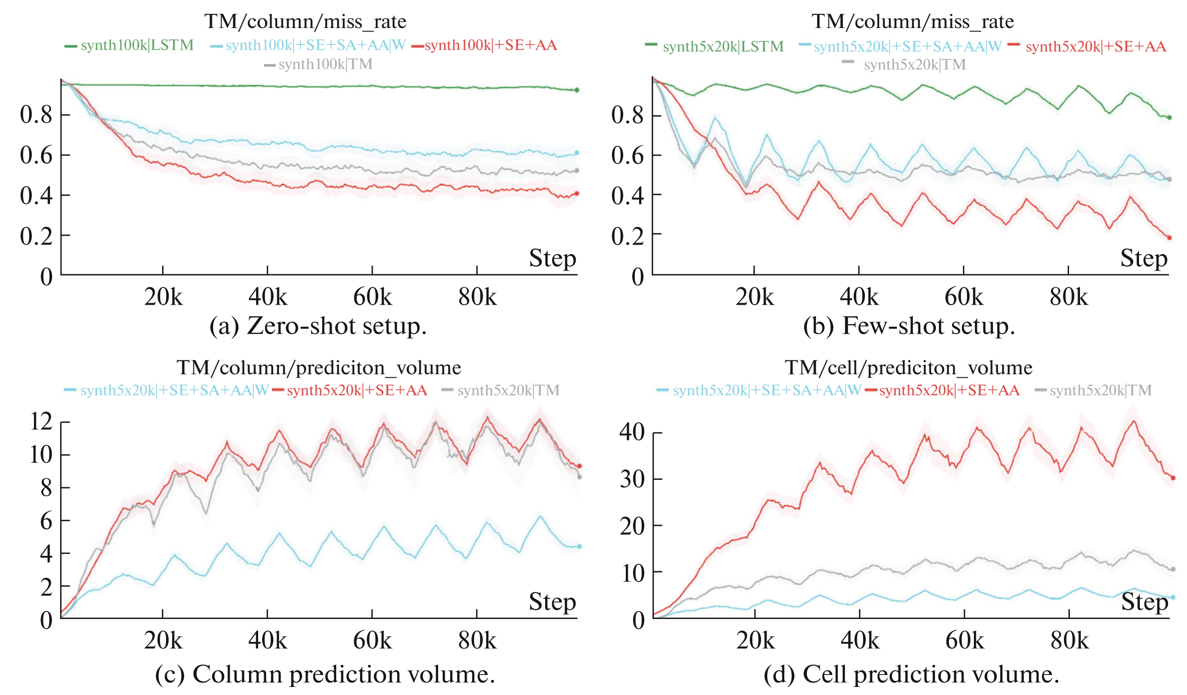
Every run is split into 50 epochs, and within an epoch the model observes 10 sequences with 200 SDRs each. The model was evaluated in two regimes: few-shot and zero-shot. The former assumes the same set of sequences is observed for 5 consecutive epochs, while for the latter each epoch the new set of sequences is sampled from the dataset. Regarding the model's ability generalize and reuse learned representations, these setups represent two difficulty levels.

The synthetic dataset consists of randomly generated symbolic sequences of length 200 from an alphabet of size 200 with controlled varying degrees of elementwise similarity between sequences. For the program code dataset we used the codebase of the Python implementation of our model, which is available online [16]. For both symbolic datasets, sequence elements are encoded into SDRs by mapping each symbol to a random SDR with 3% sparsity. DVS dataset contains outdoor walking scenes recorded with a DVS camera [17]. We preprocess its data to transform a stream of camera events to an SDR dataset. For this, we aggregate events into 0.5 ms frames, downsample resulted images from (240, 180) to (40, 30), average out polarity for image pixels and finally ternarize it (two polarity signs and off-state). Then each image is split into two binary images for both polarity signs, flattened and concatenated to a single binary SDR vector. For the DVS dataset, we use an additional SE before passing data to the model in order to normalize input sparsity level.

To evaluate the memory performance we use the model prediction miss rate—the mismatch between two SDRs representing TM prediction about the hidden state $\tilde{h}$ and the actual hidden state h: $\mathrm{MissRate}(h, \tilde{h}) = 1 - \frac{|h \cap \tilde{h}|}{|h|}$. To estimate the efficiency of the memory we also introduce column (and cell) prediction volume metric, which is the total number of predicted TM minicolumns (cells) divided by the number of active minicolumns. For minicolumns it estimates the number of possible observation outcomes TM expects, while for cells it roughly estimates the number of possible context outcomes.

Results in Figs. 2, 3, which are averaged over 10 random seeds and smoothed with the running average window of size 10, show that the attractor module significantly improves the memory performance on all datasets.

The memory performance for the zero-shot setup on the synthetic dataset (Fig. 2a) has a high plateau for all models because not much memorized information can be reused. Therefore predictably, LSTM performs poorly here, particularly in the zero-shot task. On the other hand, the improvement from the attractor module is pronounced for the few-shot setup (Fig. 2b). Figure 2c shows that the `+SE+AA` version of the attractor module predicts the similar amount of patterns compared to the baseline TM, but it is able to consolidate different contexts within the active columns and maintain them active for the successful predictions (Fig. 2d). Interestingly, while the alternative version `+SE+SA+AA|W` shows slightly worse performance even than the baseline, its active context and predictions are significantly more effective.



(a) Zero-shot setup. (b) Few-shot setup. (c) Column prediction volume. (d) Cell prediction volume.

Fig. 2. Memorization task performance on random sequences. While LSTM performs poorly compared to the others, the attractor module (`+SE+AA`) performs better than the `TM` baseline because it consolidates and maintains active different contexts within active columns (c, d).

Results for the program code and DVS datasets confirm `+SE+AA` superior performance (Figs. 3a and 3b). Due to the specifics of these datasets, the ability to consolidate different contexts plays here a supportive role for the correct pattern prediction—continuations for the subsequences, which were repeatedly seen in the contexts that differ from the current, are also predicted (Figs. 3c and 3d). However, this ability has a downside too. The attractor module learns associations excessively, which results to the much lower computational performance and lower prediction power (i.e. precision). In extreme case, the AA is able to fuse and consolidate contexts (and to some extent, patterns) entirely. As both TM and AA have a parameter responsible for the inhibition of the incorrectly predicted segments, we used increased values to balance the strength of the consolidation. Despite the fact that LSTM performs noticeably better on text data than on a synthetic dataset due to more structured nature of sequences, which is favorable for generalization, the performance still lags far behind the proposed model.

4.2. Attractor Properties of Spatial Attractor

It is known that the Spatial Pooler algorithm, when forming distributed representations, on the one hand, seeks to separate the input patterns, on the other hand, to preserve a certain portion of their similarity. This raises the question of whether it is possible to increase the proportion of intersection for similar patterns if SP is recurrently applied to the same pattern. This idea underlies the SA algorithm, which we investigate in this section. First, the original pattern is encoded to SDR with one SP (SE). Then the resulting representation is fed to the second SP (SA). The input and output representations of the SA have the same size and spacing. The output from the attractor is again transferred to its own input for several iterations.

In this experiment we use handwritten MNIST digits dataset. Since images corresponding to the same digit have many similarities, we assumed that SA should monotonically bring together the patterns of the same class in SDR space, improving the possibility of their classification. A number of studies on the properties of SP algorithm have shown that the use of SDR really improves the quality of classification compared to the classification on the original images. However, the recurrent application of SP to a SDR, like in our SA algorithm, has not yet been studied.

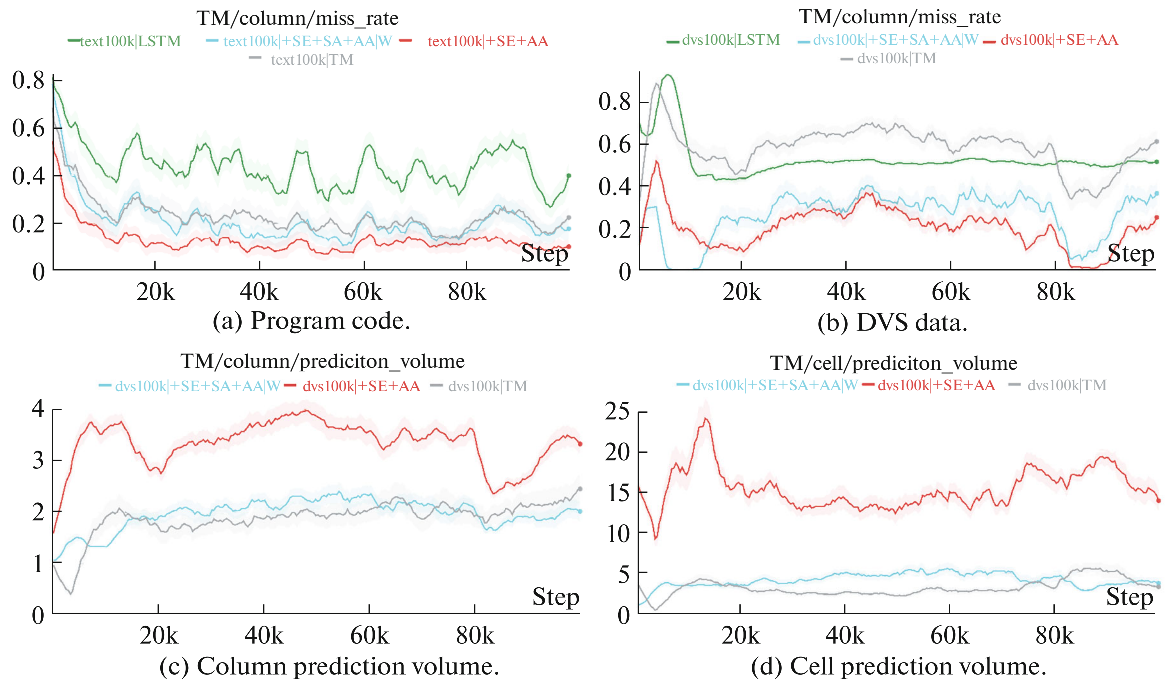


Fig. 3. Zero-shot memorization task performance on text and DVS datasets. The attractor module (`+SE+AA` and `+SE+SA+AA|W`) significantly increases the performance of the temporal memory (`TM`). `+SE+AA` memorizes more and keeps larger context active, while `+SE+SA+AA|W` represents context more effectively, leading to the better prediction precision (c, d).

As the main metric of the attractor properties of SA, the average relative proximity of patterns of one class compared to the proximity of patterns of other classes is considered, which is calculated by the formula:

$$\mathrm{rsim}_i = \frac{1}{n}\sum_j \frac{\mathrm{sim}_{ij}}{\mathrm{sim}_{ii}}, \tag{1}$$

where sim_{ij} is the average proximity of SDR from class i with respect to class j and n is the total number of classes.

The proximity of distributed representations is defined as follows:

$$\mathrm{sim}(\mathrm{SDR}_1, \mathrm{SDR}_2) = \frac{|\mathrm{a}(\mathrm{SDR}_1) \cap \mathrm{a}(\mathrm{SDR}_2)|}{|\mathrm{a}(\mathrm{SDR}_2)|}, \tag{2}$$

where $\mathrm{a}(\mathrm{SDR}_i)$ is the set of active bits of the distributed representation SDR_i, $|\cdot|$ is the cardinality of the set.

Figure 4a shows how the average rsim over all classes depends on the number of SP iterations. Zero iteration corresponds to the proximity of the original binarized images. The first iteration step corresponds to the exit from the SE, and the subsequent steps correspond to the exit from the SA. As can be seen from the plot, the *rsim* metric falls only after SE, but then gradually grows. This means that only the use of SE improves the clustering, and SA worsens it. However, this negative effect is less pronounced if the SA is trained on its own output. With the help of self-organizing maps, images of clusters were also obtained (Fig. 4b), from which it is clear that only the first iteration gives a visible improvement in the separation of patterns corresponding to different classes, and further iterations lead to mixing of classes.

A characteristic heat map was also constructed for the system under consideration, which, regardless of the class, shows the form of mapping from the proximity space of a pair of input patterns to the proximity space of a pair of final ones. Since in the MNIST sample the proximity of image pairs is unevenly represented, random patterns with different proximity were artificially generated. As can be seen from Fig. 5, as iterations increase, the patterns are separated more strongly, even those that were initially as similar as possible.

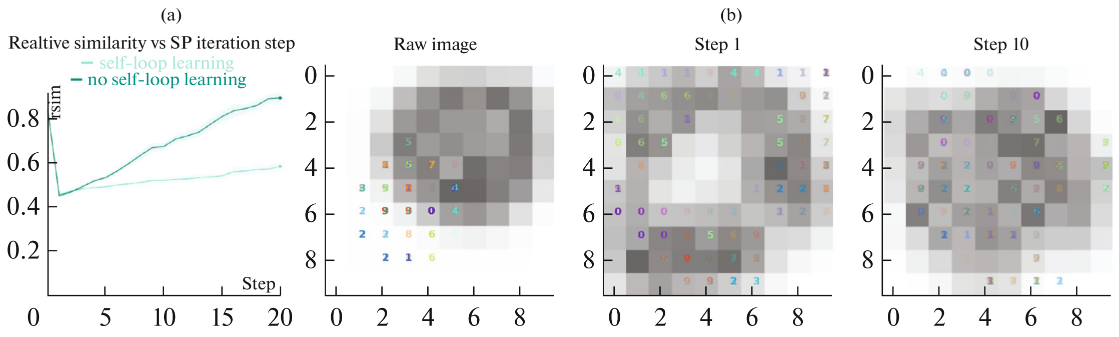


Fig. 4. (a) The average rsim over all classes depending on the iteration of the application of SP. Zero iteration corresponds to the similarity of the original binarized images, the first—to the exit from SE, and the subsequent iterations—to the exit from SA (b) Distribution of classes by neurons of a self-organizing map. The neuron indices are plotted along the axes. The cell color means the average Euclidean distance of the neuron's weights to the weights of its nearest neighbors. The clearer and richer the digit, the more often this digit activates this neuron. After the first iteration more neurons are involved but at the same time class separability is improved, however further iterations net to reversed effects as digits become blurred.

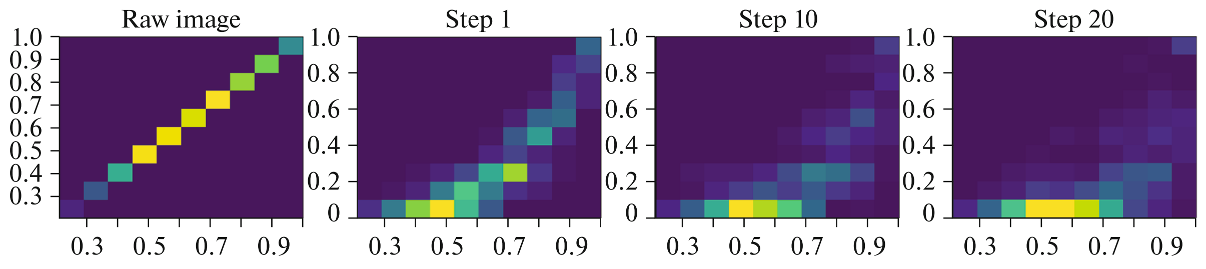


Fig. 5. Heatmaps represent mappings from the similarity space of pairs of original images (x-axis) to the similarity space of distributed representations (y-axis) for a different number of SP processing iterations. The leftmost heatmap shows pairwise class similarity in the dataset. With iterations, the similarity decreases compared to the similarity in the original space meaning that even very close patterns are separated.

The results of the experiments showed that the SP algorithm has poor attractor properties on its own and, on the contrary, tends to separate patterns. As can be seen from the visualization of self-organizing maps trained on distributed representations, such a separation can be useful for classification. However, with an increase in iterations, information about the proximity of the original images is lost in the output patterns.

5. CONCLUSION AND DISCUSSION

In this work, we proposed a biologically inspired memory model that is able to effectively memorize sequential data with the regular structure. The proposed model consists of several optional components—the spatial encoder module and the attractor module consisting of spatial and associative attractors—that support a sequence memory based on a temporal memory algorithm.

We tested the model on different datasets—random, program code and DVS camera video—to show how each component contributes to the model performance and to compare it with the LSTM baseline. The attractor module significantly changes the learned sequence memory hidden state representations by making them reflect the context similarity. This brings the sequence memory an ability to consolidate similar contexts, which improves the memory performance on all datasets in terms of the miss rate metrics. Compared to LSTM results, the proposed model shows significantly better zero-shot performance on any of the tested datasets via combination of pure TM memorization and the attractor module induced context consolidation.

Preliminary analysis [15] showed that the context consolidation effect of the attractor module is due to the associative attractor alone, while the spatial attractor brings neglible benefits to the model. In this work we separately studied the SA attractor properties and showed that it tends to distinct even close patterns.

Nevertheless, we still believe that the spatial attractor itself has potential, for example as a short-term self-attention module that is analogous to Fast Weights networks [18]. We did not study attentive mechanisms of the spatial attractor in this work and plan to further investigate it in future works.

We also supplied the memory performance results with the prediction volume analysis to show that better memorization comes with a price in terms of the computational performance and the prediction precision. Also, this analysis indicates that the large part of the context consolidation ability in the `+SE+AA` version is achieved via maintaining multiple contexts active on each timestep. We believe that ideally such feature should play lesser role, and the consolidation should take place mostly on the context representation level itself. That is, the memory should allocate repeated subsequences more strictly, and thus in the current context representation, a larger share should be occupied by the representation of the current short subsequence than by the more general context of the entire sequence. For this particular problem, we believe that the results of the attractor module alternative version `+SE+SA+AA|W` are of special importance. On the datasets with regular structure this version shows promising results and properties—while having mostly better than the baseline TM memory performance it has much more effective prediction volume, which we explain with the better consolidation properties. We leave the thorough analysis of this version for the future work.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

OPEN ACCESS

REFERENCES

1. Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T., Mastering Diverse Domains through World Models. arXiv:2301.04104 [cs, stat]. http://arxiv.org/abs/2301.04104. Accesses January 2023.
2. Sorokin, A., Buzun, N., Pugachev, L., and Burtsev, M., Explain my surprise: Learning efficient long-term memory by predicting uncertain outcomes, *Adv. Neural Inf. Process. Syst.,* 2022, vol. 35, pp. 36875–36888.
3. Rodkin, I., Kuderov, P., and Panov, A.I., Stability and similarity detection for the biologically inspired temporal pooler algorithms, *Procedia Comput. Sci.,* 2022, vol. 213, pp. 570–579.
4. Dzhivelikian, E., Kuderov, P., and Panov, A.I., Learning hidden Markov model of stochastic environment with bio-inspired probabilistic temporal memory, in *Procedia Computer Science,* 2023.
5. Hochreiter, S. and Schmidhuber, J., Long short-term memory, *Neural Comput.,* 1997, vol. 9, pp. 1735–1780.
6. Hawkins, J., Ahmad, S., and Cui, Y., A theory of how columns in the neocortex enable learning the structure of the World, *Front. Neural Circuits,* 2017, vol. 11, p. 81. ISSN: 1662-5110.
7. Dzhivelikian, E., Latyshev, A., Kuderov, P., and Panov, A.I., Hierarchical intrinsically motivated agent planning behavior with dreaming in grid environments, *Brain Inform.,* 2022, vol. 9, p. 8. ISSN: 2198-4026.
8. Mnatzaganian, J., Fokoué, E., Kudithipudi, D., A mathematical formalization of hierarchical temporal memory's spatial pooler, *Front. Rob. AI,* 2017, vol. 3. ISSN: 2296-9144. https://www.frontiersin.org/articles/10.3389/frobt.2016.00081.
9. Oster, M., Douglas, R., and Liu, S.-C., Computation with spikes in a winner-take-all network, *Neural Comput.,* 2009, vol. 21, pp. 2437–2465.
10. Graham, D. and Field, D., *Sparse coding in the neocortex*, *Evol. Nerv. Syst.,* 2007, vol. 3.
11. Skrynnik, A., Petrov, A., and Panov, A.I., *Hierarchical Temporal Memory Implementation with Explicit States Extraction in Biologically Inspired Cognitive Architectures (BICA) for Young Scientists, Advances in Intelligent Systems and Computing,* Samsonovich, A.V., Klimov, V.V., and Rybina, G.V., Eds., Springer, 2016, vol. 449, pp. 219–225. http://link.springer.com/10.1007/978-3-319-32554-5_28.
12. Stuart, G.J. and Spruston, N., Dendritic integration: 60 years of progress, *Nat. Neurosci.,* 2015, vol. 18, no. 12, pp. 1713–1721. ISSN: 1546-1726.

13. Smith, S.L., Smith, I.T., Branco, T., and Häusser, M., Dendritic spikes enhance stimulus selectivity in cortical neurons in vivo, *Nature,* 2013, vol. 503, pp. 115–120. ISSN: 1476-4687.

14. Staiger, J.F. and Petersen, C.C.H., Neuronal circuits in barrel cortex for whisker sensory perception, *Physiol. Rev.,* 2021, vol. 101, pp. 353–415. ISSN: 0031-9333.

15. Kuderov, P., Dzhivelikian, E., and Panov, A.I., Stabilize sequential data representation via attraction module, in *Lecture Notes in Computer Science*, 2023.

16. Kuderov, P., Dzhivelikyan, E., Latyshev, A., and Panov, A.I., *AIRI-Institute/him-agent: Hierarchical Intrinsically Motivated Agent Planning Behavior with Dreaming in Grid Environments,* version v3.2-hima-airi, 2022. https://doi.org/10.5281/zenodo.7133430

17. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., and Scaramuzza, D., The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM, *Int. J. Rob. Res.,* 2017, vol. 36, pp. 142–149.

18. Ba, J., Hinton, G.E., Mnih, V., Leibo, J.Z., and Ionescu, C., *Using Fast Weights to Attend to the Recent Past in Advances in Neural Information Processing Systems,* Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., Curran Associates Inc., 2016, vol. 29. https://proceedings.neurips.cc/paper/2016/file/9f44e956e3a2b7b5598c625fcc802c36-Paper.pdf.