Associative Memories With Synaptic Delays

Janusz A. Starzyk, *Senior Member, IEEE*, Łukasz Maciura, and Adrian Horzyk, *Senior Member, IEEE*

***Abstract*— In this paper, we introduce a new concept of associative memories in which synaptic connections of the self-organizing neural network learn time delays between input sequence elements. Synaptic connections represent both the synaptic weights and expected delays between the network inputs. This property of synaptic connections facilitates recognition of time sequences and provides context-based associations between sequence elements. Characteristics of time delays are learned and are updated each time an input sequence is presented. There are no separate learning and testing modes typically used in other neural networks, as the network starts to predict the next input element as soon as there is no expected input signal. The network generates output signals useful for associative recall and prediction. These output signals depend on the presented input context and the knowledge stored in the graph. Such a mode of operation is preferred for the organization of episodic memories used to store the observed episodes and to recall them if a sufficient context is provided. The associative sequential recall is useful for the operation of working memory in a cognitive agent. Test results demonstrate that the network correctly recognizes the input sequences with variable delays and that it is more efficient than other recently developed sequential memory networks based on associative neurons.**

***Index Terms*— Associative knowledge graphs, associative memories, synaptic delays, synaptic efficacy.**

I. Introduction

MEMORY plays an important role in cognitive systems, providing it with the knowledge about its environment and how to deal with it. Its structure self-organizes as a result of the past observations, actions, and their consequences [1]. The learning process includes changes in the long-term memory (LTM) cells and the synaptic connections between neurons. Associations between neurons reflect the context for the learning and representation building process [2]. However, complex patterns may require storage and associations of time domain patterns in declarative long-term memories. Time is critical for representing various temporal processes in the memory like scene recognition, learning of sequences of events, reasoning, planning, activity monitoring, and goal-driven learning. Many computational algorithms and organizational structures of long-term memories were designed and analyzed over the years to prove that they can satisfy requirements specified for such memories. Time delay neural networks [3] store a temporal sequence in static multi-layer feedforward networks. The recurrent neural network (RNN) is another way for sequence learning and prediction of dynamic responses [4]. It uses feedback links and memory of recent states. Its training is based on the back propagation through time method [5]. Sun and Giles [6] reviewed a spectrum of characteristics, problems, and challenges for sequence learning from recognition and prediction to sequential decision making. Wang and Arbib proposed a temporal sequence learning model based on two types of neurons [7]. The first type is a dual neuron, which stores a decaying signal value for a short period of time. The second type of neuron is a sequence-detecting neuron, which fires in response to the previously learned sequence of patterns. The developed model can reliably recall sequences that share similar patterns. A model capable of learning complex temporal patterns by self-organization was proposed in [8]. This memory can anticipate and regenerate the next component in a sequence comparing it with new input. A mismatch between the anticipated and actual input triggers learning. Wang used static associative neural networks with delayed feedback connections for learning and predicting spatiotemporal sequences [9]. In the learning stage, the input sequence is presented to the primary input channel, while the corresponding expected output sequence is simultaneously presented to the output channel. In [10], a dual-weight neural network (DNN) scheme was developed for fast learning, recognition, and reproduction of temporal sequences. In a DNN, each neuron is linked to other neurons by long-term excitatory connections and short-term inhibitory connections. Fast learning was accomplished using two-pass training of the temporal distance between two arbitrary pattern occurrences. A sequential extreme learning machine model was developed for fast sequential learning with or without chunking [11]. More recently a neural network structure for spatiotemporal learning and recognition inspired by the LTM model of the human cortex was proposed [12]. The developed LTM structure is able to process real-valued and multidimensional sequences. In a similar effort, a fast neural network adaptation with associative pulsing neurons (APNs) [13] was proposed. It uses a new type of APN developed in [14]. The associative neurons that are referred to in this paper have time delays, make easy associations, and are simpler than spiking neurons [15].

The main contribution of this paper is to propose a memory model in which associative connections carry information

Manuscript received March 16, 2018; revised August 30, 2018 and March 15, 2019; accepted May 30, 2019. This work was supported by the National Science Centre, Poland, under Grant DEC-2016/21/B/ST7/02220 and Grant AGH 11.11.120.612. *(Corresponding author: Adrian Horzyk.)*

J. A. Starzyk is with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701 USA.

Ł. Maciura is with the Department of Applied Information Systems, University of Information Technology and Management, 35-225 Rzeszow, Poland.

A. Horzyk is with the Department of Biocybernetics and Biomedical Engineering, AGH University of Science and Technology, 30-059 Krakow, Poland.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2921143

about expected synaptic delays for context-based sequence recognition, prediction, and memory. We develop this model and analyze its properties. We demonstrate how such a model can store and retrieve associated sequential data. Finally, we compare the performance of the proposed new type of associative memories to the active neuro-associative knowledge graphs (ANAKG) [13], long short-term memory (LSTM) [16] and hierarchical temporal memory (HTM) [17]. We proved that the proposed associative memory with synaptic delays is more accurate and efficient than these methods.

In Section II, we describe various types of long-term declarative memories and their properties. This is followed by a description of the synaptic gates that can detect input novelty and learn synaptic delays of the stored input sequences, as presented in Section III. Section IV gives an overview of the organization of synaptic delay associative knowledge graphs (SDAKG) and its implementation algorithm responsible for self-organization of the memory structure including the associative learning process, signal predictions, and control flow. Section V presents the results of the conducted experiments, and Section VI contains conclusions.

II. Long-Term Declarative Memories

Long-term declarative memories are divided into episodic and semantic memories, and they require different structural organization, storage, and recognition properties. While the episodic memory stores personal memories and requires storage and recognition of sequences, the semantic memory stores a general knowledge and relies more on associations between its elements. There are many artificial neural network models used to simulate semantic and episodic memories. Associative networks are content addressable and can retrieve stored data based on only a part of what was stored [18]. They are resistant to noise and can detect missing data and sensory failures [19]. RNNs developed to store and recognize sequences use backpropagation through time and error signal to adjust interconnection weights between neurons. As Hochreiter and Schmidhuber [16] pointed out, this error signal vanishes after several steps. As a result, learning and prediction may fail for problems with long-time domain dependencies [16]. To remedy this problem, they proposed a long short-term memory (LSTM) structure for efficient storage of sequences with bigger and varying time-scales than classical RNNs. Mikolov *et al*. [20] proposed an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents. Similar to RNNs, this algorithm is trained to predict words in a document given an input context. Weston *et al*. [21] describe a new class of learning models called memory networks. Memory networks combine an input content with the dynamic knowledge base stored in the LTM to predict the output.

A hierarchical organization of memory is important for sequence learning. The hierarchical sequence learning in a sensory-motor system using RNNs was investigated by Tani and Nolfi [22]. Based on their model, complex sequential behaviors can be learned as was demonstrated in a robot arm simulation [23]. The system was capable of dynamical self-organization across multiple levels of learning and prediction.

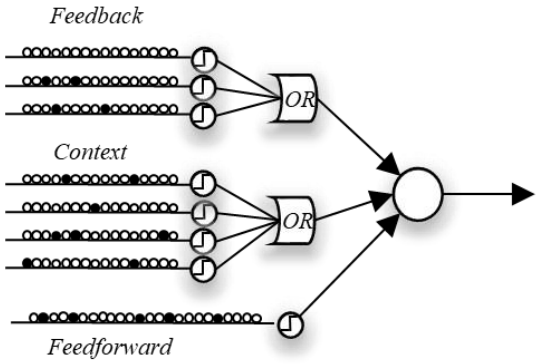

Fig. 1. Organization of the artificial neuron in the hierarchical memory of the hierarchy HTM. The HTM neuron learns by changing weights of synaptic connections—new synapses are attached as postsynaptic neurons are activated.

George and Hawkins [24] presented a hierarchical structure for temporal sequence learning aimed at invariant pattern recognition. A columnar organization of the associative memory was proposed by Hawkins *et al*. [17] where they introduced cortical learning algorithms in which minicolumns were used to store sequential information in structures known as HTM. Since then, the HTMs were further developed, their properties were analyzed and tested. Cui *et al*. [25] show that HTMs can continuously learn a large number of temporal sequences using an unsupervised learning neural network model. HTM was shown to have similar accuracy as other state-of-the-art sequence learning algorithms like echo state networks [26] or LSTM [16]. However, they also show some drawbacks like larger sensitivity to the temporal noise than the LSTM [25].

The HTM uses artificial neurons that model biological neurons. Since biological neurons are very complex, only some of their functionality is modeled in artificial neurons. The organization of the artificial neurons used in the HTM is shown in Fig. 1. A characteristic feature of these neurons not used in most of the artificial neural networks is that they use the equivalent of proximal and distal dendrites. In biological neurons, proximal dendrites are close to the cell body, while distal dendrites are further away. The proximal dendrites affect a neuron in a linear additive way—the more synapses in these dendrites are active, the higher the activation of the postsynaptic neuron. Each HTM neuron has single-proximal dendrites with a number of synapses whose activation is summed up at the neuron cell body. The distal dendrites are connected to other dendrites, and they act as thresholded coincidence detection—they affect the postsynaptic neuron activation only when sufficiently large numbers of their synapses are active at the same time. If any of the distal dendrites is activated, this activates the cell body.

An efficient way to build spatiotemporal sequential memories for long-term storage of input sequences was developed by Horzyk [14] in the form of ANAKG. The ANAKG network is constructed from APNs that incorporate temporal spiking in artificial neurons with associative neuron modeling and plasticity. These neurons learn associations between symbols or objects defined as time-spread combinations of activated inputs. These neurons are time-dependent and can

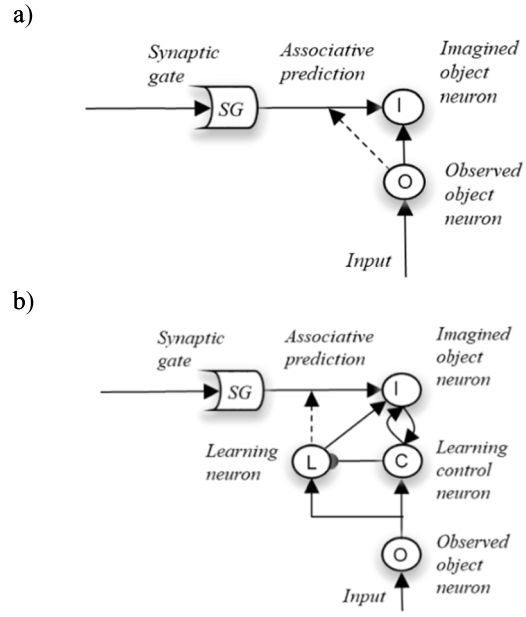

Fig. 2. Neuronal structures in SDAKG networks. (a) Each time the input signal to the observed object neuron is activated, learning takes place, changing the weight of the interconnection link between the synaptic gate and the object neuron. (b) If the observed object neuron is active without associative prediction, then the learning control neuron is not activated, and the input signal activates the learning neuron. When the imagined object neuron is activated without activation of the input to this object neuron no learning takes place.

be charged, relaxed, and refracted after spikes. However, they are simpler than spiking neurons which try to reproduce biochemical processes, while APNs only try to reproduce functional aspects of real neurons.

The most important aspect of the adaptation and learning in the APN is based on the efficacy of their synaptic connections, computed accordingly to the time that elapsed between presynaptic and postsynaptic activities of the two neurons that were activated in close temporal succession. The longer this period is, the smaller the synaptic efficacy. The synaptic efficacy significantly simplifies the adaptation process and provides an efficient tool for organizing the network structure [28]. In this paper, we apply the concept of the synaptic efficacy in a new way, to obtain crisp separation properties of the stored input sequences, as discussed in Section III.

III. Synaptic Gates and Synaptic Delays

A. Synaptic Gates

Our novel approach to learning of temporal sequences is to use explicit time delay information and use of synaptic gates in the learning process of the associative neurons. Associative learning is organized in the networks that combine imagined object representation neurons and synaptic gates supported by an observed object and learning neurons, as indicated in Fig. 2. A network that combines many such structures is called SDAKG. In SDAKG network, synaptic gates and imagined object neurons represent different aspects of temporal learning and associations. Synaptic gates carry the history of temporal activations of the observed object neurons, while imagined object neurons represent network predictions in replaying the temporal sequence. Synaptic gates respond to the time difference between their inputs and activate synaptic connections to other synaptic gates and imagined object neurons. They play a critical role for storage and recall of temporal sequences and making associations in the semantic memory. Imagined neurons are important not only to recall the stored sequence, but their activations are critical for anticipation and planning by autonomous agents. A synaptic gate is activated the most if all the inputs to the synaptic gate are synchronized in time. An output from the synaptic gate is distributed with a specified delay to several imagined object neurons and their corresponding synaptic gates using trainable connections with variable weights. An imagined object neuron may have several inputs, each from the different synaptic gate and each with a different weight and time delay. In addition, each observed object neuron may be connected to several synaptic gates, and the activation of any of its synaptic gates will activate the imagined object neuron.

An observed object and learning neurons associated with an imagined neuron play supporting roles for creating the structure of an SDAKG, for novelty detection and learning. An observed object neuron recognizes the input pattern in a similar way as associative neurons discussed in [14].

An associative prediction signal is obtained from the activation of a synaptic gate multiplied by the weight of the associative link between the synaptic gate and the imagined object neuron. A simplified organization of the learning process in the SDAKG memory is presented in Fig. 2(a), where each time the input signal to the observed object neuron is activated, learning takes place changing the weight of the interconnection link between the synaptic gate and the imagined object neuron, and if such a link does not exist, it establishes a new one. Besides, the synaptic delay between the synaptic gate and the imagined object neuron is also modified. The dashed line in Fig. 2 indicates which synaptic connection weight will be changed after the observed object neuron is activated.

B. Novelty Detection

Familiarity with the sequence of temporal observations allows for novelty detection in the observed scene. The novelty detection is useful since it tells the system that observed the response from the environment was not represented in the semantic memory and may require learning. The machine learns new knowledge if it is useful for its operation. Since temporal sequence learning depends on the episodic memory, novelty is recorded first by the episodic memory, and it is transferred to the semantic memory only when the episode or its elements need to be learned. If the observed scene is either expected or unimportant to the agent, no new knowledge is registered in the semantic memory. Proposed in Fig. 2 one of the two configurations of the SDAKG network can be selected depending on how the novelty observations are treated.

For novelty detection in this memory system, we can use the organization shown in Fig. 2(b). In Fig. 2(b), if an imagined object neuron is activated through the associative prediction, it sends a signal to a learning control neuron. The learning control neuron C has a higher threshold and

requires both of its inputs to be activated—one from the imagined object neuron and one from the observed object neuron. Thus, if the observed object neuron receives an input signal sent at the time of the associative prediction, then the learning control neuron is activated and inhibits the learning neuron L. As in Fig. 2(a), the dashed line in Fig. 2(b) indicates which synaptic connection weight will be changed after the observed object neuron is activated. However, if the input signal for the observed object neuron is active without associative prediction, then the learning control neuron is not activated, and the input signal activates the learning neuron. The activated learning neuron initiates changes to the weight of the interconnection link between the synaptic gate and the object neuron, and if such link does not exist, it establishes one. When the imagined object neuron is activated without activation of the input to its observed object neuron, no learning takes place. This may happen during recall of the temporal sequence from the SDAKG memory.

The major difference between the two methods is that in Fig. 2(b), there was no change in weights between the synaptic gate and the imagined object neuron when the associative prediction link predicted the imagined object activation. Only in this structure, predicted activations cause no learning, which may be useful for the organization of episodic memory. No repetition of the input sequence is necessary for such networks, while in the structure, shown in Fig. 2(a), learning takes place each time an input sequence is observed so that the synaptic weights can be changed gradually. In both methods, recalling the sequences from memory do not change the existing SDAKG structure nor the interconnection weights.

C. Sequence Learning Through Synaptic Delays

Let us explain the idea of sequence learning through synaptic delays in SDAKG networks. The SDAKG is a self-organizing network, establishing synaptic gates and associations between object neurons as needed. In learning temporal sequences, we distinguish presynaptic and postsynaptic gates in a similar fashion as used in determining presynaptic and postsynaptic neurons. An input signal activates first a presynaptic gate and then an associated postsynaptic gate. In our approach, the SDAKG neural network contains explicit delay information in addition to the interconnection weights between associated synaptic gates and neurons. Each time an observed object neuron is activated by the input signal, the synaptic connection weight and the delay time between the predecessor synaptic gate and the successor gate related to this object neuron change. The delay information in the synaptic gates represents the level of similarity of the observed time domain sequence to the sequence represented by the synaptic gate.

During training, the synaptic gate learns the characteristic delays of the received signals. Since we want the synaptic gate to tolerate small time deviations of the received signals, we approximate unknown distribution of these delays. According to the central limit theorem, we may approximate this unknown distribution by computing its average value t_{12} and standard deviation σ_G. After training synaptic gate will use these values to compute its attenuation function as described next.

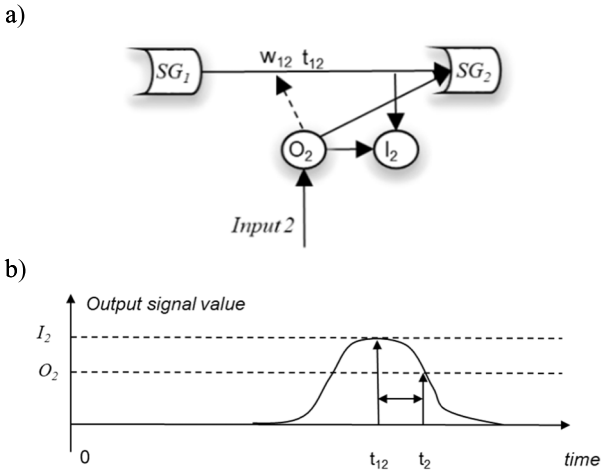

Fig. 3. (a) Propagation of signals between synaptic gates. The output signal O_2 of the synaptic gate SG_2 is attenuated depending on the temporal difference between the expected and the observed signals. (b) Sketch of the signal attenuation through the synaptic gate.

A synaptic gate is activated if its inputs are activated at the same time (or in proximity to each other). The gate uses the temporal difference to attenuate the strength of its input signals. The strength of the input signal will be maximum at the average delay time t_{12} observed during training and can gradually decay if the arrival time of the input signal is either higher or lower than the t_{12}. If the delay of the input signal received by the postsynaptic gate t_2 is different than t_{12}, the output of the postsynaptic gate is attenuated depending on $(t_2 - t_{12})$ using

$$O_2 = I_2 * f(t_2 - t_{12}) \tag{1}$$

where $f(t_2 - t_{12})$ is a synaptic gate attenuation function. In our simulation, the attenuation function is chosen as a Gaussian function with t_{12} means and standard deviation σ_G:

$$f(t_2 - t_{12}, \sigma_G) = e^{-\frac{(t_2 - t_{12})^2}{2\sigma_G^2}}. \tag{2}$$

Attenuation is large when the delay of the received input signal t_2 is significantly different than t_{12}.

To illustrate this, let us consider that an object neuron O_2 that activates the postsynaptic gate SG_2 in Fig. 3 was activated in time t_2. Fig. 3 shows the gate organization and the sketch of its attenuation function.

The input signal to a synaptic gate is either equal to the activation level of the presynaptic gate output O_1 multiplied by the synaptic weight w_{12}:

$$I_2 = O_1 * w_{12} \tag{3}$$

or it is computed using

$$I_2 = O_1 * w_{12} + 1. \tag{4}$$

In the first case, we only observe the attenuation of the input signal strength, so this case is suitable when we have no input, for instance in the case of prediction when the sequence is recalled from the memory or during testing mode when no input from object neuron is received at a specific gate.

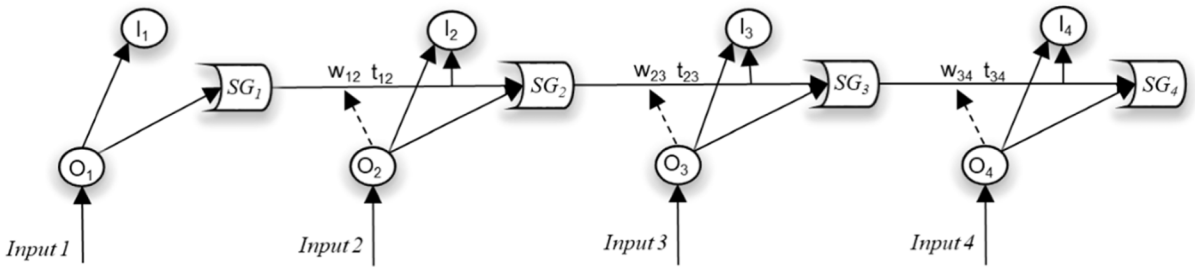

Fig. 4. Network structure obtained after observing the input sequence (Input 1, Input 2, Input 3, and Input 4).

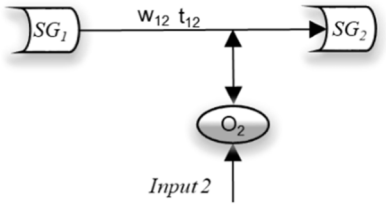

Fig. 5. Simplified propagation of signals. Imagined and object neurons are merged and represented by the oval shape. Bidirectional connection stands for two separate links. One is to the imagined object neuron and the second one from the observed object neuron to the postsynaptic gate. While the first link has weight w_{12} and delays t_{12}, the second one (send with no delay) has a weight equal to 1 with arrival time t_2 according to activation of the object neuron.

TABLE I
EXAMPLE OF SENTENCES USED TO BUILD A SIMPLE MEMORY

No.	Sentence
1.	I have a monkey.
2.	My monkey is very small.
3.	It is very lovely.
4.	It likes to sit on my head.
5.	It can jump very quickly.
6.	It is also very clever.
7.	It learns quickly.
8.	My monkey is lovely.
9.	I also have a small dog.

Since each synaptic gate carries the history of temporal activations of the observed object neurons, its activation is equivalent to the recognition of the input sequence that it represents. Thus $+1$ in (4) indicates that a new input was received and was combined with the propagated accumulated signal.

IV. SDAKG NETWORK ORGANIZATION

The SDAKG network is created gradually by observing various input sequences. In Fig. 4, we show the structure of the SDAKG network obtained after observing the input sequence (Input 1, Input 2, Input 3, and Input 4).

In general, a single synaptic gate can be linked to several postsynaptic gates, each time using the different synaptic connection with trainable weights and time delay. This is useful if the sequence represented by a synaptic gate can be continued in several different ways. Likewise, an imagined neuron can be stimulated by several different presynaptic gates as a given object can be included in several different sequences.

To simplify the graphical representation of connections between imagined object neurons, observed object neurons, and synaptic gates, we illustrate the propagation of signals between synaptic gates as shown in Fig. 5.

Using this simplified notation and the sequences of words, shown in Table I, we can obtain the structure of the SDAKG network that represents these sentences as shown in Fig. 6. In this structure, we can see that individual words can be connected to several synaptic gates, which indicates their use in different sentences. We also see some cases when a synaptic gate connects to several postsynaptic gates.

A. Synaptic Weights

To establish synaptic weights between a pair of presynaptic and postsynaptic gates, we use the so-called synaptic efficacy (5) computed accordingly to the time that elapsed between presynaptic and postsynaptic activities of synaptic gates if both gates were activated in close temporal succession. The synaptic efficacy is a measure of the influence of the synaptic stimulations on the postsynaptic neuron activity. The synaptic efficacy is also dependent on a frequency of the contribution of this synapse stimulation to the postsynaptic gate activity.

The synaptic efficacy significantly simplifies the association process in comparison to other models of sequential learning.

Let $S^s = [S^s_1, \ldots, S^s_k, \ldots, S^s_{k+r}, \ldots, S^s_{K_n}]$ be an input sequence from the observed sequence set $\mathbb{S} = \{S^1, \ldots, S^N\}$. The synaptic efficacy is computed for each of two connected synaptic gates G_m and G_r representing two partial sequences from the observed set in each sequence S^n that contains them. Gates G_m and G_r, are activated by activation of the corresponding object neurons N_m and N_r. The time differences between the activations of G_m and G_r gates affect the computation of various components of the sum (3). The final synaptic efficacy for this synapse considers all input sequences that contain time ordered succession of gates G_m and G_r. The synaptic connection between gates G_m and G_r has the weight w_{mr} and the average time delay $\hat{t}_{\mathrm{mr}}$

$$\delta_{G_m,G_r} = \sum_{\{(G_m,G_r)\in S^n\in\mathbb{S}\}} \left(\frac{1}{1+\frac{|t_r-\hat{t}_{\mathrm{mr}}|}{3.(\sigma_r+\frac{2*\hat{\delta}_{\mathrm{mr}}}{\sqrt{}}+\epsilon)}} \right)^{\gamma} \tag{5}$$

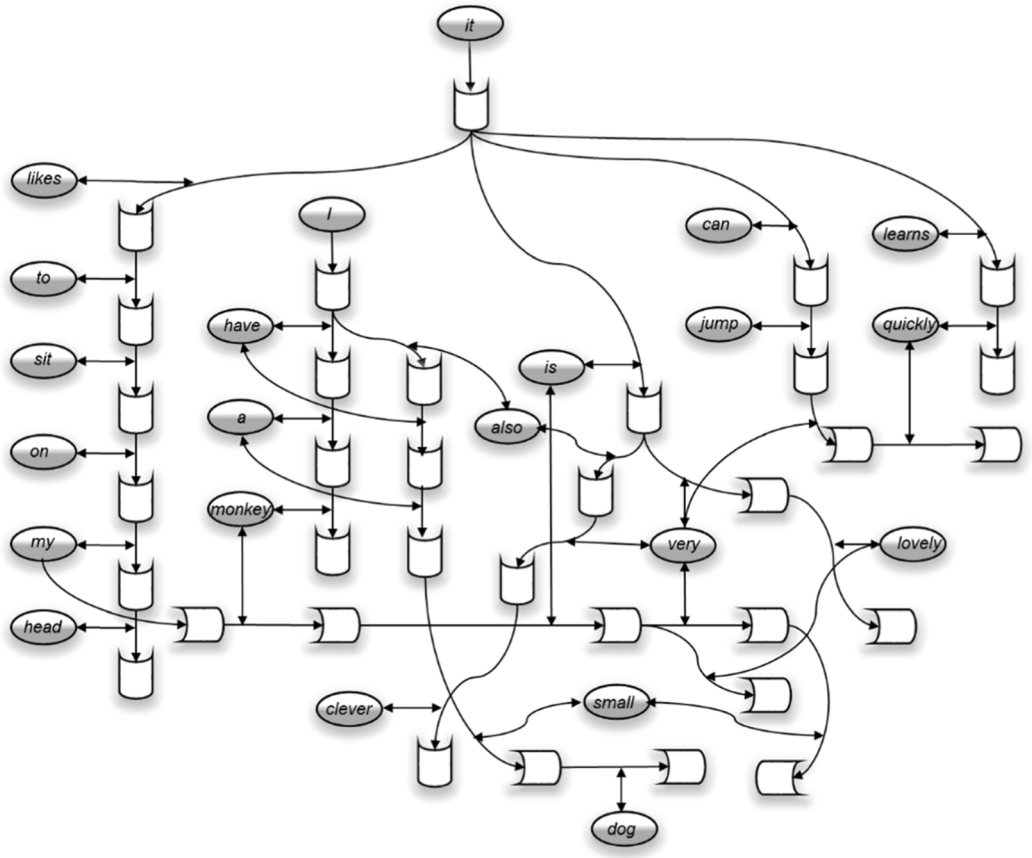

Fig. 6. Structure of the SDAKG network that represents the presented sentences.

where

- t_r is the period of time that lapsed between the stimulation of the synapse between gates G_m and activation of the postsynaptic gate G_r;
- $\hat{t}_{\mathrm{mr}}$ is the average period of the time delay between the activation of synaptic gates G_m and G_r;
- σ_r is the calculated standard deviation of the observed delay between the activations of the presynaptic and postsynaptic gates G_m and G_r, as expressed by (10);
- ϵ is the small positive number;
- n is the number of the prior activations of the postsynaptic gate G_r;
- γ is the context influence factor changing the influence of the previously activated synaptic gate on the postsynaptic gate (here equal to 4).

The synaptic efficacy is used to compute a synaptic permeability (also called connection weight). The synaptic permeability is computed after the activity of the presynaptic gate G_m by considering its influence on the postsynaptic gate activity. Synaptic weight w is computed using

$$w = \frac{\eta}{2\eta - \delta} \tag{6}$$

where η is the number of activations of the presynaptic gate and δ is the synaptic efficacy computed for this synapse.

The weight function satisfies the requirement that the weight approaches 1 when synaptic efficacy approaches the number of activations of the presynaptic gate and approaches 0.5 when synaptic efficacy is low. Synaptic efficacy increases with the number of activations. It approaches the number of activations when each time when the presynaptic gate is activated, the postsynaptic gate is activated as well.

The average period of time delay $\hat{t}_{\mathrm{mr}}$ can be updated dynamically, each time the postsynaptic gate is activated using

$$\hat{t}_{\mathrm{mr}} = \frac{\breve{t}_{\mathrm{mr}} * n + t_{\mathrm{mr}}}{n+1} \tag{7}$$

where $\breve{t}_{\mathrm{mr}}$ is the previous average value and t_{mr} is the observed delay between the activations of the presynaptic and postsynaptic gates G_m and G_r.

Similarly, the standard deviation can be updated using

$$\begin{aligned}\sigma_r &= \sqrt{\frac{n * \left(\breve{\sigma}_r^2 + \breve{t}_{\mathrm{mr}}^2\right) + t_{\mathrm{mr}}^2}{n+1} - \hat{t}_{\mathrm{mr}}^2} \\ &= \sqrt{\frac{n}{n+1} * \left(\breve{\sigma}_r^2 + \frac{(\breve{t}_{\mathrm{mr}} - t_{\mathrm{mr}})^2}{n+1}\right)}\end{aligned} \tag{8}$$

where $\breve{\sigma}_r$ is the previous value of the standard deviation.

Thus to maintain the information about the state of a synapse, we need to store only three values: n, $\breve{t}_{\mathrm{mr}}$, and $\breve{\sigma}_r$ related to this synapse. We set the initial values of all $\breve{t}_{\mathrm{mr}}$, and $\breve{\sigma}_r$ to 0.

When the input sequences are played at a different time scale, it is necessary to establish parallel synapses between a pair of presynaptic and postsynaptic gates with different link values and different delays. We decided to establish a new synapse when the following condition is met:

$$|t_{\mathrm{mr}} - \hat{t}_{\mathrm{mr}}| > 5 * \left(\sigma_r + \frac{2 * \hat{t}_{\mathrm{mr}}}{\sqrt{n+1}} + \epsilon\right) = t_{\mathrm{mr}}^{\max}. \quad (9)$$

Each newly established synapse will be equipped with its own counter of the number of activations to establish its weight.

B. Organization of SDAKG Algorithm

The implemented SDAKG algorithm contains several critical steps for associative learning, signal propagation, selection of a dominant signal, and control flow of the working SDAKG network as described in this section. The associative learning process responds to each input sequence and is responsible for organizing the associative connection structure of the SDAKG network as well as learning characteristics of synaptic delays.

The associative learning process (ALP) runs as follows.

1) If the input signal was predicted by a synaptic connection between presynaptic and postsynaptic gates then the ALP updates $\hat{t}_{\mathrm{mr}}$ and the related values of σ_r, σ_G, n, as well as the synaptic efficacy δ, and the weight w for this synaptic connection. The number of activations of a postsynaptic gate η is increased by 1.
2) If the input signal was not predicted, because there was no synapse from the presynaptic gate, a new postsynaptic gate is created, and its η is set to 1.
3) If the input signal was not predicted or the waiting time t_{mr} for this signal was such that $|t_{\mathrm{mr}} - \hat{t}_{\mathrm{mr}}| > t_{\mathrm{mr}}^{\max}$ a new synaptic connection is established with $\hat{t}_{\mathrm{mr}} = t_{\mathrm{mr}}$, $\sigma_r = 0$, $\sigma_G = 2 * \hat{t}_{\mathrm{mr}} + \epsilon$, $n = 1$, $\delta = 1$, and $w = 1$. In Steps 2 and 3 described here, the graph structure as in Fig. 6 is formed automatically by automatic creation of synaptic gates and synapses between gates according to new data sequences.
4) If the input signal was the first element of the sequence, then no prediction can be made. In this case, η—the number of activations of the synaptic gate that represents this input signal—is increased by 1. If no such gate exists a new gate is created and its η is set to 1.

Once activated by the input signals, the SDAKG network propagates signals through its gates. The SDAKG signal propagation algorithm is organized as follows.

1) Once a gate receives an input signal, it starts to calculate its delay related attenuation according to the Gaussian function with zero mean and standard deviation equal to

$$\sigma_G = \sigma_r + \frac{2 * \hat{t}_{\mathrm{mr}}}{\sqrt{n+1}} + \epsilon. \quad (10)$$

2) At the moment the input signal comes to a postsynaptic gate, the attenuation function (2) is computed, and the output of the postsynaptic gate is computed using (1).
3) If the delay between received signals is greater than $t_{\mathrm{mr}}^{\max}$ [determined by equation (9)], then a new synapse between the presynaptic and postsynaptic gates is created and its synaptic delay is set to the input delay.
4) Once an input signal is received, it initiates the associative learning process (ALP) and stops propagation of all signals sent from the gates that were in the prediction mode. Notice that this may happen much later than $t_{\mathrm{mr}}^{\max}$, so it cannot stop the signals already propagated.
5) If the input signal does not come (time greater than $t_{\mathrm{mr}}^{\max}$, or end of the sequence is announced), all the gate signals are propagated through the existing network structure using the winner-takes-all control mechanism (WTACM). These are the prediction signals that activate the imagined object neurons.

Prediction signals are useful for associative recall from SDAKG memory. They depend on the presented input context and the knowledge stored in the graph.

The WTACM that governs the prediction process is as follows.

1) The postsynaptic gate which presynaptic gate was last activated by the input signal initiates the WTACM after the minimum waiting time $t_{\min}$ determined from

$$t_{\min} = \left(\hat{t}_{\mathrm{mr}} + t_{\mathrm{mr}}^{\max}\right) \quad (11)$$

and starts the transition to the prediction mode.
2) The WTACM tree nodes are created by connecting the root node to its postsynaptic gates. This will generate new leaf nodes. This step is applied to all new leaf nodes.
3) Starting from the root node, the WTACM chooses one available node connected to the postsynaptic gate with the highest weight of the synapse from its presynaptic gate and sends a prediction signal to the postsynaptic gate and its object neuron. This node becomes the current node of the WTACM, and the pointer is set to this node.
4) Prediction signals are not attenuated but are multiplied by the corresponding synaptic weights.
5) If the pointer reaches one of the leaves of the tree or the propagated prediction signal strengths falls below the set threshold, then the available flag of this node is set to false.
6) After reaching the leaf node, the WTACM returns to the root node and can select another path of the tree. If this is impossible, then the WTACM stops working.
7) When a new input signal is obtained during the prediction process, then the prediction process stops, the event queue of the prediction mode is erased, the WTACM tree is saved, and the program switches to the learning mode.

Random inputs distort sequence recognition. If the input probability of individual symbols is known, we can make a minor adjustment for random activation of synaptic gates by subtracting estimated a randomly generated value O_{ri} from the output value of each synaptic gate

$$O_{ri} = \sum_{k=0}^{i} \binom{n}{k} p^k (1-p)^{n-k} \quad (12)$$

where p is the probability of activation of an object neuron, n is the length of the input sequence, and i indicates the level of the synaptic gate (the length of the sequence represented by the synaptic gate). The summation in (16) is over all object neurons k of the sequence represented by the synaptic gate.

C. Control Flow of SDAKG Network

To handle input from object neurons and switch between the learning mode and the prediction mode in the SDAKG network, an event-driven simulation control flow is applied. There are three event queues: one for data handling, one for the learning process and one for the prediction. While the network is in the learning mode, then the events from the current location of the learning event queue are received and handled. Respectively, while the network is in the prediction mode, then events from the current location of the prediction event queue are received and handled. When the time of the current element in data event queue is smaller than the time of the current element of the learning event queue and the current element of the prediction event queue, then this data element is handled, and the network switches to the learning mode.

The SDAKG network has the following four submodes in order to automatic switch between learning and prediction modes properly and one testing mode.

1) *Learning Mode:* The network switches to this mode whenever a new data comes from its object neurons. From this mode, the network cannot switch to the prediction mode. When any gate in the network is firing, then the network switches to mode 2.
2) *Conditional Learning Mode:* Only from this mode network can switch to the prediction mode. It occurs when any gate after waiting time $t_{\mathrm{mr}}^{\max}$ did not receive a signal from the object neuron (it received the only signal from another gate through its input synapse).
3) *Prediction Mode:* In this mode, the WTACM mechanism provides a working prediction associated with one path in the WTACM tree. If the WTACM mechanism reaches the tree leaf (one full path was played), then the network switches to mode 4.
4) *Conditional Prediction Mode:* This mode is used to generate a new path of the WTACM tree and return to mode 3. If there are no new paths (that were not played), then the WTACM mechanism stops its work.
5) *Testing Mode:* This mode is used to test recognition time sequences. Switching to this mode (and switching from this mode to another mode) is forced by an SDAKG user.

Submodes of the learning mode are necessary to prevent switching into the prediction mode when the network receives a new signal from its object neurons before the maximum waiting time $t_{\mathrm{mr}}^{\max}$ or to exit from the prediction mode when new data come.

D. Testing Mode

The testing mode is used for time sequences recognition tasks. The user of SDAKG neural graphs can switch the network to this mode after learning is completed. In this mode, all learning functionalities (i.e., construction of the graph, changing of synaptic weights) are blocked, so the memory structure does not change anymore, and the tests check if the input sequences are correctly recognized.

In the testing mode, the gate propagates a signal whenever a signal from its presynaptic gate is received and is greater than the threshold and simultaneously greater than the last output signal of this gate. If these conditions are not met, then the new signal is not propagated. The input and output signals in all gates are cleared before each recognition process call.

After propagation of a signal, the input signals are saved (until new ones replace them), so they can be reused if needed. For example, if a gate received a signal from its presynaptic gate and propagated its output signal to its postsynaptic gates, and next it received a signal from the object neuron, then both signals are used to compute and propagate a new output signal. If a gate received signals from its object neuron and the presynaptic gate, then (1) is used for output signal calculation. In other cases, the output is determined by the value of the received signal.

V. Results of Experiments

A. Testing of SDAKG Memories

Several experiments were performed using SDAKG memories to illustrate recall accuracy, context-based prediction, and computational effort. These experiments were performed in the prediction mode. Input sequences were the truncated versions of the stored sequences, to see if the memory provides correct predictions. Experiments for time complexity evaluation and accuracy of SDAKG and ANAKG networks were conducted on the laptop Dell Inspiron 15R with Intel Core i3 and 4GB RAM.

The first type of experiment was to check the accuracy of SDAKG memory responses after learning in comparison to the response from the associative learning graphs ANAKG presented in [13]. Like SDAKG, the ANAKG uses associative learning based on the input sequences; however, it does not learn synaptic delays.

In these experiments, sentences from Table I were used. In the learning stage, time intervals between consecutive words in sequences were set randomly with a normal distribution with 500 ms mean and standard deviation equal to 20 ms.

Table II shows the results of tests with stimulation inputs and responses of SDAKG and ANAKG neural graphs. The results of the performed experiments show that the SDAKG neural graphs give the correct answers in the prediction mode.

The second type of experiment was conducted using automatic tests on a bigger set of data. The first 1000 sentences with a number of symbols $N \geq 10$ from "The Brothers Grimm Fairy Tales" were chosen to construct SDAKG graphs. To test the created SDAKG structures, a test set of the first 500 sequences was chosen. Each test sequence was gradually applied to the SDAKG network, and the strongest prediction response was compared to the test sequence, and the number of correct responses was noted.

The results, shown in Table III, demonstrate that the increasing number of context input elements quickly increases the percentage of the correct responses. We can conclude that stimulation length equal to 4 is sufficient to obtain results with very high accuracy (98%).

TABLE II

RESULTS OF EXPERIMENTS VERIFYING THE ACCURACY OF SDAKG NEURAL GRAPHS RESPONSES

Simulation inputs	Responses of SDAKG neural graphs	Response of neural effectors of ANAKG neural graphs
I	HAVE A MONKEY HAVE ALSO A SMALL DOG	I HAVE ALSO A MONKEY SMALL DOG
MY	MONKEY IS VERY SMALL MONKEY IS LOVELY	MY
IT	IS VERY LOVELY IS ALSO VERY CLEVER LIKES TO SIT ON MY HEAD CAN JUMP VERY QUICKLY LEARNS QUICKLY	IT
I HAVE	A MONKEY ALSO A SMALL DOG	I HAVE ALSO A MONKEY SMALL DOG
I HAVE A	MONKEY	I HAVE ALSO A MONKEY SMALL DOG
IT IS	VERY LOVELY ALSO VERY CLEVER	IT IS
IT CAN	JUMP VERY QUICKLY	IT CAN JUMP VERY QUICKLY

TABLE III

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF INPUTS

Length of the context	1	2	3	4	5	6	7	8	9
Percentage of correct responses	16.4	55.2	88.2	98.0	98.6	99.2	99.6	99.8	100

The third type of experiment was conducted to check the computational cost of SDAKG neural graphs in comparison to ANAKG neural graphs. We tested how the number of stored sequences influences the number of neurons, gates, synapses, and learning time.

The numbers of neurons are the same as the number of new symbols (words) in the set of sequences. In all experiments, input sequences were taken from *The Brothers Grimm Fairy Tales*. In the following test, time intervals between words were set randomly with a normal distribution with the mean value of 500 ms and a standard deviation of 20 ms. Table IV shows the results of the conducted experiments.

B. Sequence Recognition Under Distortions

The next group of experiments checks recognition of distorted input sequences. During these experiments, the memory operated in the testing mode. One-thousand sentences from *The Brothers Grimm Fairy Tales*, which had at least ten words, were chosen for learning. To lower the impact of randomness on the results, the following experiments were repeated ten times, and the averaged results are displayed in Tables V–X.

The recognition process is realized by checking the activation level of the leaf-gates. Checking of recognition correctness is realized by the comparison of the checked sequence identifier, and the identifier saved in the winner leaf-gate.

First, the recognition of the original sequences (without damages) was tested. In this test, delays between symbols were random with the same parameters as in the learning process. Per 1000 sequences all answers of the networks were correct.

TABLE IV

COMPUTATIONAL COST OF SDAKG AND ANAKG NEURAL GRAPHS

Number of sequences	Number of neurons	Number of SDAKG gates	Number of synapses		Learning time in sec	
			SDAKG	ANAKG	SDAKG	ANAKG
100	590	2773	2739	13610	0.07	1.03
200	856	4563	4497	21506	0.11	1.86
300	1140	6627	6543	30298	0.20	2.52
400	1477	9729	9634	42823	0.52	3.90
500	1618	11515	11407	48735	0.72	4.54
600	1768	13486	13371	55255	0.94	5.71
700	1882	14809	14688	59450	1.25	6.33
800	2040	17036	16908	66681	1.55	7.98
900	2121	18861	18728	72038	1.78	8.41
1000	2264	21395	21254	79938	2.05	10.93

TABLE V

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF REMOVED ELEMENTS

N	1	2	3	4	5	6	7	8	9
%	97.7	96.3	95.3	94.6	93.0	89.1	73.1	44.7	15.1

TABLE VI

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF REMOVED ELEMENTS WITH DIFFERENT STANDARD DEVIATIONS σ_G

N	1	2	3	4	5	6	7	8	9
1000	97.5	95.9	95.2	94.5	93.4	**89.4**	**74.2**	**44.1**	15.1
2000	99.1	98.5	98.0	97.7	95.8	86.6	64.0	43.0	15.0
3000	99.7	99.5	99.3	98.6	**96.0**	84.5	63.0	43.9	15.0
4000	99.8	**99.8**	**99.5**	**98.9**	94.4	83.0	62.0	42.5	15.0
5000	99.9	99.7	99.5	98.5	94.1	81.0	62.1	43.2	14.4
6000	**99.9**	99.7	99.4	98.5	93.6	80.3	62.5	43.4	**15.1**
7000	99.9	99.7	99.4	98.1	92.5	80.9	61.6	43.2	15.1

TABLE VII

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF INSERTED ELEMENTS

N	1	2	3	4	5	6	7	8	9
%	94.5	85.6	72.5	53.6	30.3	14.1	5.7	3.1	0.9

TABLE VIII

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF INSERTED ELEMENTS WITH DIFFERENT STANDARD DEVIATIONS σ_G

N	1	2	3	4	5	6	7	8	9
2000	95.6	92.6	88.2	79.9	62.9	30.8	10.7	3.7	0.9
3000	97.3	94.9	91.5	84.4	69.6	39.0	16.3	4.8	0.9
4000	99.2	97.8	94.7	88.4	72.4	43.5	20.1	5.7	1.0
5000	99.6	99.4	97.3	91.5	75.4	46.0	21.0	6.1	0.9
6000	**99.8**	99.5	97.9	92.7	77.2	47.6	21.8	6.2	**1.0**
7000	99.8	**99.5**	**98.0**	**92.7**	**77.6**	48.5	22.6	6.5	**1.0**
8000	99.7	99.3	97.9	92.6	77.5	49.2	22.6	6.8	1.0
9000	99.7	99.3	97.8	92.2	77.1	49.6	23.1	6.9	1.0
10000	99.7	99.2	97.6	92.1	76.6	49.8	24.1	6.9	0.9

TABLE IX

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF REPLACED ELEMENTS

N	1	2	3	4	5	6	7	8	9
%	97.1	91.3	84.0	73.7	61.1	43.8	28.3	16.5	2.0

In the first test, some elements of the original sequences were removed at random locations within each tested sequence. Standard deviation σ_G used in the synaptic gate

TABLE X

ACCURACY OF SDAKG NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF REPLACED ELEMENTS WITH DIFFERENT STANDARD DEVIATIONS σ_G

N	1	2	3	4	5	6	7	8	9
1000	97.4	91.7	84.0	73.3	60.9	43.9	28.8	16.0	2.2
2000	98.9	98.0	96.2	93.6	85.2	63.9	39.6	19.7	2.2
3000	99.6	99.3	98.9	97.1	91.6	74.0	46.5	21.3	2.0
4000	99.9	99.7	99.4	98.3	93.3	76.7	50.0	21.6	2.0
5000	99.9	99.8	**99.5**	**98.6**	**93.6**	76.3	50.2	**23.0**	**2.1**
6000	**99.9**	**99.8**	99.5	98.3	93.2	**76.9**	**51.4**	21.7	2.0
7000	99.9	99.8	99.4	98.3	93.0	76.7	50.1	22.7	1.9

attenuation function (2) were computed during the memory organization and learning process from $\sigma_G = \sigma_r + (2 * \hat{t}_{\text{mr}}/(n+1)^{1/2}) + \epsilon$. Table V shows the recognition correctness statistic as a function of the number N of removed elements.

Next, tests were to determine the optimum value of the standard deviation σ_G used in the synaptic gate attenuation function (2). Table VI shows the recognition correctness statistic as a function of the number of removed elements N (columns) with different standard deviations σ_G (rows). In Tables VI, VIII, and X, the best accuracy values (in percent) are marked in bold.

In the next test, a recognition level of damaged sequences with several extra elements inserted into random locations within each tested sequence was computed. Only the symbols that do not belong to the examined sequence were taken into account. Table VII shows the recognition correctness statistic as a function of the number of inserted elements N.

As before, we performed multiple tests to determine the optimum value of the standard deviation σ_G used in the synaptic gate attenuation function. Table VIII shows the recognition correctness as a function of the number of inserted elements N (columns) with different standard deviations σ_G (rows).

In the next tests, a recognition level of damaged sequences with several elements within each tested sequence replaced at random locations was computed. Table IX shows the recognition correctness statistic as a function of the number of replaced elements N.

In addition, Table X shows the dependence of the correctness statistics on the standard deviation σ_G (rows).

Since in practical situations all tested types of distortions may occur, we combine the optimum σ_G values to obtain settings for the robust recognition of tested sequences. The optimal values $\sigma_{G1}, \sigma_{G2}, \sigma_{G3}$ for a different number of distorted elements are from the corresponding columns in Tables VI, VIII, and X and are combined using

$$\mu(\sigma_{G1}, \sigma_{G2}, \sigma_{G3}) = \frac{1}{\frac{\frac{1}{\sigma_{G1}} + \frac{1}{\sigma_{G2}} + \frac{1}{\sigma_{G3}}}{3}} = \frac{3}{\frac{1}{\sigma_{G1}} + \frac{1}{\sigma_{G2}} + \frac{1}{\sigma_{G3}}}. \tag{13}$$

Table XI presents these optimum values of σ_G.

Since we cannot anticipate the type of damage, we recommend choosing the value of the standard deviation based on all possible type of damages for a given problem. Thus for the optimum performance with a small number of errors in the input sequence, we recommend using σ_G that is

TABLE XI

AVERAGED ACCURACY OF SDAKG NETWORK FOR COMBINED σ_G VALUES AS A FUNCTION OF THE NUMBER OF FAULTS

N	1	2	3	4	5	6	7	8	9
σ_G	6000	5362	5060	5060	4437	2571	2571	2500	5764

TABLE XII

AVERAGED ACCURACY OF SDAKG NETWORK FOR $\sigma_G = 5000$ AS A FUNCTION OF THE NUMBER OF FAULTS

N	1	2	3	4	5	6	7	8	9
%	99.8	99.6	98.8	96.2	87.7	67.8	44.4	24.1	5.8

ten times the average delay between sequence elements. Applying such single $\sigma_G = 5000$ ms, we get the average recognition results for all tests performed as shown in Table XII.

C. Comparison With Other Sequential Memories

In the next tests, a comparison between recognition level of the SDAKG, LSTM networks [16] and HTM networks [17] was performed on the set of 100 ten-element sequences from "The Grimms Brothers Fairy Tales."

1) Long Short-Term Memory: The test results depend on a type of configurations of the LSTM network used as a sequence classifier. In all experiments with LSTM, we used a special vector with length equal to 32, which represents all separate symbols, N separate outputs for each sequence, and a different number of cells. All the structures had an embedded layer (as the first layer), and a dense layer (as the last layer). In addition to these two layers, we checked several network structures that had the following specifications:

1) an LSTM layer;
2) a) a dropout layer with the noise level equal to 0.2;
   b) an LSTM layer;
   c) a dropout layer with the noise level equal to 0.225.

an LSTM layer with the input and recurrent dropout with the noise equal to 0.2

In these experiments, the noise was introduced to prevent the overfitting of networks during their training. In each example of network structure training which uses the binary cross-entropy loss function, an Adam optimizer was processed in 1000 learning epochs.

Each experiment was repeated 100 times and averages were calculated. In all tests, we used 1000 learning epochs. Damaging of original sequences was introduced to obtain a suitable test set which contains sequences (each represents a separate class) which are similar to corresponding sequences in the training set. The best results were obtained for the configuration Band shown in Table XIII.

Now we can compare the best results obtained by the LSTM experiments of classifications of sequences after the tuning parameters with the results from SDAKG (see Table XIV).

Finally, we can conclude that the results of the optimized LSTM networks are significantly worse than those obtained by the SDAKG networks.

2) Hierarchical Temporal Memory: Next, we compared performance SDAKG and HTM networks on a sequence

TABLE XIII

ACCURACY OF LSTM NETWORK RESPONSES AS A FUNCTION OF THE NUMBER OF DAMAGES (REPLACED ELEMENTS) FOR DIFFERENT NUMBER OF LSTM CELLS (SECOND SERIES OF EXPERIMENTS)

LSTM size	800	900	1000	1100	1200
0	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
1	95.83	95.67	97.11	**97.83**	96.56
2	88.39	87.17	90.81	**92.28**	88.84
3	77.37	75.96	80.49	**81.69**	76.3
4	62.64	62.68	66.31	**66.94**	60.74
5	48.05	49.0	**51.18**	51.04	46.06
6	34.05	34.88	**36.86**	36.48	31.18
7	21.99	23.93	**24.13**	23.5	19.08
8	11.52	**13.53**	12.93	11.74	9.76
9	5.48	**5.72**	5.51	5.04	3.31

classification problem. Each element of each sequence was encoded using random distributed scalar encoder (RDSE) vectors with a length of 2048 bits. The temporal memory network structure had 2048 minicolumns with 32 cells per each column.

During training, associative connections are created in each time step between cells of minicolumns. Only one epoch was used in the training process, as it gave the best results.

After training, both training and test data (damaged sequences with N damages) were used to obtain test statistics. In these tests, three classification methods were used.

1) The first method (called Cell Dist [29]) builds sets of correctly predicted cells, i.e., cells which are both active and were in the predictive mode in the previous time step. These sets are used for similarity matrix calculation between all sequences from the test and training sets. In this matrix, rows represent sequences in the test set, and columns represent sequences in the training set. Let us assume that s_1 represents a set of the correctly predicted cells in the kth element of the ith test sequence, and s_2 represents a set of the correctly predicted cells in the kth element of the jth training sequence. Values of the cells of the similarity matrix are calculated from the following formula:

$$D[i, j] = \sum_{k=1}^{N} \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|} \tag{14}$$

where N is the number of elements of the analyzed sequences.

Finally, accuracy is calculated in the following way. For each sequence i, in the test set, the row from the similarity matrix that represents this sequence is chosen ($D_i[j]$). If the label of the jth sequence for which $D_i[j]$ is maximum is the same as label the of the ith sequence, then the recognition is correct.

2) In the second method (called Active FreqDist [29]), the RDSE vectors from the training and test sets are tested using the temporal memory obtained earlier. For each sequence, the histogram of active cells in all minicolumns is built and is divided by the sum of the histogram elements. The histograms obtained from the training and test sets are used for the similarity matrix calculation using the following formula:

$$M[i, j] = \sum_{k=1}^{H} \sqrt{h_i[k] * h_j[k]} \tag{15}$$

where:

h_i is the histogram of the ith sequence from the test set;

h_j is the histogram of the jth sequence from the training set;

k is the index the of element from the histogram;

H is the length of the histogram.

Finally, the accuracy is calculated identically as in the previous method.

3) The third method (called Pred-Active FreqDist [29]) uses histograms of cells that are both active in the current time step and were in the predictive mode in the previous time step while sequence testing use histograms of active cells as in the second method.

Results of the above methods for a different number of damages in the original sequences were shown in Table XV and compared to results obtained using SDAKG. Each experiment was repeated 100 times, and the averages were saved in Table XV.

Among the methods using temporal memory, the best results were obtained using the third method. However, all methods gave results worse than those using SDAKG.

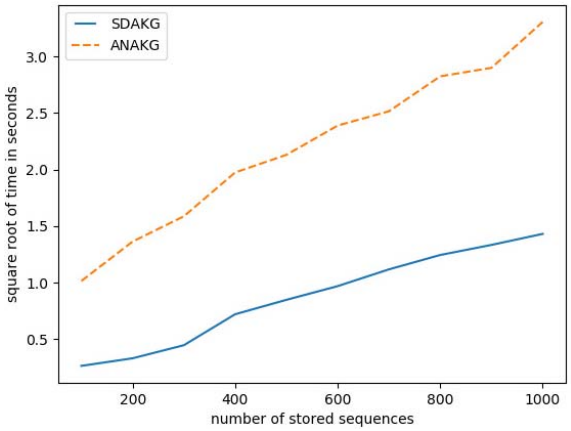

Fig. 7. Square root of time needed to store the specified number of sequences.

D. Timing Analysis

Fig. 7 compares the learning times needed to construct ANAKG and SDAKG memories. The vertical axis shows the square root of the computation time. We can conclude that the time complexity of learning of both methods is $O(N^2)$, where N is the number of sequences. However, the construction and analysis of the SDAKG graphs are more efficient.

The time performance comparison was also tested for SDAKG, LSTM [16], and HTM [17] networks. Table XVI shows the total training times of these networks for the increasing number of sequences.

For LSTM memories, the 1000 learning epoch and 1100 memory cells were used. We noticed that increasing the number of memory cells in LSTM resulted in a fast

TABLE XIV
ACCURACY OF SDAKG AND LSTM NETWORKS RESPONSES AS A FUNCTION OF THE NUMBER OF REPLACED ELEMENTS

N	1	2	3	4	5	6	7	8	9
SDAKG	100	99.9	99.7	99.1	94.9	86.4	62.4	31.2	3.9
LSTM	98.2	92.4	82.3	68.1	53.3	37.8	23.2	12.6	5.1

TABLE XV
ACCURACY OF HTM NETWORKS RESPONSES AS A FUNCTION OF THE NUMBER OF REPLACED ELEMENTS

N damages	Method 1	Method 2	Method 3	SDAKG
1	71.4	65.2	75.9	100.0
2	46.1	37.3	49.6	99.9
3	26.5	20.1	28.9	99.7
4	15.1	11.1	15.8	99.1
5	8.4	6.4	9.4	94.9
6	4.7	4.1	5.0	86.4
7	2.7	2.7	2.9	62.4
8	1.9	1.8	1.9	31.2
9	1.2	1.1	1.1	3.9

TABLE XVI
LEARNING TIMES FOR TEMPORAL MEMORY IN LSTM, HTM, AND SDAKG

Sequences	Training SDAKG [s]	Training HTM [s]	Training LSTM [s]
100	0.01	5.35	1898
200	0.03	14.59	3694
300	0.05	29.94	5273
400	0.10	50.57	6940
500	0.14	76.17	8661
600	0.17	107.03	10335
700	0.25	143.26	11691
800	0.32	185.75	13684
900	0.41	240.04	15118
1000	0.54	290.73	16859

increase in testing time and memory requirement. However, keeping a constant number of LSTM cells resulted in a drop in test accuracy. For HTM memories, the networks with a constant number of minicolumns (2048) and cells for each mini-column (32) were used. The training of HTM networks using datasets consists of building sparse representation and temporal memory training stage when associative connections are forming using a different number of sequences. In Table XVI, HTM training times contain both of described stages. Notice that SDAKG training times are smaller than in Table IV because of the sequences limited to ten elements are used here.

Table XVII shows testing times per sequence comparing SDAKG, LSTM, and HTM memories. For HTM these times contain a calculation of responses for training and test sets, calculation of appropriate similarity matrices and final recognition, calculated for all methods described earlier (M1—Cell Dist, M2—Active FreqDist, M3—Pred-Active FreqDist). The averaged testing times are presented for all networks in msec.

1) Testing Larger Data Set: Next, we checked if SDAKG analysis scales well for larger networks. The averaged learning and testing times for original sequences are presented in Table XVIII for increasing the number of sequences from 1000 up to 10 000.

During these experiments, the tests were performed for all sequences with damages from 0 to 9 replaced elements without using special σ_G settings. The results of the recognition of sequences for 10 000 sequences for a different number of replaced elements were similar to those obtained testing

TABLE XVII
TESTING TIMES FOR LSTM, SDAKG, AND HTM NETWORKS

Sequences	Testing of SDAKG [ms]	Testing of LSTM [ms]	Testing of HTM M1 [ms]	Testing of HTM M2 [ms]	Testing of HTM M3 [ms]
100	0.11	9.52	18.52	78.12	77.59
200	0.15	8.19	20.3	161.34	153.76
300	0.19	7.75	43.26	222.93	218.59
400	0.26	7.68	51.96	302.75	306.89
500	0.29	7.55	51.23	357.7	349.74
600	0.33	7.49	90.76	454.55	445.6
700	0.36	7.47	157.49	506.38	498.5
800	0.41	7.49	117.48	580.6	570.99
900	0.49	7.59	144.2	685.48	676.16
1000	0.56	7.57	76.49	715.86	702.23

TABLE XVIII
LEARNING AND TESTING TIMES FOR SDAKG NETWORKS FOR A LARGER DATASET

Sequences	Learning [s]	Testing [ms]
1000	0.5	0.53
2000	1.6	0.77
3000	3.1	1.00
4000	5.3	1.28
5000	8.3	1.73
6000	12.2	1.99
7000	16.5	2.34
8000	21.2	2.76
9000	27.0	2.99
10000	32.9	3.37

damages in 1000 sequences, which indicates the robustness of this approach.

Finally, we tested 57 090 sequences on the set of sequences obtained from the entire story *Les Miserables* by using ten consecutive words for each sequence. Learning time was 1511 s.

VI. CONCLUSION

We presented a new concept of associative memories in which synaptic connections of the self-organizing neural network elements learn time delays between the input sequence elements. Thus, synaptic connections represent both the synaptic weight and the expected delays. This facilitates recognition of time sequences and, at the same time, provides the context-based association between the observed sequence elements. Characteristics of time delays are learned each time an input sequence is presented. The corresponding SDAKG is constructed, and then it operates from the moment a first input sequence is presented. There are no separate learning and testing modes, as the network starts to predict the next input element as soon as there is no expected input signal. In such a case, the network enters the prediction mode and continues to predict remaining elements of the stored sequence. These prediction signals depend on the presented input context and the knowledge stored in the graph. This mode of operation is preferred for the organization of episodic memories in which memories are used to store the episodes as well as to recall them if a sufficient context is provided. Such associative sequential recall is useful for the operation of working memory in a cognitive agent.

The network can accommodate more than one distribution of sequential delays by building parallel synapses with

different statistics. Postsynaptic gates are activated according to the signal strength that results from time deviation from the expected delays and synaptic efficacy which represents how often a specific postsynaptic gate was activated in comparison to activation of the presynaptic gate. Synaptic efficacy may indicate a learned preference of an expected input over another input. Such preference can be enhanced by including the significance of the incoming signal. However, due to space limits, we did not present the results of the significance-based preference. Moreover, a hierarchical organization of SDAKG networks will yield memories in which multiple sequences can be played back after partial sequence recognition was taken place on the lower level of the hierarchy. This will be a subject of further studies.

Test results demonstrate that the network correctly recognizes the input sequences if the delays between the input elements are within learned statistical distributions of such delays. The comparison to ANAKG networks shows that the developed approach is more efficient, requires less simulation time to construct and analyze the network, and provides more accurate results. Performed sequence recognition under distortions of the sequence elements demonstrated the great robustness of SDAKG memories, particularly as compared to LSTM and HTM memory networks. This feature is extremely important in practical applications where recognition of the input symbols, missing symbols, or other types of distortions are frequently observed.

This paper deals only with symbolic inputs, where each input neuron represents a single observed object, and only a single Observed object neuron is activated each time. Thus, the resulting memory describes the storage of sequences and associations typically performed on higher levels of the memory hierarchy. We currently work on vector/matrix form of the SDAKG input that could be used on the lower levels of the memory hierarchy and applied to problems like speech recognition, when an input is a sequence of vectors that contain numerical values of the cepstral coefficients.

REFERENCES

[1] J. R. Bider and R. H. Desai, "The neurobiology of semantic memory," *Trends Cogn. Sci.*, vol. 15, no. 11, pp. 527–536, Nov. 2011.

[2] W. Kintsch, "The role of knowledge in discourse comprehension: A construction-integration model," *Psychol. Rev.*, vol. 95, pp. 163–182, Apr. 1988.

[3] K. J. Lang, A. H. Waibel, and G. E. Hinton, "A time-delay neural network architecture for isolated word recognition," *Neural Netw.*, vol. 3, no. 1, pp. 23–43, 1990.

[4] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[6] R. Sun and C. L. Giles, "Sequence learning: From recognition and prediction to sequential decision making," *IEEE Intell. Syst.*, vol. 16, no. 4, pp. 67–70, Jul. 2001.

[7] D. Wang and M.-A. Arbib, "Complex temporal sequence learning based on short-term memory," *Proc. IEEE*, vol. 78, no. 9, pp. 1536–1543, Sep. 1990.

[8] D. Wang and B. Yuwono, "Anticipation-based temporal pattern generation," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 4, pp. 615–628, Apr. 1995.

[9] L. Wang, "Learning and retrieving spatio-temporal sequences with any static associative neural network," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 45, no. 6, pp. 729–739, Jun. 1998.

[10] J. H. Wang, M. C. Tsai, and W. S. Su, "Learning temporal sequences using dual-weight neurons," *J. Chin. Inst. Eng.*, vol. 24, pp. 329–344, Apr. 2001.

[11] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 6, pp. 1411–1423, Nov. 2006.

[12] V. A. Nguyen, J. A. Starzyk, W.-B. Goh, and D. Jachyra, "Neural network structure for spatio-temporal long-term memory," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 6, pp. 971–983, Jun. 2012.

[13] A. Horzyk and J. A. Starzyk, "Fast neural network adaptation with associative pulsing neurons," in *Proc. IEEE Symp. Ser. Comput. Intell.*, 2017, pp. 339–346.

[14] A. Horzyk, "How does generalization and creativity come into being in neural associative systems and how does it form human-like knowledge?" *Neurocomputing*, vol. 144, pp. 238–257, Nov. 2014.

[15] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] J. Hawkins, S. Ahmad, and D. Dubinsky, "Cortical learning algorithm and hierarchical temporal memory," Tech. Rep., 2011. [Online]. Available: http://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf

[18] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.

[19] M. A. Kramer, "Autoassociative neural networks," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, 1992.

[20] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Prague, Czech Republic, May 2011, pp. 5528–5531.

[21] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–15.

[22] J. Tani and S. Nolfi, "Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems," *Neural Netw.*, vol. 12, pp. 1131–1141, Oct. 1999.

[23] J. Tani, "Learning to generate articulated behavior through the bottom-up and the top-down interaction processes," *Neural Netw.*, vol. 16, no. 1, pp. 11–23, 2003.

[24] D. George and J. Hawkins. (2011). *Invariant Pattern Recognition Using Bayesian Inference on Hierarchical Sequences*. [Online]. Available: http://www.cs.cmu.edu/~./dgovinda/pdf/recog/DilJeffTechReport.pdf

[25] Y. Cui, C. Surpur, S. Ahmad, and J. Hawkins, "Continuous online sequence learning with an unsupervised neural network model," *J. Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, Nov. 2016.

[26] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA, USA: MIT Press, 2003, pp. 593–600.

[27] J. A. Starzyk and J. Graham, "MLECOG: Motivated learning embodied cognitive architecture," *IEEE Syst. J.*, vol. 11, no. 3, pp. 1272–1283, Jul. 2017.

[28] A. Horzyk and J. A. Starzyk, "Associative fine-tuning of biologically inspired active neuro-associative knowledge graphs," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2018, pp. 2068–2075.

[29] *Numenta Platform for Intelligent Computing*. Accessed: Jun. 20, 2019. [Online]. Available: https://programtalk.com/vs2/?source=python/9411/nupic.research/projects/sequence_classification/run_sequence_classifcation_experiment.py

Janusz A. Starzyk (LSM'16) received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the Warsaw University of Technology, Warsaw, Poland, and the Habilitation degree in electrical engineering from the Silesian University of Technology, Gliwice, Poland.

He was an Assistant Professor with the Institute of Electronics Fundamentals, Warsaw University of Technology. He was a Professor of electrical engineering and computer science with Ohio University, Athens, OH, USA. Since 2007, he has been the Head of the Information Systems Applications, University of Information Technology and Management, Rzeszow, Poland. His current research interests include embodied machine intelligence, motivated goal-driven learning, self-organizing associative spatiotemporal memories, active learning of sensory-motor interactions, machine consciousness, and applications of machine learning to autonomous robots and avatars.

Łukasz Maciura received the Ph.D. degree in computer science (computer vision specialty) from the Silesian University of Technology, Gliwice, Poland, in 2011.

He was a Post-Doctoral Fellow with the University of Information Technology and Management, Rzeszow, Poland, where he is currently an Assistant Professor. He was an Assistant Professor at the University of Rzeszow, Rzeszow. His Ph.D. thesis "Mosaicing of images from capsule endoscopy" was published in Lap Lambert Academic Publishing, Saarbrucken, Germany in 2015. He is a researcher experienced in computer vision and machine learning. His current research interests include computer vision, machine learning (especially deep learning and time-series recognition), HCI, medical informatics, and robotics.

Adrian Horzyk (SM'18) received the M.S. degree in computer science from Jagiellonian University, Krakow, Poland, and the Ph.D. and Habilitation degrees in computer science from the AGH University of Science and Technology, Krakow.

He is currently an Associate Professor with the Faculty of Electrical Engineering, Automatics, Computer Science, and Biomedical Engineering, Department of Biocybernetics and Biomedical Engineering, AGH University of Science and Technology. His current research interests include the development of knowledge-based models and methods of artificial intelligence and computational intelligence, associative and spiking models of neurons and their networks, self-developing semantic associative memories, new machine learning strategies and techniques, data science, data mining, knowledge engineering methods, and cognitive systems.

Dr. Horzyk has been the Co-Founder and a member of the Polish Association of Artificial Intelligence since 2009 and has been a Board Member of the Polish Neural Network Society PTSN since 2011. He has been the Deputy Team Leader of AGH University of Science and Technology in CERN Alice experiments and projects since 2017.