Concurrent Associative Memories With Synaptic Delays

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

***Abstract*— This article presents concurrent associative memories with synaptic delays useful for processing sequences of real vectors. Associative memories with synaptic delays were introduced by the authors for symbolic sequential inputs and demonstrated several advantages over other sequential memories. They were easy to organize and train. It was demonstrated that they were more robust than long short-term memories in recognition of damaged sequences. The associative memories can be applied in combination with deep neural networks to solve such symbol grounding problems, such as speech recognition, and support sequential memories triggered by sensory inputs. Several practical considerations for developed memories were discussed and illustrated. A continuous speech database was used to compare the developed method with LSTM memories. Tests demonstrated that the developed approach is more robust in recognition of speech sequences, particularly when the test sequences are damaged.**

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

I. Introduction

MEMORIES are essential elements of cognitive systems, providing storing learned knowledge and episodes of its past interactions with the environment [1]. Artificial neural network memories for storage and recall of input sequences were intensively studied over many years [2]–[7]. In a similar effort, a fast neural network adaptation with associative pulsing neurons [8] was proposed as a self-organizing associative memory capable of storing time-domain sequences. The time delay is a critical factor in recognizing time-domain sequences. In spiking neural networks [9], the neurons can spontaneously self-organize into groups and generate patterns of polysynchronous activity [10]. Synchronization in polysynchronous groups is possible since all neurons have different axonal conduction delays, which, in natural neural networks, is a function of the distance between the neurons and axonal transmission speed. If spikes arrive at polysynchronous neurons simultaneously, the neurons fire, and if the time of arrival is different for various inputs to a polysynchronous neuron, the neuron does not fire. Thus, effectively, a polysynchronous group of neurons can recognize the time-domain sequence when the input neurons are activated at different moments. Because these neurons transmit their spikes to the polysynchronous group with various delays, if the difference between these delays corresponds to the differences between input activations, then the polysynchronous neurons fire [10]. This idea was developed in [11] to create an associative network of pulsing neurons that can store and recognize the sequences of activation of neurons that represent the input symbols. The synaptic delay associative knowledge graph (SDAKG) network described in [11] implemented this idea of synchronous firing for recognition of sequences that contain symbols of elements, such as letters or words.

SDAKG is a new type of associative memory in which synaptic connections of the self-organizing neural network include information about time delays between input sequence elements. Thus, synaptic connections represent both the synaptic weights and expected delays between the network inputs. The SDAKG structure facilitates the recognition of time sequences and provides context-based associations between sequence elements. Time delays are learned and are updated each time an input sequence is presented. There are no separate learning and testing modes, as the network starts to predict the next input element if there is no expected input signal. The network generates output signals that are useful for associative recall and prediction. These output signals depend on the presented input context and the knowledge stored in the graph.

SDAKG addressed several practical issues that made the application of polysynchronous groups of spiking neurons difficult to implement in practical situations. First, building and training in such networks are very long—it took 6 h of computer time for a 1000-neuron network to initialize its synaptic connections, and it took 24 h of simulation time to establish stable polysynchronous groups. The detection of polysynchronous groups in the network of spiking neurons is difficult, and it is challenging to control associations between neuronal representations. In the SDAKG network, we used a fast neural network adaptation with associative pulsing

Manuscript received August 8, 2019; revised May 13, 2020 and September 30, 2020; accepted November 23, 2020. Date of publication December 10, 2020; date of current version August 4, 2021. This work was supported by the Polish National Science Centre, Twardowskiego 16, 30-312 Krakow, Poland, under Grant DEC-2016/21/B/ST7/02220 and Grant AGH 16.16.120.773. *(Corresponding author: Adrian Horzyk.)*

Janusz A. Starzyk is with the Faculty of Applied Computer Science, University of Information Technology and Management in Rzeszów, 35-225 Rzeszów, Poland, and also with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701 USA.

Marek Jaszuk and Łukasz Maciura are with the Faculty of Applied Computer Science, University of Information Technology and Management in Rzeszów, 35-225 Rzeszów, Poland.

Adrian Horzyk is with the Department of Biocybernetics and Biomedical Engineering, AGH University of Science and Technology in Kraków, 30-059 Kraków, Poland (e-mail: horzyk@agh.edu.pl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2020.3041048.

Digital Object Identifier 10.1109/TNNLS.2020.3041048

neurons [8] to organize the memory of synaptic gates capable of storing input sequences.

SDAKG's focus was on the development of the network structure in the processing of the symbolic inputs. However, many practical applications (such as recognition of speech signals or sequences of images) require recognition of sequences represented by vectors of real numbers. Also, many of the input neurons can be simultaneously partially activated, unlike neurons that represent symbols that are either ON or OFF. In this work, we modify the SDAKG algorithm to recognize such sequences using partially activated input neurons and synaptic gates. As a result, a new algorithm for time-series recognition tasks by the propagation of signals in the separate paths of synaptic gates that represent classes of sequences was created and described in this article. This tool was connected with a convolutional neural network on the input to assess the class membership of frames.

Section II describes the concept of the concurrent SDAKG organization. In Section III, practical considerations about SDAKG networks and experiments on random data were described. In Section IV, several series of experiments on a speech data set using a combination of SDAKG with deep and convolutional neural networks were presented. These architectures were compared with the combination of LSTM with deep and convolutional neural networks and gave better results than the LSTM-based structures for damaged data. Section V presents tests on the sign language data set comparing SDAKG to LSTM. Section VI contains time analysis and tests of processing speed, comparing LSTM and SDAKG implementations.

II. SDAKG With Partially Activated Inputs

In the SDAKG network, synaptic connections represent both the synaptic weights and expected delays between the network inputs. Both the synaptic weights and their delay characteristics are learned during the training process. The network generates output signals that are useful for associative recall and prediction. These output signals depend on the presented input context and the knowledge stored in the graph. Effectively, the whole input sequences can be recalled when sufficient context information is provided. This is useful for the self-organization of episodic memories that are used for the storage of the observed episodes. The stored episodes can be later recalled if a sufficient context is provided. It was demonstrated that the network correctly recognizes the input sequences with variable delays and that it is more efficient than the recently developed sequential memory network based on associative neurons [8]. Sequence recognition under distortions of the sequence elements demonstrated the greater robustness of SDAKG memories compared with long short-term memories (LSTMs) [12]. This feature is extremely important in practical applications where recognition of the input symbols, missing symbols, or other types of distortions is frequently observed.

Presented in this work modified version of the synaptic delay associative knowledge graph called a concurrent SDAKG here is to replace the input neurons that represent symbols with neurons that represent objects based on features extracted from the sensory data. Several such features may be required to represent a single object for recognition of an object or its part. In addition, the features may be recognized to different degrees of similarity between the stored and observed ones. The final difference between the previous and this implementation of the SDAKG network is that several (or all) features may be simultaneously partially activated in the modified approach; thus, the modified SDAKG network must process parallel inputs.

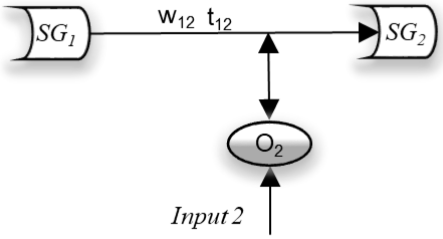

Fig. 1. Basic links between the synaptic gates and an input neuron.

A. Self-Organization of the Synaptic Gates Memory

The SDAKG memory is a self-organizing network, establishing synaptic gates and associations between object neurons as needed. The basic structure of SDAKG memory uses input (object) neurons O_i and synaptic delay gates SG_i connected through the learned weights $w_{(i-1)i}$. The input neurons are responsible for object recognition, while synaptic delay gates are responsible for context-based signal propagation. If an activated input neuron corresponds to the first element of a memorized sequence, then it directly stimulates a synaptic delay gate; otherwise, its activation is combined with the weighted output of the presynaptic gate that represents the context of the received input sequence (see Fig. 1).

In learning temporal sequences, an input signal activates, first, a presynaptic gate and, then, the associated postsynaptic gate. In Fig. 1, SD1 represents a presynaptic gate, and SG2 represents a postsynaptic gate to input neuron O2.

A presynaptic gate may point to several postsynaptic gates. Thus, in the training mode, if a postsynaptic gate that is pointed to by a presynaptic gate receives the strongest input activation a_{ni}^t, it combines the input signal from its presynaptic gate output multiplied by the interconnection weight plus the activation of its input neuron

$$I_i^t = a_{g(i-1)}^t * w_{(i-1)i} + a_{ni}^t \tag{1}$$

where I_i^t is the input activation of a postsynaptic gate, $a_{g(i-1)}^t$ is an output activation level of the presynaptic gate, $w_{(i-1)i}$ is an interconnection weight between the presynaptic and postsynaptic gates, and a_{ni}^t is a level of activation of the postsynaptic gate input neuron. Activation of the postsynaptic gate input neuron a_{ni}^t depends on the match between its weights (that represent a given object) and the input signal vector at a given time instant.

In the training mode, if the most activated input neuron is not connected to a postsynaptic gate, then a new synaptic delay gate is established and is connected with the presynaptic gate. The interconnection weight (as introduced in Section IV-A [11]) between presynaptic and postsynaptic gates is computed using the formula described in Section II-B. Notice that, in the training mode, only the activation of the most activated input neuron is considered, and it is used to modify the input state of its synaptic delay gate using (1). In the testing mode, activation of all input neurons is considered, and all of them modify the input activation of their corresponding synaptic delay gates. This makes all the synaptic gates in the network working concurrently. For simplicity of the description and the discussion, we assume here that the SDAKG network structure was developed during the training mode that is separate from the testing mode, while this restriction is not necessary and is not used in the real use of SDAKG networks.

B. Gate Deactivation Rates

In applications such as speech recognition, sequences of data contain vectors of cepstrum coefficients represented by real numbers. After normalization, these numbers can represent partial activations of the input neurons in SDAKG memory adjusted to recognize such sequences. Since, in continuous speech recognition, we do not know the exact moment when the sequence begins, the SDAKG memory must respond to such sequences dynamically whenever they occur. This requires dynamic adjustment of the activation levels of various synaptic delay gates. Partially activated gates will gradually lose their activation level, getting ready for a new activation in response to new data. Therefore, proper adjustment of deactivation rates of the synaptic gates is required.

One way of doing this is to relate the deactivation rates to the significance of the activated input neuron. Synaptic gates with more significant inputs should have lower deactivation rates. This will result in holding significant information in such gates for longer periods of time. In intelligent agents (e.g., motivated learning agents described in [13] and [14]), the significance of the observed inputs can be related to agent motivations and its reward system. Events that are associated with desirable situations (and a potential award) or those that carry a potential threat (or a punishment) to the agent will have higher significance than other events that the agent either observed in the past or those that he/she is unfamiliar with. However, in applications, in which no significance of the input activations can be established based on the agent motivations, a simple approach can be used based on the event frequency. More frequent events can be declared as less significant and, therefore, can be quickly forgotten, while those that occur seldom should be sustained activation of synaptic gates for a longer period of time. While this way of establishing significance is easier, it does not represent a true significance of events that are specific to the agent, its value system, and the history of interactions within a complex environment. Therefore, it should not be used when other more meaningful ways are available.

In the SDAKG approach, the deactivation rate of the synaptic gates is related to the frequency of activations of their input neurons. In the symbolic version of SDAKG memory, this will correspond to finding probabilities of input neuron activations. If an input neuron is activated with low probability, its synaptic gate will have a low deactivation rate. Using the deactivation rate, the activation of the synaptic gate is reduced using (2), depending on the time difference between consecutive activations of the input neurons

$$a_g^t = a_g^{t_0} - \alpha * p_g * (t - t_0) \tag{2}$$

where α is a constant scale coefficient, p_g is a probability of firing of the synaptic gate of the input neuron, t_0 is the time of the last activation of the synaptic gate, and t is the current time.

For simultaneous partial activations of many input neurons, instead of estimating neuron's firing probabilities, we accumulate sums of partial activations of input neurons and divide them by the total sum of activations of all input neurons

$$r_{g_k} = \frac{\sum_{i=1}^{m} a_{n_{ki}}}{\sum_{k=1}^{n} \sum_{i=1}^{m} a_{n_{ki}}} \tag{3}$$

where m is the number of activations of input neurons, n is the number of input neurons, $a_{n_{ki}}$ is the kth neuron activation level at the ith instance of the input vector activation, and r_{g_k} is a deactivation rate of the synaptic gate associated with the kth input neuron. Activation of a synaptic gate is then computed using (2) with p_g replaced by r_{g_k}.

Since the deactivation rates are relatively low in short periods of time, the synaptic gates maintain their long-term activation levels. In a short time, they frequently change their activation levels according to changes in the activation of their corresponding input neuron. Since, during concurrent processing of input data, many input neurons change their activation levels, the short-term operation of the synaptic gates is a key feature for sequence recognition. Thus, the long-term activation of synaptic gates provides a broader context for sequence recognition. In this article, we will focus on short-term changes in the synaptic gate activations.

In the concurrent implementation, the computation of the gate input activation level is obtained from (4) for all inputs simultaneously

$$I_{gi}^t = \max\left(a_{g(i-1)}^{t-1} * w_{(i-1)i} + a_{ni}^t, I_{gi}^{t-1}\right) \tag{4}$$

where $w_{(i-1)i}$ is an interconnection weight between the presynaptic and postsynaptic gates, a_{ni}^t is an activation level of the input neuron, $a_{g(i-1)}^{t-1}$ is an activation level of the output of the presynaptic gate at time $t-1$, and I_{gi}^t is an activation level of the input of the gate at time $t-1$. Simultaneous activations of various gate inputs mean that no signal propagation is used. Instead, the gates are activated based on the previous gate input activation level, the current activation of the gate input neuron, and the previous output activation level of the presynaptic gate.

III. Practical Considerations in Concurrent SDAKG

Several practical issues must be considered while dealing with real-world data and organizing concurrent SDAKG memory. These issues include the effect of noise, variable length of input sequences, and weight adjustment for the input gates. This section discusses some of these issues. We illustrate these issues on synthetic data to have a better understanding of their role in the recognition of real sequences.

A. Uniform Noise Consideration

In real-life applications, the recognition of the input signals is not perfect. Neurons that represent specific objects are often partially activated, even when the sensory input does not include the object represented by this neuron. We consider this partial activation as a noise of object recognition. This noise may harm the recognition of sequences stored in SDAKG memories.

Therefore, it is useful to establish a noise level for the inputs of the synaptic gates. In the most straightforward approach, we compute the noise n_{av} as the average activation of all inputs in the network by all signals that are in the training set and do not belong to the sequence that we want to recognize.

Once we know the noise level, we generate the input noise sequences and apply them to all input nodes. These input nodes stimulate the synaptic gates, which accumulates the total noise signal from all presynaptic gates and their inputs. In many applications, including speech or object recognition, all the input neurons are fully connected to the sensory inputs. In such cases, similarity to the input patterns is measured on all input nodes. Therefore, only the length of the input sequences stored in the SDAKG memory and the connection weights of their input neurons differentiate the synaptic gates response to the noise. In general, longer sequences accumulate more noise in synaptic gates and distort sequence recognition. To offset this effect, we need to consider that the expected noise will influence the synaptic gate activations.

For uniform noise distribution, the gate input activation level is obtained from (5) for all inputs simultaneously

$$I_{gi}^{t} = \max\left(a_{g(i-1)}^{t-1} * w_{(i-1)i} + a_{ni}^{t}, I_{gi}^{t-1}\right) - n_{av} \tag{5}$$

where n_{av} is an average noise level.

Example 1: To verify that such modification of the SDAKG operation is capable of concurrent operation and correct recognition of the input sequences, we tested the recognition rates for 100 runs of simulation with synthetic 47 phonemes, where we assumed a uniform noise from 0 to 0.4. Individual signal frame recognition [the level of activation of the postsynaptic gate of the input neuron in (1)] was the uniformly sampled noise from 0.2 to 1. The simulated phoneme signals were of variable length between 3 and 26 frames. The results were the average recognition level of 93.81%, with a standard deviation of 2.62%.

In real phonemes, both noise and signal activation were different than in the assumed uniform noise model in Example 1, but the simplification used in Example 1 is useful to demonstrate the properties of the concurrent SDAKG.

B. Scaling Factor

We argued that, since, in the database, we have sequences of various lengths, the longer sequences can accumulate more signals, and the effect of the noise signal is stronger than in the shorter sequences. To offset this effect, we tried to gradually reduce the maximum increment in the accumulated signal strength due to the noise for longer sequences. We related the rate of the decrease to a normalized integral from the sequence length histogram. In the test data, the average sequence length was 8.85, its standard deviation was 4.47, and its histogram was shown in Fig. 2.

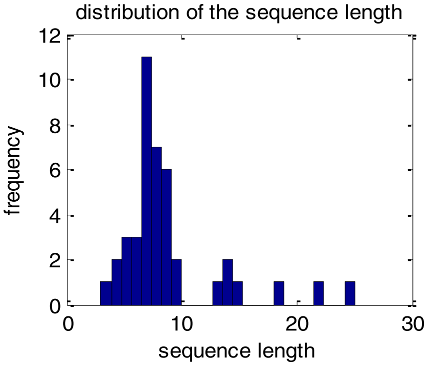

Fig. 2. Histogram of the tested sequence length.

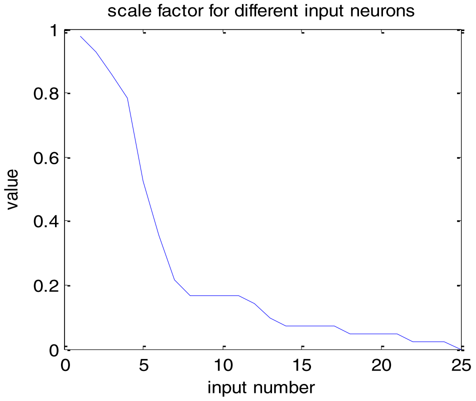

Fig. 3. Scale factor as a function of the input number.

Using the normalized integral from the histogram of the length, we obtained the following scaling factor for each consecutive input in the synaptic gates network:

$$S_f(l) = 1 - \frac{\int_0^l f(l)dl}{\int_0^{l_{\max}} f(l)dl} \tag{6}$$

where l corresponds to the location of the synaptic gate from the beginning of the stored input sequence (input number), $l_{\max}$ is the maximum length of the sequence in SDAKG memory, and $f(l)$ is the frequency of the input sequences of the length l. The obtained scale factor vector for a given distribution of phonemes lengths was as follows:

$$\begin{aligned} S_f = [&0.97, 0.92, 0.86, 0.79, 0.52, 0.36, 0.21, 0.17, 0.17,\\ &0.17, 0.17, 0.14, 0.10, 0.07, 0.07, 0.07, 0.07, 0.05,\\ &0.05, 0.05, 0.05, 0.02, 0.02, 0.02, 0].\end{aligned}$$

Fig. 3 shows the values of the scale factor as a function of the input number. As we can see, the longer sequence can experience lower scale factor values.

TABLE I

RECOGNITION RATES FOR VARIOUS LEVELS OF NORMALLY DISTRIBUTED NOISE WITH VARIOUS MEANS AND STANDARD DEVIATIONS

Standard deviation	Various noise signals						
	μ=0	μ=0.125	μ=0.25	μ=0.5	μ=1	μ =2	μ =3
σ=0.125	75.4	76.3	77.0	80.7	90.5	99.5	100
σ=0.25	33.3	33.8	35.3	41.5	65.1	98.0	100
σ=0.5	8.82	9.11	9.61	11.4	22.6	90.5	99.9
σ=1	2.37	2.38	2.49	3.23	6.32	53.4	98.5
σ=2	0.07	0.07	0.08	0.17	1.05	9.5	68.4
σ=3	0.00	0.00	0.00	0.01	0.09	3.5	17.5

When the scale factor is used, the gate input activation level is obtained from (7) for all inputs simultaneously

$$I_{gi}^{t} = \max\left(a_{g(i-1)}^{t-1} * w_{(i-1)i} + S_f(i) * a_{ni}^{t}, I_{gi}^{t-1}\right) - S_f(i) * n_{av}. \quad (7)$$

The next few examples show the effect of using the scale factor on sequence recognition.

Example 2: We tested the effect of the scaling factor on the recognition rates for 100 runs of simulation with 47 phonemes, organized as in Example 1. After using the obtained scale factor to weigh the input signals, the average recognition rate was improved from 93.81% to 99.76%, with a standard deviation of 0.72%.

Example 3: In this example, we tested how the recognition level is affected by changes in the noise and signal levels.

When the noise was increased to a uniform noise between 0 and 0.6, the average recognition level after 400 runs of the simulation was dropped to 94.42%, with a standard deviation of 2.63%. Finally, when the noise was increased to a uniform noise between 0 and 0.8, the average recognition level after 400 runs of the simulation was dropped to 50.45%, with a standard deviation of 5.96%.

An additional test was performed with the noise level returned to a uniform noise between 0 and 0.4, and the useful signal was noisier and uniformly sampled between 0 and 1. After 400 runs of simulation, the correct recognition was 99.27%, with a standard deviation of 1.23%.

C. Normal Noise Distribution

The second set of tests was performed, assuming that both the signal and the noise have normal distributions. We assumed that the signal has a normal distribution with zero mean and the standard deviation equal to 1. Noise signals had means and the standard deviations, as shown in Table I. Table I has the average recognition rate results of 400 simulation runs each.

The recognition rates are sharply reduced when the two distributions overlapped. The reason for such a dramatic drop, when the difference between the mean values is reduced to 1, is that there are many more synaptic delay gates that receive the noisy input than the number of gates that receive the signal input. With numerous noise signals, some of them receive signal strength higher than the correct input and cause misclassification.

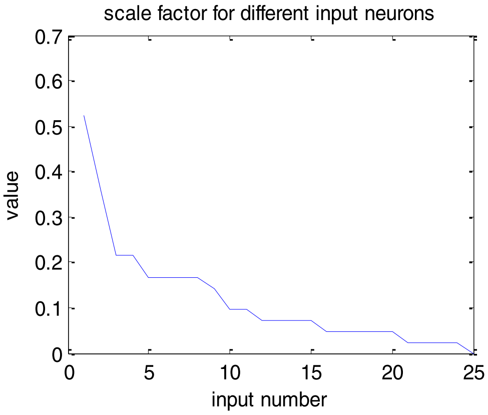

Fig. 4. Scale factor as a function of the input number.

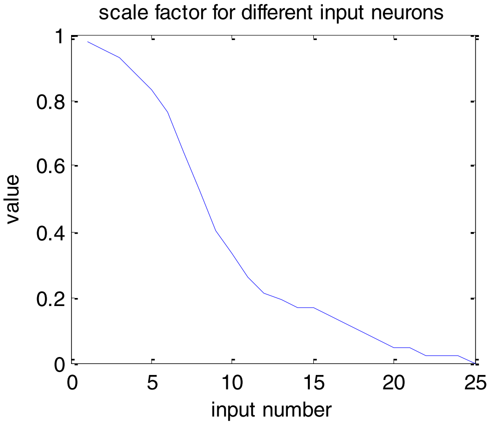

Fig. 5. Histogram of the sequence length and the scale factor vector.

D. Effect of Sequence Length Distribution

The next group of tests was performed to check how variations of the sequence length affect the recognition results.

In the first test, short sequences (of the lengths lower than the median length of all sequences) were replaced by the longer ones giving the scale factor vector, as shown in Fig. 4. The average recognition rate for mean = 3 and std = 2 was 60.48%, which is lower than for the corresponding case in Table I (68.4%).

Equally unsuccessful was the increase in the number of very short sequences with the scale factor vector, as shown in Fig. 5. The average recognition rate for mean $\mu = 3$ and the standard deviation $\sigma = 2$ was 58.81%, which is lower than for the corresponding case in Table I.

Since Figs. 4 and 5 are obtained like Fig. 3 using (6), they reflect the statistical distribution of the length of the analyzed sequences in the data set, and their shape may be used to obtain the optimum value of the scale coefficient. From its definition, we see that the scale factor depends on the distribution of the sequence length and corresponds to the probability of finding sequences longer than a given length. In general, the optimum scale coefficient depends on the distribution characteristic for a given database and should be recomputed depending on data.

E. Power of Scale Factor Vector

Since differentiation of the scale factors improved the recognition rate, we have raised the scale factor vector to various

TABLE II

RECOGNITION RATES IN % FOR VARIOUS LEVELS OF POWER SCALING FACTOR VECTORS FOR THE NORMALLY DISTRIBUTED NOISE WITH SELECTED MEANS AND STANDARD DEVIATIONS

Power value α	1	1.25	1.5	1.75	2	2.5	4	8
μ =.5 σ=0.125	80.7	80.9	**80.9**	80.6	80.6	80.1	78.7	73.8
μ =1 σ=0.25	65.1	66.2	**67.0**	66.5	66.0	65.7	64.4	60.1
μ =2 σ=1	53.4	56.2	57.5	59.3	59.7	**60.3**	60.0	57.8
μ =3 σ=2	68.4	72.4	75.9	78.0	79.0	80.8	82.4	**82.8**
μ =4 σ=3	77.4	81.8	85.1	86.8	88.0	89.5	91.2	**92.4**

powers α between 1 and 8. Thus, the gate input activation level is obtained from (8) for all inputs simultaneously

$$I_{gi}^{t} = \max\left(a_{g(i-1)}^{t-1} * w_{(i-1)i} + S_f^{\alpha}(i) * a_{ni}^{t}, I_{gi}^{t-1}\right) - S_f^{\alpha}(i) * n_{av} \qquad (8)$$

where α is determined experimentally.

The results shown in Table II indicate that the optimum value of the scaling factor power α depends on the type of distribution of the training data. This value is smaller for more narrow noise distributions (comparing to signal distribution) and is larger for a broad noise distribution with larger values of the standard deviation.

When the mean value is very close to the mean value of the signal, good recognition may be obtained with a smaller value of α. However, no matter what was the distribution, selecting power greater than 1 improved the results, in some cases, vastly, e.g., for $\mu = 4$ and $\sigma = 3$, the recognition was improved by 15.26%, and for $\mu = 3$ and $\sigma = 2$, the recognition was improved by 11.68%. On the other hand, using too large values α may lower recognition rates for noise distributions close to the signal distribution with the small standard deviation σ; for instance, for $\mu = 0.5$ and $\sigma = 0.125$, the recognition was dropped by 15.77%.

The important question is if the optimum power scaling factor improves recognition for all values of μ and the given σ. This question is important when a distribution of a given standard deviation is getting closer to the distribution of the signal. Table III shows the test results for the power scaling factor set to 2.

The results were better in most cases than when no scaling factor was used (compare with Table I).

The cases when the noise standard deviations are higher than the standard deviation of signals are more important for phoneme distribution since the number of noisy signals is many times larger than the number of useful signals for a given phoneme. With power $\alpha = 2$, the scale factor vector results in a much smaller influence of the longer sequences.

F. Covariance-Based Similarity

Since the real data, distributed in multidimensional space, are often characterized by a multivariate normal distribution with the mean value vector μ and the covariance matrix Σ, we tested the effect of changes in the similarity between points that result from various covariance matrices. We then tested how these changes in covariance-based similarity affect the sequence recognition in SDAKG structures.

TABLE III

RECOGNITION RATES FOR THE POWER OF SCALING FACTOR VECTORS SET TO 2 FOR NORMALLY DISTRIBUTED NOISE WITH THE SELECTED MEANS AND STANDARD DEVIATIONS

Noise signal μ	0	0.125	0.25	0.5	1	2	3
σ							
0.125	74.9	75.9	77.0	80.6	90.0	99.4	100
0.25	37.6	38.4	40.3	45.4	66.0	97.7	100
0.5	10.9	11.1	11.5	13.8	27.7	90.5	100
1	2.32	2.44	2.55	3.32	6.95	59.7	98.7
2	0.13	0.11	0.15	0.24	1.08	15.5	79.0
3	0.00	0.00	0.01	0.05	0.20	5.44	40.8

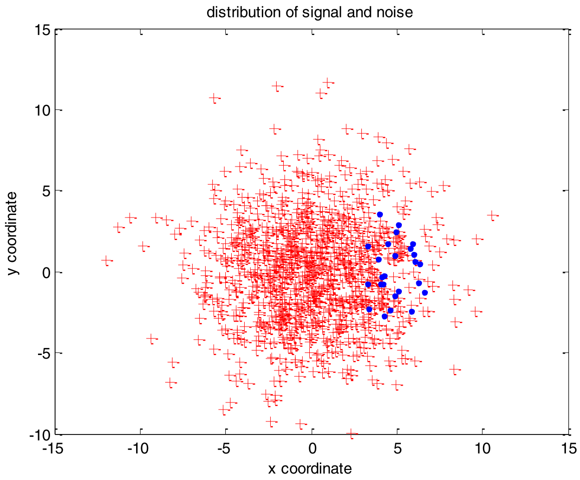

Fig. 6. 2-D distribution of the signal and the noise for testing of the covariance-based similarity of test sequences.

To simplify our discussion and illustrate it with the plots of data, we consider a 2-D case and assume that the noise signals are generated by the multivariate normal distribution with zero mean vector $\mu_n = [00]$, and the covariance matrix is characterized by σ_n

$$\Sigma_n = \begin{bmatrix} \sigma_n & 0 \\ 0 & \sigma_n \end{bmatrix}. \qquad (9)$$

The signal data are generated using the mean vector $\mu_s = [\mu 0]$, and the covariance matrix is characterized by σ_s

$$\Sigma_s = \begin{bmatrix} 1 & 0 \\ 0 & \sigma_s \end{bmatrix}. \qquad (10)$$

Fig. 6 shows an example distribution of the noise signal with $\mu_n = [0\ 0]$ and $\Sigma_n = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$ and the signal distribution with $\mu_s = [5\ 0]$ and $\Sigma_s = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$ that follow these simplified conditions.

In covariance-based similarity, we calculate the similarity of the input test vector x to the noise and the signal and set the activation level of the input neuron at time t using

$$a_{ni}^{t} = e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}. \qquad (11)$$

TABLE IV

RECOGNITION RATES WITH α =2 FOR THE NORMALLY DISTRIBUTED NOISE WITH SELECTED MEANS AND STANDARD DEVIATIONS

Signal μ	μ=0	μ=1	μ =2	μ =4	μ =8
Signal σ					
σ=0.125	94.8	95.5	96.4	98.7	99.9
σ=0.25	90.9	91.7	93.5	97.6	99.9
σ=0.5	84.1	84.8	88.5	95.5	99.8
σ=1	73.2	75.0	79.8	92.2	99.8
σ=2	57.9	60.6	67.7	86.7	99.6
σ=4	42.0	44.5	51.2	77.7	99.5
σ=8	27.6	29.6	36.6	66.4	99.2

Table IV shows recognition rates in % for various values of μ_s and σ_s. As we can see in 2-D space, the recognition is more robust than in the 1-D case, whose results were presented in Tables I–III. Even with a total overlap of the noise and the signal distributions (for $\mu_s = 0$ and $\sigma_s = 1$), the sequence recognition using concurrent SDAKG is over 73% correct.

If the distribution of signal samples is more concentrated, as in the case when σ is smaller, the recognition correctness improves. Separation of mean values also helps but less than the concentration of samples in the signal distribution. This property will facilitate the recognition of speech signals, as noise signals will not be so much concentrated as a useful signal.

IV. TESTS ON SPEECH DATA SET

In practical tests, the developed associative memories were tested on a speech data set [15]. We cut all samples of phonemes to the maximum number of 10 frames to simplify numerical calculations. Thus, each sample of phoneme consists of various numbers of frames (from 3 to 10) after the length of each sequence was limited to 10. Each frame consists of 39 factors: 13 Mel Frequency Cepstral Coefficients (MFCCs) [16], 13 first derivatives of MFCC, and 13 second derivatives of MFCC. MFCCs are widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein [16] in the 1980s and have been the state of the art ever since.

A. Adjusting the Sequence Length

Phonemes in the selected database are of variable lengths changing from 3 to 801 frames. This makes the task of phoneme recognition more difficult than when the sequence length is more uniform. To obtain all sequences with an equal number of frames, we stretch and compress the sequences, respectively.

1) Sequence Stretching: Let us assume that we have N elements of the original sequence. We need to obtain a new sequence with a bigger number of elements ($M > N$). To stretch the sequence, the values of a longer sequence are computed based on the values of the original (shorter) sequence.

First, virtual indices of the new sequence elements (in the index space of the original sequence) are computed. If the indices of the original sequence are from 0 to $N - 1$ (and from 0 to $M - 1$ in the new sequence), then the virtual index v_j is computed from

$$v_j = j * \frac{N-1}{M-1} \tag{12}$$

where v_j is a virtual index mapped from the new sequence to the index space of the original sequence (where j changes from 0 to $M - 1$); j is an index of the longer sequence; N is the number of elements of the original sequence; and M is the number of elements of the new sequence.

When a virtual index is a real number, this number is used to calculate an interpolated value from two elements of the original sequence. Elements in the new sequence are calculated from the following formula:

$$b_j = a_i * (i + 1 - v_j) + a_{i+1} * (v_j - i) \tag{13}$$

where $i < v_j < i + 1$.

In the case when the virtual index v_j is an integer (e.g., for the first and last elements in the sequence), then the new element in the longer sequence is equal to

$$b_j = a_{v_j} \tag{14}$$

where a_i and a_{i+1} are the values of two nearest elements (from the original sequence) to a virtual index v_j where i is lower than v_j and $i + 1$ is bigger than v_j; b_j is the value of an element in the new longer sequence; i is an index of the original sequence; and j is an index of the new longer sequence.

Example 4: Assume that the original sequence contains three elements (0.5, 3, 1), and we want to get a sequence consisting of four elements. The virtual indices computed using (12) are 0, 0.67, 1.33, and 2, and new coefficients computed from (13) and (14) are 0.5, 2.17, 2.33, and 1.

Although the example that we gave computed elements of a scalar sequence, the similar sequence stretching can be applied to a sequence of vector values of the same dimension.

2) Sequence Compression: If we have the original sequence with the number of elements $N > M$ and we want to compress this sequence to the sequence of M elements, we use the following process. First, we must obtain the virtual indices of the new elements of the sequence. For this purpose, we use (12).

Elements in the new sequence are calculated from the following formula:

$$b_j = \sum_{i \in S_j} a_i * w_{ij} \tag{15}$$

where S_j is a set of indices of the old sequence which fulfills the following condition:

$$|v_j - i| < t \tag{16}$$

and w_{ij} is a weight that can be calculated from the following formula:

$$w_{ij} = \frac{w^*_{ij}}{\sum_{k \in S_j} w^*_{kj}} \tag{17}$$

where

$$w^*_{ij} = 1 - \frac{|v_j - i|}{t} \tag{18}$$

TABLE V

STRUCTURE OF THE SEQUENCE OF NEURAL NETWORKS USED IN THE EXPERIMENT

Type of layer	**Number of Neurons**	**Activation**
Input	39	Identity
Dense	500	Relu
Dense	500	Relu
Dense	500	Relu
Dense	500	Relu
Dense	500	Relu
Dense	500	Relu
Dense	500	Relu
Dense	7*47=329	Sigmoid
SDAKG or LSTM	329	Eq. (7) (SDAKG) or tanh (LSTM)
Dense	47	Sigmoid

TABLE VI

RECOGNITION RATES IN % OF SDAKG AND LSTM NETWORKS FOR A DIFFERENT NUMBER OF DAMAGES

Number of damages	0	1	2	3		4	5	6
LSTM	60.0	52.1	48.1	42.1		33.2	21.9	10.30
SDAKG	56.0	54.3	51.6	47.4		40.4	29.1	13.4

t is a step size calculated from

$$t = \frac{N-1}{M-1} \tag{19}$$

v_j is a virtual index mapped from the shorter sequence to the index space of the original sequence, a_i is a value of the original sequence, b_j is a value of the new shorted sequence, i is an index of the original sequence, j is an index of the new shorted sequence, N is a number of elements of the original sequence, and M is the number of elements of the new shorted sequence.

B. Experiments on All Classes of the Resampled Sequences of Phonemes

The following experiments were conducted on a set of all 47 phonemes treated as separate classes. Each class contained a different number of sequences with a different number of frames taken from the database [15]. After resampling, (stretching or compression) sequences are represented by matrices 7 × 39, where 7 represents the number of frames after resampling and 39 represents MFCC factors with first and second derivatives. The data set was divided into the training set, which contains 202 515 sequences, and the test set, which contains 103 093 sequences.

First, a deep neural network was trained on the training set to recognize individual frames in each of 47 phonemes based on 39 MFCC factors obtained from separate frames. This deep neural network was constructed using the Keras library [16] (with TensorFlow backend) and contains eight layers, including seven layers with 500 neurons and the last output layer with $7 * 47 = 329$ neurons (see Table V).

The first layer of the deep neural network is connected to the 39-D input. In the last (output) layer, the sigmoid activation function was used, while, in the other layers, the rectifier activation function (“relu”) was used. The network was trained using the binary cross-entropy loss function, Adam optimizer, and accuracy metric. Finally, the network structure with trained weights was saved for future use. This configuration was found experimentally. To assess the quality of the trained network, we used the standard procedure of dividing the data into training and validation sets in proportions of 2/3 and 1/3, respectively. We determined that the specified network gave the maximal level of validation accuracy.

In the following test, the outputs of the previously trained deep neural network were used to activate SDAKG or LSTM networks [12] used for sequence recognition in the whole 47 phonemes data set. As demonstrated in [11], the symbolic SDAKG was superior to LSTM in recognition of damaged sequences. Two experiments were designed to check if such properties apply to the concurrent SDAKG that uses real number input vectors instead of symbolic inputs.

In this experiment, the eight-layer deep neural network was used to obtain similarities of the frame to different synaptic gates that represent different steps of phonemes. The outputs of the last layer of the deep network that contains 329 gates were used as inputs to SDAKG networks and LSTM networks for the comparison of these networks in the sequence recognition task. The LSTM network architecture consists of 329 cells that represent elements of sequences recognized by the deep neural network and one fully connected layer with 47 neurons, each to recognize one class of phonemes. The number of cells of the LSTM network was selected to match the number of the synaptic delay gates in SDAKG.

To achieve comparability between both these networks, the fully connected layer trained using the pseudoinverse method was added on the top of the SDAKG network in the following way. After SDAKG simulations, the values of SDAKG gates were taken as the input to a fully connected layer. This fully connected layer has 329 inputs and 47 outputs that represent classes of phonemes. The layer is trained using the states of gates after SDAKG simulation on the training data. Both networks were tested on test data sets used as input of the deep neural network, while the outputs of the deep network were inputs of SDAKG and LSTM networks.

After the test on clean test data sets, the data were damaged in the following way. In the beginning, the noise sets were created for each class of phonemes. These noise sets contain foreign frames that do not belong to a current data set but belong to all other data sets. For each sequence in the original test data set, a damaging operation is as follows. First, the positions of frames in the damaged sequence related to the number of damages are chosen randomly. In the second step, the randomly chosen foreign frames from the noise data sets are used to replace the original frames in the sequence.

Table VI shows the results of SDAKG and LSTM networks. The LSTM network was trained with the same number of cells as the number of gates in the SDAKG network in order to achieve comparable results with the SDAKG network.

TABLE VII

RECOGNITION RATES OF SDAKG AND LSTM NETWORKS WITH 1000 LSTM CELLS FOR A DIFFERENT NUMBER OF DAMAGES

Number of damages	0	1	2	3	4	5	6
LSTM	60.6	57.3	52.7	45.8	36.1	23.4	11.1
SDAKG	56.0	54.3	51.6	47.4	40.4	29.1	13.4

TABLE VIII

RECOGNITION RATES OF SDAKG AND LSTM NETWORKS WITH 1000 LSTM CELLS FOR A DIFFERENT NUMBER OF DAMAGES (WITH THE DEEP NEURAL NETWORK ON THE INPUT AND THREE DENSE LAYERS WITH DROPOUTS ON THE OUTPUT OF BOTH NETWORKS)

Number of damages	0	1	2	3	4	5	6
LSTM	61.3	56.7	50.7	42.3	31.4	19.8	9.7
SDAKG	61.3	57.5	52.6	46.2	36.9	24.0	10.9

TABLE IX

OBTAINED MODEL OF THE 1-D CONVOLUTIONAL NEURAL NETWORK AFTER A SERIES OF EXPERIMENTS

Type of layer	**Number of neurons**	**Size of kernel**	**Activation**
Convolutional 1D	64	5	Relu
Convolutional 1D	64	5	Relu
Convolutional 1D	64	3	Relu
Max Pooling 1D	-	2	-
Convolutional 1D	128	3	Relu
Convolutional 1D	128	3	Relu
Max Pooling 1D	-	2	-
Convolutional 1D	256	3	Relu
Convolutional 1D	256	3	Relu
Max Pooling 1D	-	2	-
Dropout=0.15	-	-	-
Flatten	-	-	-
Dense	1024	-	Relu
Dense	1024	-	Relu
Dense	7*47=329	-	Softmax

The results for LSTM were obtained using the output of the network after examination of all frames.

The results presented in Table VI show that, for the same number of LSTM cells as SDAKG gates, the SDAKG networks gave better results in the recognition of damaged sequences.

The next experiment was conducted using 1000 LSTM cells instead of 329. As demonstrated in Table VII, the LSTM networks with these parameters gave better results than the SDAKG networks for damages between 0 and 2. When the number of damages was higher than 2, then the SDAKG networks gave better results.

C. Experiments With the Three-Layer Perceptron on the Output

New series of experiments were conducted with the addition of three dense layers with dropouts on the output of both concurrent SDAKG and LSTM networks instead of one perceptron layer learned using the pseudoinverse method. On the output of the LSTM network, two dense layers with the number of neurons equal (1000 and 47) and two dropouts before each of them was added and trained together with LSTM. On the output of the SDAKG network, three dense layers with several neurons equal (550, 1000, and 47), and two dropouts between them were added and trained. These final dense layers were optimized separately for the SDAKG and LSTM networks. Table VIII shows the results of the described experiments, namely, an increase in accuracy in both the networks, but, in the case of the SDAKG networks, the increases were higher, and the results are now comparable for data without any damages. In the case of damages, using SDAKG gave better results than LSTM.

D. Experiments Using Convolutional Neural Networks on the Input

To obtain better results of both networks (SDAKG and LSTM) and achieve better SDAKG advantage on data with damages, the 1-D convolutional neural network was used instead of a deep neural network.

First, we designed a 1-D convolutional neural network for the recognition of individual sequence frames, aiming at better recognition accuracy than the deep neural network model obtained earlier. The number of layers and layer types, the number of neurons, kernel sizes, and other parameters of the convolutional network were optimized.

The model of the obtained 1-D convolutional neural network presented in Table IX gave 49.38% recognition accuracy of individual frames, while the deep neural networks used earlier had 48.08% recognition of individual frames of sequences.

This model has 39 inputs and 329 outputs, as in the earlier eight-layer deep neural network. The training process was conducted using the binary cross-entropy loss function, Adam optimizer [17], [18], and accuracy metrics. Since the model gave better results for individual frames, it could be used as input of SDAKG and LSTM networks to compare them. The final tests were conducted to compare LSTM and SDAKG networks using the obtained 1-D convolutional neural network (see Table IX) on the input and a few dense layers with dropouts on the output of both networks, as in the previous experiments.

On the output of the LSTM network, two dense layers with the numbers of neurons equal to (1000 and 47) and two dropouts before each layer were added and trained together with LSTM. On the output of the SDAKG network, three dense layers with the numbers of neurons equal to (550, 1000, and 47) and two dropouts between these layers were added and trained. Table X shows the final results of these experiments.

Both these networks gave better accuracy than in experiments described in Sections IV-B and IV-C for the test data with and without damages with the advantage of SDAKG memories in all cases.

V. TESTS ON THE SIGN LANGUAGE DATA SET

To extend the verification of the model performance on real noisy data, we tested it on the Australian Sign Language data set [20]. We used the methodology analogous to the one

TABLE X

RECOGNITION RATES OF SDAKG AND LSTM NETWORKS WITH 1000 LSTM CELLS FOR A DIFFERENT NUMBER OF DAMAGES (WITH THE CONVOLUTIONAL NEURAL NETWORK ON THE INPUT AND THREE DENSE LAYERS WITH DROPOUTS ON THE OUTPUT OF BOTH NETWORKS)

Number of Damages	0	1	2	3	4	5	6
LSTM	61.0	57.3	52.1	44.9	35.6	23.2	11.1
SDAKG	61.6	57.9	53.6	47.6	39.0	27.0	12.4

applied to the speech data. In this case, the data consist of samples of 95 symbols recorded on five signing persons with the aid of the Nintendo Power Glove device.

A. Network Architecture and Results of Sequence Damaging

The particular signs are represented as variable-length sequences of vectors of 11 numerical parameters. The sequence length ranged between 6 and 4494, but most of the sequences were in the range of 40–50 frames. To adapt the data to our experiments, we adjusted the sequence length using the procedure described in Section IV-A. Taking into account the specifics of data, we could not make the sequences too short because this leads to losing precision. On the other hand, we wanted to avoid too long sequences because, in the applied system of coding input data, this leads to a large number of units in the SDAKG and the compared LSTM networks. We chose 20 frames per sequence as a compromise between the data precision and the network size.

For processing with the SDAKG network, the original data were encoded with a deep neural network. In this case, we used a six-layer network with 11 inputs and 1900 outputs, which results from the product of the number of symbols and the number of frames ($95 * 20 = 1900$). The number of units in each of the hidden layers was 2000. This network encodes the class of the symbol and the number of frames for a given input vector. The network configuration was identified analogously as in the previous case, i.e., on the basis of a validation error.

The further experiment is analogous to the experiment for the network presented in Table V. We trained the SDAKG network based on the encoding generated with the deep neural network. The SDAKG was followed by the fully connected layer to recognize the symbol after processing the whole sequence of frames. We analyzed the network performance after damaging the input data. The damaging procedure was analogous to what we did in the case of the speech data. The results are compared with the results obtained with the LSTM network, which is preceded by the deep network, and followed by one dense layer used for classification. This experiment shows that the SDAKG network is more resistant to damages in input data than the LSTM network (see Fig. 7).

In the case of undamaged data, the results are very similar for both models, but, with the growing number of damages, the results of LSTM begin to deteriorate quickly, while the SDAKG results deteriorate much slower.

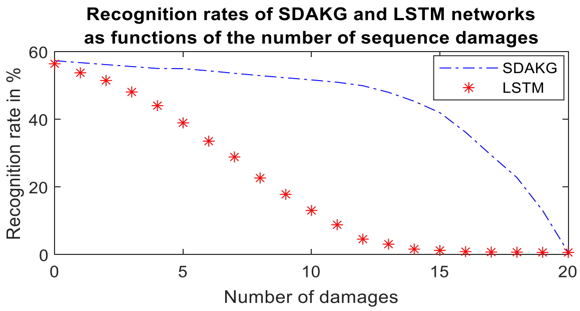

Fig. 7. Comparison of the SDAKG and LSTM network performances on the Australian Sign Language data [20] with respect to the number of damages in the input signal.

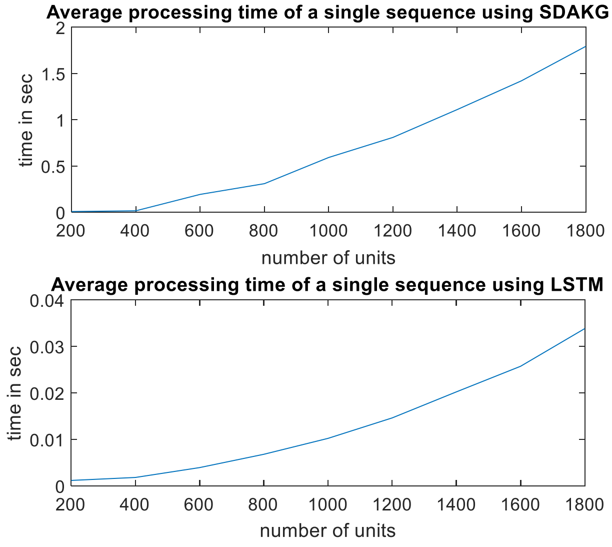

Fig. 8. Average times of processing of a single sequence of the SDAKG and LSTM networks.

VI. PROCESSING SPEED

An essential factor in each computational model is its processing speed. To estimate the performance of the SDAKG network in terms of speed of data processing, we compared results for both the SDAKG and the LSTM networks. Signal preprocessing involves adjusting sequence length by stretching (13) or compression (15) operations, and its computational effort is proportional to the input sequence length. For a given range of sequence length variability, the worst case effort is proportional to the length of the longest sequences, and it grows linearly with the sequence length. After resampling, all sequences are represented as sequences of vectors. Each such vector is processed by a deep neural network and outputs of the network update gate activation levels simultaneously of SDAKG structure using (7), so the processing time depends linearly on the number of sequence elements. Since, after the preprocessing stage, all sequences have the same number of elements, the processing time of the SDAKG algorithm is almost linear and proportional to the number of sequence elements.

To test the computational time complexity of the processing of a single sequence, we measured the performance of the networks for the different numbers of symbols to be recognized. As already discussed, in the presented experiments, the number of symbols (assuming a constant number of frames per symbol) defines the size of the SDAKG network. We used 20 frames in each simulation and the number of symbols ranging between 10 and 90. Thus, the resulting number of network units was ranging between $10 * 20 = 200$ and $90 * 20 = 1800$. The compared LSTM network in each step of the experiment had the same number of units as the SDAKG network. Fig. 8 shows the average times of processing of a single sequence in seconds. From the time analysis plot shown in Fig. 8, we can see that, although SDAKG is slower than LSTM, its time complexity is almost linear, so it can be applied to recognize long sequences.

The implementation of the SDAKG network, used in the experiments, is in pure Python with the NumPy library, while the LSTM network comes from the Keras library with TensorFlow backend (CPU version). The difference in implementations may explain the difference in execution speed, as presented in Fig. 8.

VII. Conclusion

This article presented the organization and functionality of the new concurrent associative memories with synaptic delays. This work described an extension of the symbolic form of SDAKG, where the inputs were represented by sequences of symbols (e.g., words) instead of vectors of real sensory data. SDAKG structures are easy to organize and train and are robust to distortions of input data.

Several practical issues, including the effect of noise, variable length of input sequences, and weight adjustment for the input gates, were considered in order to deal with real-world data. This form of associative memories can be applied in combination with deep neural networks to solve such symbol grounding problems, such as speech recognition and other forms of sequential memories triggered by sensory inputs.

Several series of experiments on real-world data were conducted using this tool. The best results for speech data were obtained when 1-D convolutional neural networks for recognition of individual frames (elements of sequences) were used as input to the SDAKG network. Experiments on both the speech data set and the sign language data set demonstrated SDAKG's advantage over LSTM networks in the case of damaged data. From the analysis of the data processing speed, we can conclude that the SDAKG networks could be handy for real-time recognition tasks of sequential inputs.

References

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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] A. Horzyk and J. A. Starzyk, "Fast neural network adaptation with associative pulsing neurons," in *Proc. IEEE Symp. Comput. Intell.*, Honolulu, HI, USA, Nov./Dec. 2017, pp. 1–8.

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

[10] E. M. Izhikevich, "Polychronization: Computation with spikes," *Neural Comput.*, vol. 18, no. 2, pp. 245–282, Feb. 2006.

[11] J. A. Starzyk, L. Maciura, and A. Horzyk, "Associative memories with synaptic delays," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 1, pp. 331–344, Jan. 2020, doi: 10.1109/TNNLS.2019.2921143.

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

[13] J. A. Starzyk, J. Graham, and L. Puzio, "Needs, pains, and motivations in autonomous agents," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 11, pp. 2528–2540, Nov. 2017.

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

[15] M. Szymański and J. Bachan, "Interlabeller agreement on segmental and prosodic annotation of the jurisdict polish database," *Speech Commun.*, vol. 14, no. 15, pp. 105–121, 2012.

[16] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech Signal Process.*, vol. ASSP-28, no. 4, pp. 357–366, Aug. 1980.

[17] *Keras*. Accessed: May 8, 2020. [Online]. Available: https://keras.io/

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[19] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.

[20] M. W. Kadous. *Australian Sign Language Signs Data Set*. The UCI Machine Learning Repository. Accessed: Apr. 20, 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Australian+Sign+Language+sign

Janusz A. Starzyk (Life Senior Member, IEEE) received the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the Warsaw University of Technology, Warsaw, Poland, in 1971 and 1976, respectively, and the Habilitation degree in electrical engineering from the Silesian University of Technology, Gliwice, Poland, in 2008.

He has been an Assistant Professor with the Institute of Electronics Fundamentals, Warsaw University of Technology. He has been 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 in Rzeszów, Rzeszów, Poland. His current research interests include embodied machine intelligence, motivated goal-driven learning, self-organizing associative spatiotemporal memories, active learning of sensor–motor interactions, machine consciousness, and applications of machine learning to autonomous robots and avatars.

Marek Jaszuk received the Master of Physics degree from the Faculty of Mathematics and Physics, Maria Curie-Skłodowska University, Lublin, Poland, in 1997, and the Doctor of Physical Sciences degree from the Faculty of Mathematics, Physics and Informatics, Maria Curie-Skłodowska University, in 2002.

He has carried out numerous research projects financed by the Polish National Science Center, Krakow, Poland, Polish National Center for Research and Development, Warsaw, Poland, and other institutions supporting scientific activities. He has been an Assistant Professor with the Department of Information Systems Applications, University of Information Technology and Management in Rzeszów, Rzeszów, Poland, since 2004. His research focuses on modeling semantic memory in cognitive systems and robot perception systems. He also deals with issues related to processing and integration of data received from sensors and creating representations of the environment by mobile robots.

Łukasz Maciura received the Ph.D. degree in computer science (computer vision specialty) from the Silesian University of Technology, Gliwice, Poland, in 2011. His Ph.D. thesis titled "Mosaicing of Images From Capsule Endoscopy" was published in Lap Lambert Academic Publishing, Saarbrücken, Germany, in 2015.

He was an Assistant Professor with the University of Rzeszów, Rzeszów, Poland. He is also a researcher experienced in computer vision and machine learning. He is currently an Assistant Professor with the University of Information Technology and Management in Rzeszów, Rzeszów, where he was a Post-Doctoral Fellow. His research interests contain computer vision, machine learning (especially deep learning and time-series recognition), human-computer interfaces (HCI), medical informatics, and robotics.

Adrian Horzyk (Senior Member, IEEE) received the M.S. degree in computer science from Jagiellonian University, Kraków, Poland, in 1997 and the Ph.D. and Habilitation degrees in computer science from the AGH University of Science and Technology, Kraków, in 2001 and 2014, respectively.

He has been a Deputy Team Leader of the CERN Alice experiments and projects at the AGH University of Science and Technology since 2017. 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 encompass 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 a Board Member of the Polish Neural Network Society (PTSN) since 2011.