Neuroevolutionary mechanisms in the synthesis of spiking neural networks

Serhii Leoshchenko[a], Andrii Oliinyk[a], Sergey Subbotin[a], Matviy Ilyashenko[a] and Artem Borovikov[a]

[a] *National university "Zaporizhzhia polytechnic", Zhukovskogo street 64, Zaporizhzhia, 69063, Ukraine*

Abstract

Since the advent and start of active research on the results of IBM True North, attention to spiking neural networks (SNN) has increased significantly. Such neural networks have even greater potential in the field of artificial intelligence than deep, recurrent and other modern artificial neural network (ANN) architectures. This is easily explained by the ease of their arrangement into neuromorphic systems. However, in most cases, it is difficult to synthesize and train such neuromodels, because classical methods cannot be applied to such complex models. A number of works suggest the use of Composite Pattern Producing Networks. This approach is more similar to another method of ANN synthesis, namely a group of neuroevolution methods that, together with the meta-parameters of the network, evolutionarily modify its structure: Topology and weight Evolving Artificial Neural Network (TWEANN). Therefore, the aim of this work is to investigate the possibility of using neuroevolution methods and its individual mechanisms during SNN synthesis.

Keywords [1]

machine learning, artificial intelligence, artificial neural networks, neuroevolution, synthesis, topology, spiking neural network, recurrent neural network, deep neural network, deep learning

1. Introduction

Quite often, the operation of ANN is compared to the work of the brain. However, such networks have taken only the most superficial form and essence from real, living organisms. Due to the too large difference between real neural networks and ANNs, it is necessary to invent different surrogate methods of network training. Naturally, nowhere in nature does not exist something like backpropagation network training, just as there is no unsupervised learning in its pure form [1-4].

SNNs were created with a greater reference to the real work of the brain, and use a method of transmitting information in the likeness of biological neurons (Fig. 1). So, for example, in brain neurons, an impulse is generated at the moment when the current sum of changes in the membrane potential crosses the threshold [5-7]. The pulse occurrence rate and time model of pulse bundle carry information about the external stimulus and ongoing calculations. SNN is based on a similar method of pulse generation and information transmission: neurons use differentiated, nonlinear activation functions, the use of which allows you to create structures with a thickness of more than one layer.

The first SNN model was proposed back in 1952 by Alan Hodgkin and Andrew Huxley [8]. The main feature of such a model is the generation and propagation of action potentials in neurons [8]. After that, with biological refinements and high computational costs, various models of neurons were proposed: Jolivet, Timothy and Gerstner [9]; Izhikevich [10]; Delorme [11]. The latter is very

CMIS-2022: The Fifth International Workshop on Computer Modeling and Intelligent Systems, May 12, 2022, Zaporizhzhia, Ukraine
EMAIL: sergleo.zntu@gmail.com (S. Leoshchenko); olejnikaa@gmail.com (A. Oliinyk); subbotin@zntu.edu.ua (S. Subbotin); matviy.ilyashenko@gmail.com (M. Ilyashenko); atemiks@gmail.com (A. Borovikov)
ORCID: 0000-0001-5099-5518 (S. Leoshchenko); 0000-0002-6740-6078 (A. Oliinyk); 0000-0001-5814-8268 (S. Subbotin); 0000-0003-4624-4687 (M. Ilyashenko); 0000-0003-1429-5930 (A. Borovikov)

popular, as it takes into account the properties of an external stimulus, accumulating the flow of charge through the cell membrane, when crossing a certain threshold.

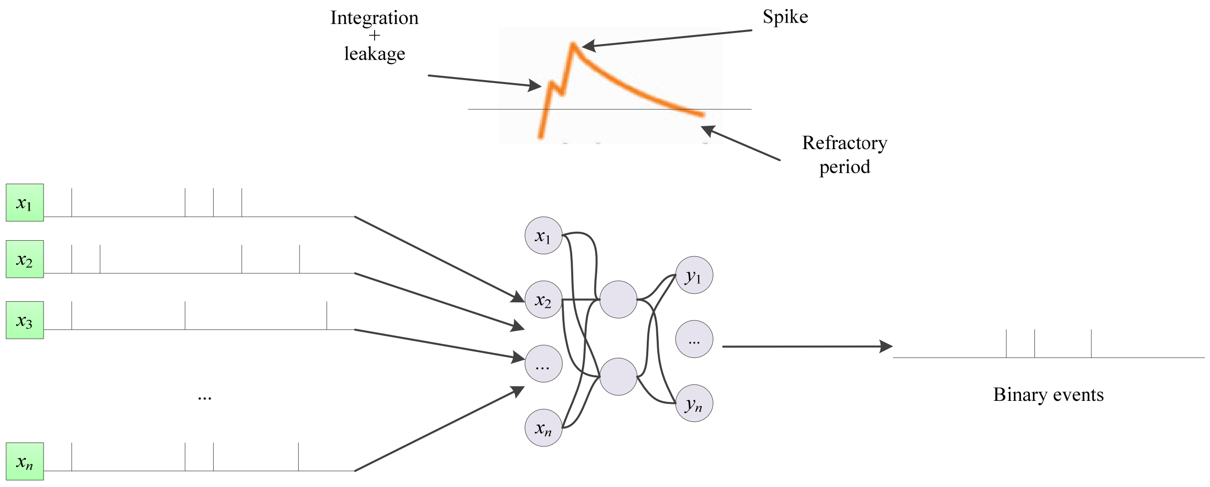


Figure 1: The simple example of SNN architecture

Later, thanks to the research of Kohonen, Grossberg and Anderson, a powerful theoretical foundation was formed, with the help of which it became possible to further develop ANN, namely the design and implementation of multilayer structures [5-7]. But learning is still a huge challenge. Since activation functions are derived, there is room for using gradient optimization methods to train neural networks. With the proliferation of available large labeled data sets for training neural networks, with the increasing computing power of GPUs, and advanced regularization techniques, neural networks are becoming incredibly multi-layered, allowing you to generalize large amounts of invisible data. Layering is a huge advantage in the performance of neural networks [1-3].

It is well known that the brain's ability to recognize complex visual patterns or identify a speaker in a noisy environment is the result of several consecutive processing steps and a variety of learning mechanisms that are also built into SNN with deep learning [8-11]. Compared to ANN and deep learning, SNN learning is at the earliest stages of development. An important feature of this type of system is the neural architecture. It is much better suited for processing space-time data, especially in online mode. The representation of data in time and space that SNNs possess allows such neural networks to perform calculations at the level of the human brain, as well as to understand brain activity in the spatiotemporal structure. It is very important to understand in the coming years how to train such neural networks to perform various tasks [1-7].

In case of looking SNN from an engineering point of view, it can also be found interesting tasks here. This type of neural network has a number of advantages in hardware implementation over conventional ANNs. The pulse bundle in the SNNs is scattered over time, each of them contains a huge amount of information, which can significantly reduce power consumption. Thus, you can create a hardware platform with moderate resource intensity indicators that adapts its work to pulse activity.

It is worth noting that SNNs inspired by the biological example, in principle, recognize images much better and faster than conventional ANNs [12]. Moreover, SNNs allow the use of training methods that depend on the time of occurrence of impulses between pairs of directly connected neurons, in which information for changing the weight of edges is available locally. This training method is very similar to what happens in many parts of the brain.

The resulting pulse bundle is represented as undifferentiated sums of Delta functions. Accordingly, it is difficult to apply derivative-based optimization methods to SNN training, although various types of approximate derivatives have recently been actively studied. Despite the fact that in theory pulse neural networks have equivalent computing power in turing [13], it is still problematic to train SNNs, especially those with a multi-layered architecture. In many existing pulse neural networks, only one layer can be trained. If it is possible to provide pulse systems with multi-layer training, then productivity in various tasks will increase tenfold.

The architecture of pulsed neural networks consists of pulsed neurons and interconnected synapses. Pulse bundle in neural networks of impulse neurons propagate through synaptic connections. A synapse can be either excitatory, when the membrane potential of a neuron increases after receiving a signal, or slowing down. The amount of rib weight can change as a result of training. Deep learning of multilayer SNNs is a real mystery, since the undifferentiation of pulse bundle does not allow the use of popular methods, such as the backpropagation method (Fig. 2) [14-19].

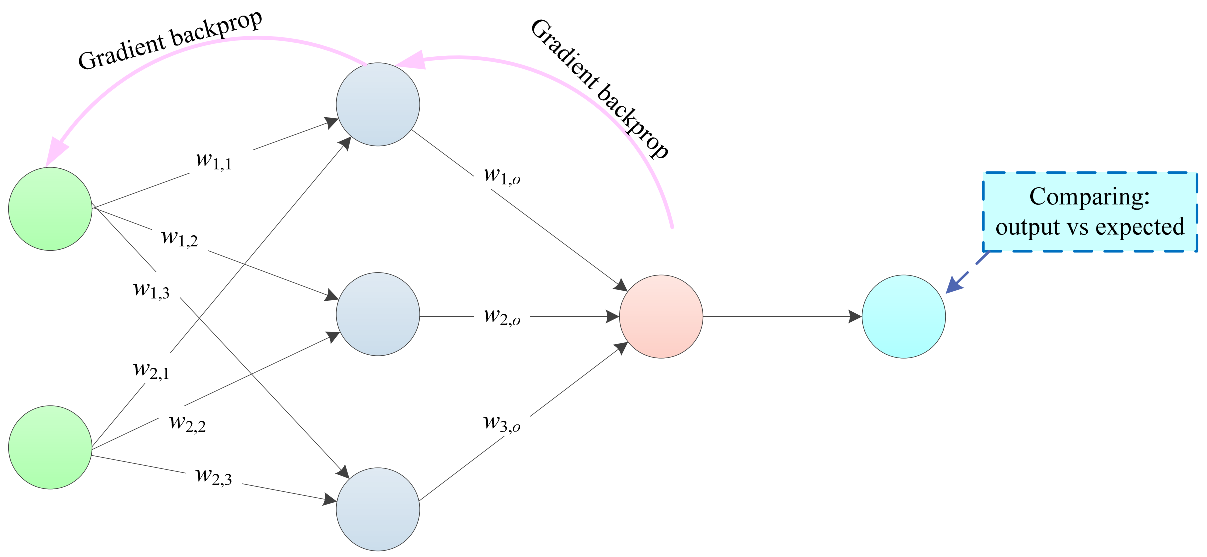


Figure 2: Using the backpropagation method for correcting weights coefficients

2. Related Works

As mentioned earlier, the training of all kinds of artificial neural networks occurs by adjusting scalar synaptic weights. In SNNs, can be used training methods that are close to those used by the brain. Scientists have identified many variants of this training method, but all of them fall under the general term dendritic plasticity (STDP) [20]. A key feature of dendritic plasticity is that the weight of the rib connecting pre- and postsynaptic neurons is regulated with their pulse time in the range of approximately tens of milliseconds in duration.

2.1. Method SpikeProp

This is the first method of training SNNs by error propagation [21], [22]. The cost function takes into account the fluctuation period, and thus this method can classify nonlinearly separated data for a time-encoded XOR problem using a 3-level architecture. The main decision at the stage of development of the method was the choice of the Gerstner neuron model (so-called, SRM) [23]. Using this model, the question of taking derivatives at the oscillation output was circumvented, since the required result was directly modeled as a continuous value. One of the limitations of this method is that each original unit was forced to generate exactly one oscillation. In addition, the values of continuous variables, such as in XOR problems, had to be encoded as delays between fluctuations, which can be quite long.

2.2. Method Remote supervised learning (ReSuMe)

This training method consists of a single oscillating neuron, which receives vibrations from many other oscillating presynaptic neurons [24]. The goal is to train the synapse to trigger a postsynaptic neuron to generate oscillation waves with the desired oscillation period. ReSuMe adapted the delta rule used for non-pulse linearized units to SNN. In the delta rule, the weights change proportionally:

$$\Delta\omega = \left(y^{d} - y^{real}\right)x = y^{d}x - y^{real}x, \qquad (1)$$

where x is presynaptic input;

y^d is desired output value;

y^{real} is real value at the output.

By reformulating the equation above, you can get the sum of STDP s anti- STDP:

$$\Delta\omega = \Delta\omega^{STDP}\left(S^{in}, S^{d}\right) + \Delta\omega^{\alpha STDP}\left(S^{in}, S^{real}\right). \qquad (2)$$

The above $\Delta\omega^{STDP}$ is a function of correlation between presynaptic and desired oscillation periods, where $\Delta\omega^{\alpha STDP}$ is depends on presynaptic and real oscillation periods. Since this method uses a correlation between a set of presynaptic neurons and a wavering neuron, there is no physical connection between them. This is why this method is called remote.

2.3. Method Chronotron

This method was developed based on the Temporal method [25], which could train individual neurons to recognize the encoding at the exact time of arrival of the oscillation. A limitation of the Temporal method was the ability to output 0 or 1 at the output during the selected time interval. Because of this, it was impossible to encode information about the time of arrival of the oscillation in the output data. The creation of Chronotron was influenced by the success of SpikeProp and its successors. The idea of Chronotron was to use more complex units of distance measurement: VP [26] between two vibrations for the training. They adapted the VP distance so that it was piecewise differentiated and suitable as a function of cost to perform gradient descent with respect to weights.

2.4. Problems of existing methods

Among the existing methods, there are several main disadvantages. First, each channel needs a separate quick convergence scheme. That is, there is a risk of multiplying derivatives on each channel, which in the future can impose large computational needs [21-26].

Secondly, it becomes quite difficult to get a large number of channels due to pulse distortions that occur when connecting matching circuits to coaxial cables [21-26].

But the key problem is always the question of topology (the structure of such a network). There are many approaches that apply reverse propagation to SNNs. But such methods cannot use traditional reverse propagation and change gradient updates or SNNs themselves to overcome this complexity. These types of methods often impose architectural constraints on the resulting SNNs, for example, requiring a feedforward connections SNN architecture. Therefore, there are many SNN topologies with desired properties, such as repeatability, that cannot be optimized using these methods [21-26].

That is why it is an urgent task to develop evolutionary methods for the full synthesis of the SNN model. When implementing the methods of the evolutionary approach, SNN population assessment will allow us to further use the processes of stochastic variation and selection to create the next population or control these processes. During successive iterations of mutations and selection of individual networks with increasing fitness, which can be broadly defined as accuracy in the classification problem or maximum reward from the environment, the new solutions will be qualitatively different from the previous ones. One of the significant advantages of evolutionary approaches is their flexibility. As for SNNs, evolutionary optimization can potentially affect any network topology, as well as optimize any network parameter.

3. Proposed method

In general, the method proposed in this paper is similar to a modified genetic algorithm (MGA) for synthesizing ordinary ANNs [27]. So, at the beginning, it should be noted that direct encoding of the structure of individuals is used to perform synthesis, each of which is a separate neural network, but the basis of such encoding is inter-neural connections. This makes it possible to track the origin of each parameter (node or connection) in the genome in more detail. In the subsequent stages of the method, this greatly simplifies the processes of crossing and mutation, allowing you to build and change genetic information about individuals [27].

The main innovation is the use of certain template mechanisms, namely neural patterns (NP) based on Composite Pattern Producing Networks (CPPN) technology [28]. This mechanism provides abstraction of the processes of natural evolution [28]. NPs will consist of neurons with various activation functions, including periodic functions such as sinusoidal and symmetric functions such as absolute value. The basic idea is based on the fact that a CPPN-based NP with multiple entry points can determine the relationship between a pair of points. A fixed set of neurons is introduced as a specific substrate to the NP, which further simplifies neurosynthesis using MGA.

In general, at the beginning of evolution, we obtain a population with NPs: $P = \{Ind_1, Ind_2, Ind_3, ..., Ind_n\} = \{NP_1, NP_2, NP_3, ..., NP_n\}$, that determine the connection patterns of the resulting SNNs $P = \{NP_1, NP_2, NP_3, ..., NP_n\} = \{SNN_1, SNN_2, SNN_3, ..., SNN_n\}$. Each NP is defined by encoded genetic information with the definition of its internal weights and activation functions. However, it is important to note that each individual neuron in an NP can have any activation function from a specific set (sin, tanh, gauss, relu, identity): $NP = SNN = \{struct, param\} = \{Node, Connections, weights, Function\}$ [29-31]. To decipher SNN from NP, the coordinates for a pair of neurons are passed to the network that creates the NP. The network output determines the weight and delay of the connection between two neurons in the dive neural network. This process is repeated for each pair of neurons to build a complete network.

After preparing the population, it is possible to perform further synthesis. The main steps for synthesizing and decoding SNN from NP are shown in Fig. 3.

However, several nuances of this synthesis should be noted:

- the positions of SNN neurons, in the form of NP, are pre-recorded during the evolutionary search for each individual. That is, any mutation and modification is possible only in the form of a new individual;
- each solution is decoded and evaluated to complete a given task (classification or management). Therefore, the suitability of the solution is determined by the performance of the generated SNN for the specified task;
- speciation is used to protect innovations in new solutions and can stop if they are not available.

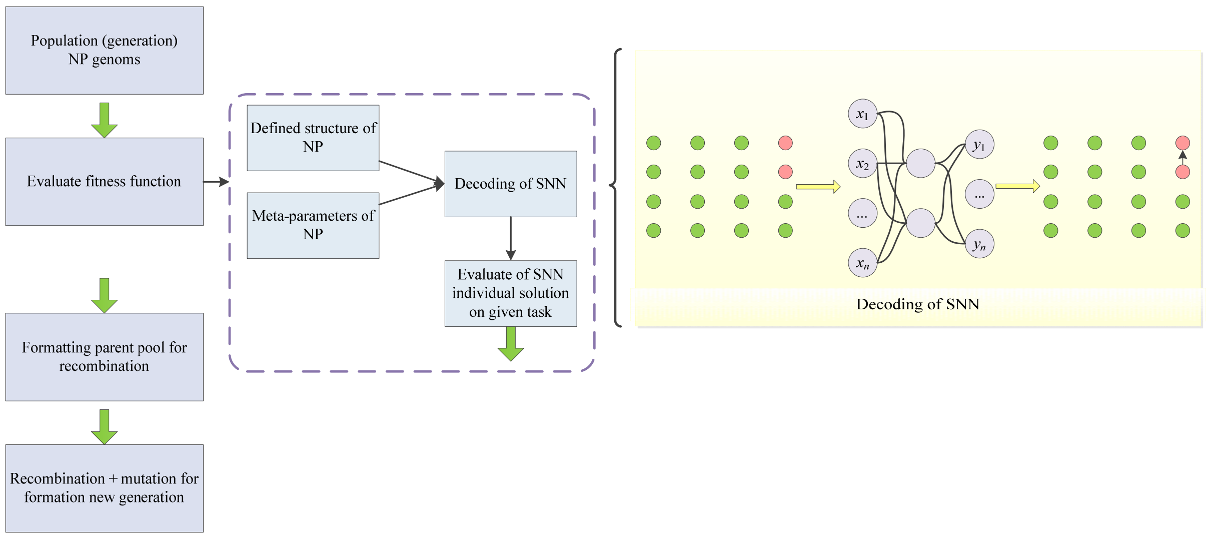

Population (generation) NP genoms
Evaluate fitness function
Formatting parent pool for recombination
Recombination + mutation for formation new generation
Defined structure of NP
Meta-parameters of NP
Decoding of SNN
Evaluate of SNN individual solution on given task
x_1
x_2
...
x_n
y_1
...
y_n
Decoding of SNN

Figure 3: The general scheme of the method

4. Experimental research

To study the results of the method, the task of medical diagnostics was chosen. A data set was selected as input about characteristics of patients with pneumonia, which was recently presented by authors M.-A. Kim, J. Seok Park,C. W. Lee, and W.-I. Choi [32]. Total sample size: 77490 values. Table 1 shows the characteristics of the data set.

For this task, the development of neuromodels will make it much easier to determine the further diagnosis of a person after collecting data on their well-being. Given that pneumonia is one of the most important signs and complications of COVID-19 [33], [34], after additional training on advanced data, this model can be used to diagnose patients or predict the further development of disease dynamics.

The effectiveness of SNN models will be compared with neuromodels based on recurrent neural networks (RNNs) (Fig. 2.a)) and deep neural networks (DNNs) (Fig. 2.b)) [35].

RNN is a type of ANN and is used in natural language processing and speech recognition applications. The RNN model is designed to recognize consistent data characteristics and then use templates to predict the future scenario [35]. A DNN is a neural network with a certain level of complexity, a neural network with more than two layers. DNNs use sophisticated mathematical modeling to process data in complex ways [35].

For the experiments, a workstation with the following characteristics was used: Intel Core i5-12600 central processor (3.30-4.80 GHz (Intel Turbo Boost 2.0), 6 cores and 12 threads), 16 Gb RAM (dual-channel mode), SK hynix SC308 128 GB SSD (M.2), the Python programming language.

Table 2 shows the results of synthesis methods with different typologies of neural networks.

Table 1

General characteristics of the data set

Total number of values	77490	Number of attributes	54
The type of the data	Numeric (after consideration of)	Number of instances'	1435

Table 2

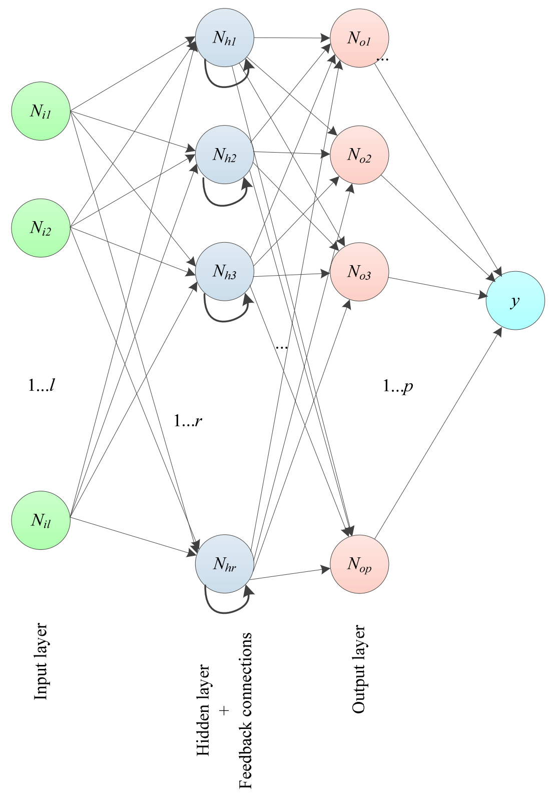
General results on the data set

Methodof synthesis	Synthesis Time, s	Error at the training sample	Error at the test sample
MGA SNN	9352	0.01	0.137
MGA RNN	6793	0.018	0.148
MGA DNN	7625	0.014	0.138

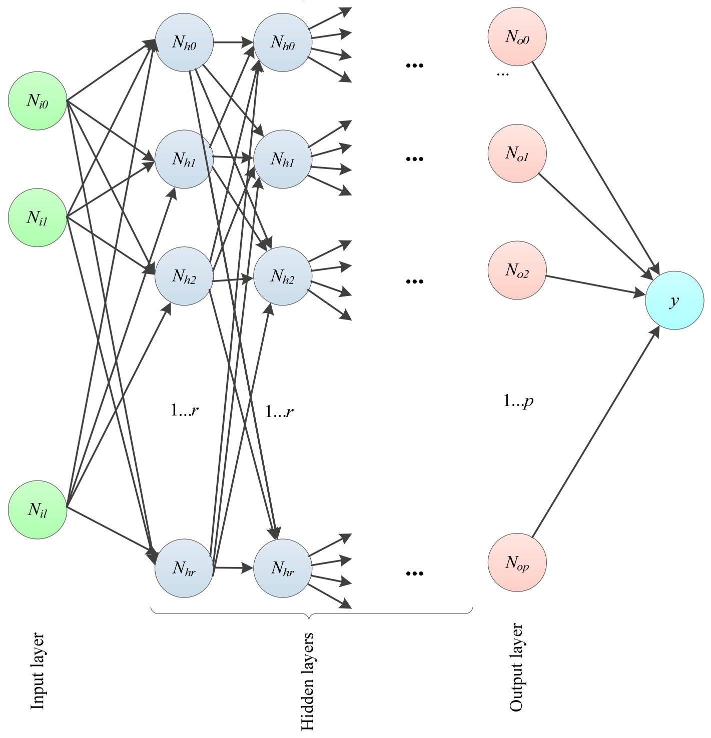
5. Discussions of results

From the analysis of the obtained experimental results, we can conclude that the proposal to use neuroevolution methods for SNNs synthesis shows good results for its further deployment and use. After all, for networks of this class, the issue of a complex and sometimes not fully clear structure is always relevant. The use of neuroevolutionary approaches with NP encoding and decoding greatly facilitates this task, because in this case structural synthesis is reduced to well-known approaches to neuroevolutionary synthesis using TWEANNs. Parametric synthesis, on the other hand, is fine-tuning at exclusively defined stages without having to iteratively calculate fitness functions or its derivatives over and over again.

On the other hand, we can conclude that the practical use of SNN-class neural networks is still extremely promising, because the results obtained for the accuracy of work cannot fully demonstrate the feasibility of significant time costs. Thus, the values of errors in both cases (training and testing) do not differ in significance, which could cover the time spent on the synthesis of such networks and their subsequent calculations. After all, it is worth noting that in the absence of specialized equipment for emulating the operation of SNN, the execution of this process on a regular workstation is also accompanied by time overlays.



a)



b)

Figure 4: The comparing of RNN and DNN topologies

It is also worth analyzing the work with complex topologies of classical ANNs separately. Thus, the differences in synthesis time are explained by a simpler calculation and correction of recurrent connections in contrast to the calculation of nodes and weights of interneuronal connections in a set of hidden layers. And the analysis of the accuracy of the synthesized solutions based on training and test data shows that the difference in accuracy is not critical and is acceptable within the task. That is why we can talk about a specific indication, about the use of a particular ANN topology, depending on which of the resources is more important: the time of synthesis and operation of the neuromodel or its accuracy.

6. Conclusion

This paper presents an approach to neuroevolution synthesis of complex ANN topologies, namely SNN. The aim of the work was to demonstrate the feasibility and effectiveness of the method. The experiments shown demonstrate the strengths and weaknesses of the approach, as well as identify ways for future research.

One of the main problems that is tracked from the very beginning is the sensitivity to mutations of rigidly defined neuromodel structures. Because mutation of even one structural unit (for example, between neural connections) leads to the creation of a completely new individual, which can sometimes be perceived as undesirable computational difficulties.

However, the main theoretical advantage of this approach to SNN synthesis is the possibility of controlled network development to a large size. In this work, only networks with fewer than a hundred neurons were tested. Therefore, network scaling properties are a further area of research.

Another important research perspective is the introduction and use of methods for indirect encoding of NP and ANN individuals in general during synthesis. Such approaches can help to study in more detail the many variants of relationships between unique SNN characteristics. For example, different input encoding strategies can lead to variability in the configuration of neurons for NP.

Future work may use more of the relationships between neuroevolution and SNNs and extend this method to more complex tasks.

7. Acknowledgements

The work was carried out with the support of the state budget research projects of the state budget of the National University "Zaporozhzhia Polytechnic" “Development of methods and tools for analysis and prediction of dynamic behavior of nonlinear objects” (state registration number 0121U107499) and “Intelligent methods and tools for diagnosing and predicting the state of complex objects” (state registration number 0122U000972).

8. References

[1] D. Ping, The Machine Learning Solutions Architect Handbook: Create machine learning platforms to run solutions in an enterprise setting, Packt Publishing, Birmingham, 2022.
[2] A. Burkov, Machine Learning Engineering, True Positive Inc., Quebec City, 2020.
[3] A. Park, Machine Learning: 2 Books in 1 – The Complete Guide for Beginners to Master Neural Networks, Artificial Intelligence, and Data Science with Python, 2nd. ed., Independently published, 2022.
[4] S. Alkhalifa, Machine Learning in Biotechnology and Life Sciences: Build machine learning models using Python and deploy them on the cloud, Packt Publishing, Birmingham, 2022.
[5] C.C. Aggarwal, Neural Networks and Deep Learning: A Textbook, Springer, Berlin, 2018.
[6] P.R. Hiesinger, The Self-Assembling Brain: How Neural Networks Grow Smarter, Princeton University Press, Princeton, 2021.
[7] D. Graupe, Principles of Artificial Neural Networks: Basic Designs to Deep Learning (4th Edition) (Advanced Circuits and Systems), 4th ed., WSPC, Singapore, 2019.

[8] A.Hodgkin, A. Huxley, A quantitative description of membrane current and its application to conduction and excitation in nerve, J. Physiol, vol. 117(4), (1952) 500-544. doi: 10.1113/jphysiol.1952.sp004764

[9] R. Jolivet, J. Timothy, W. Gerstner, The Spike Response Model: A Framework to Predict Neuronal Spike Trains, in: Proceedings of the 2003 joint international conference on Artificial neural networks and neural information processing, ICANN/ICONIP'03, ACM Digital Library, 2003, pp. 846-853. doi: 10.1007/3-540-44989-2_101

[10] E. Izhikevich, Simple model of spiking neurons, Transactions of neural networks, vol. 14, (2003) 1569-1572. doi: 10.1109/TNN.2003.820440

[11] A. Delorme et al., SpikeNET: A simulator for modeling large networks of integrate and fire neurons, Neurocomputing, vol. 26(7), (1999) 989-996. doi: 10.1016/S0925-2312(99)00095-8

[12] 14 . M. Pfeiffer, T. Pfeil, Deep learning with spiking neurons: Opportunities and challenges, Frontiers in Neuroscience, vol. 12: 774 (2018). doi: 10.3389/fnins.2018.00774

[13] 13 W. Maass, Lower bounds for the computational power of networks of spiking neurons, Neural Computation, vol. 8, (1996) 1-40.

[14] J.A.J. Alsayaydeh, A. Aziz, A.I.A. Rahman, S.N.S. Salim, M. Zainon, Z.A. Baharudin, M.I. Abbasi, A.W.Y. Khang, Development of programmable home security using GSM system for early prevention, vol. 16(1) (2021) 88-97.

[15] J.A.J. Alsayaydeh, W.A.Y. Khang, W.A. Indra, V. Shkarupylo, J. Jayasundar, Development of smart dustbin by using apps, ARPN Journal of Engineering and Applied Sciences, vol. 14(21) (2019) 3703-3711.

[16] J.A.J. Alsayaydeh, W.A.Y. Khang, W.A. Indra, J.B. Pusppanathan, V. Shkarupylo, A.K.M. Zakir Hossain, S. Saravanan, Development of vehicle door security using smart tag and fingerprint system, ARPN Journal of Engineering and Applied Sciences, vol. 9(1) (2019) 3108-3114.

[17] J.A.J. Alsayaydeh, W.A.Y. Khang, A.K.M.Z Hossain, V. Shkarupylo, J. Pusppanathan, The experimental studies of the automatic control methods of magnetic separators performance by magnetic product, ARPN Journal of Engineering and Applied Sciences, vol. 15(7) (2020) 922-927.

[18] J.A.J. Alsayaydeh, W.A. Indra, W.A.Y. Khang, V. Shkarupylo, D.A.P.P. Jkatisan, Development of vehicle ignition using fingerprint, ARPN Journal of Engineering and Applied Sciences, vol. 14(23) (2019) 4045-4053.

[19] J.A. Alsayaydeh, M. Nj, S.N. Syed, A.W. Yoon, W.A. Indra, V. Shkarupylo, C. Pellipus, Homes appliances control using bluetooth, ARPN Journal of Engineering and Applied Sciences, vol. 14(19) (2019) 3344-3357.

[20] S. Tazerart, D. E. Mitchell, S. Miranda-Rottmann, R. Araya, A spike-timing-dependent plasticity rule for dendritic spines, Nature Communications, vol. 11 (2020). doi: 10.1038/s41467-020-17861-7

[21] R.V. Florian, Supervised Learning in Spiking Neural Networks. In: Seel N.M. (eds) Encyclopedia of the Sciences of Learning. Springer, Boston, MA, 2012. doi: 10.1007/978-1-4419-1428-6_1714

[22] S. M. Bohte, Error-backpropagation in temporally encoded networks of spiking neurons, in: Proceedings of the Artificial Neural Network and Machine Learning, ICANN 2011, ACM Digital Library, 2011, pp. 60-68.

[23] W. Gerstner, W. M. Kistler, Spiking neuron models: Single neurons, populations, plasticity, Cambridge University Press, Cambridge, 2002.

[24] F. Ponulak, A. Kasinski, Supervised learning in spiking neural networks with ReSuMe: Sequence learning, classification, and spike shifting, Neural Computation, vol. 22, (2009) pp. 467-510. doi: 10.1162/neco.2009.11-08-901

[25] R. Gütig, H. Sompolinsky, The tempotron: A neuron that learns spike timing-based decisions, Nature Neuroscience, vol. 9, (2006) 420-428. doi: 10.1038/nn1643

[26] T. Serre, Hierarchical models of the visual system, Encyclopedia of Computation Neuroscience, Springer, New York, NY, 2014. doi: 10.1007/978-1-4614-7320-6_345-1

[27] S. Leoshchenko, A. Oliinyk, S. Subbotin, N. Gorobii, T. Zaiko, Synthesis of artificial neural networks using a modified genetic algorithm, in: Proceedings of the 1st International Workshop on Informatics & Data-Driven Medicine, IDDM 2018, CEUR-WS, 2018, pp. 1-13

[28] D. Elbrecht, C. Schuman, Neuroevolution of Spiking Neural Networks Using Compositional Pattern Producing Networks, in: Proceedings of the ICONS 2020: International Conference on Neuromorphic Systems 2020, ICONS 2020, ACM Digital Library, 2020, pp. 1-5. doi: 10.1145/3407197.3407198

[29] A.Oliinyk, S. Skrupsky, S. Subbotin, Experimental research and analysis of complexity of parallel method for production rules extraction. Automatic Control and Computer Sciences, 52 (2) (2018) 89-99. doi: 10.3103/S0146411618020062

[30] S. Leoshchenko, A. Oliinyk, S. Skrupsky, S. Subbotin, T. Zaiko, Parallel Method of Neural Network Synthesis Based on a Modified Genetic Algorithm Application, in: Proceedings of the 8th International Conference on Mathematics. Information Technologies. Education, MoMLeT&DS-2019, CEUR-WS, 2019, pp. 11–23.

[31] A.Oliinyk, S. Leoshchenko, V. Lovkin, S. Subbotin, T. Zaiko, Parallel data reduction method for complex technical objects and processes, in: Proceedings of the 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT-2018, IEEE, 2018, pp. 496-501. doi: 10.1109/DESSERT.2018.8409184

[32] M.-A. Kim, J. Seok Park,C. W. Lee, W.-I. Choi, Pneumonia severity index in viral community acquired pneumonia in adults, PLoS One, vol. 14(3) (2019). doi: 10.1371/journal.pone.0210102

[33] G. Raghu, K. C Wilson, COVID-19 interstitial pneumonia: monitoring the clinical course in survivors, The Lancet Respiratory Medicine, vol. 8(9) (2020) 839-842. doi: 10.1016/S2213-2600(20)30349-0

[34] A. Hani, N.H. Trieu, I. Saab, S. Dangeard, S. Bennani, G.Chassagnon, M.-P. Revel, COVID-19 pneumonia: A review of typical CT findings and differential diagnosis, Diagnostic and Interventional Imaging, vol. 101(5) (2020), 263-268. doi: 10.1016/j.diii.2020.03.014

[35] S. Leoshchenko, A. Oliinyk, S. Subbotin, T. Zaiko, A Using modern architectures of recurrent neural networks for technical diagnosis of complex systems. Proceedings of the 5th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PICST2018), IEEE, Kharkiv, Ukraine, 2018, pp. 411–416. doi: 10.1109/INFOCOMMST.2018.8632015