6. THE USE OF NEURAL NETWORKS IN PSEUDO-RANDOM NUMBER GENERATORS DEVELOPMENT

Boltak S.V.

Belarusian State University of Informatics and Radioelectronics
Minsk, Republic of Belarus

Liakh Y.V. – Senior Lecturer

The information about using different types of neural networks for developing pseudo-random number generators is presented in this paper.

Random number generators are widely used in computer security issues and cryptography. Most cryptographic procedures require random numbers: generating the key, authentication, and identification. One of the major problems in cryptography today is developing a pseudo-random number generator (PRNG) with acceptable quality in terms of randomness. There are many methods that can help generate pseudo-random numbers.

Using neural networks for generating pseudo-random numbers is a new approach. The idea was put forward by Donald Knuth. According to his theory, the complexity of the generation algorithm is not connected to the quality of the resulting sequence [1]. A group of researchers from Russia conducted a study and claimed that Knuth's idea was fully supported by the example of the Xorshift generator, which despite its simplicity showed the results at the level of a much more complex Mersenne twister [2]. It can be assumed that using neural networks as pseudo-random number generators will be less productive, as it will require significantly more resources to initialise the structure of neurons, as well as time for training. However, some features of neural networks like non-linearity and unpredictability make them suitable for generating random numbers.

Different types of neural networks and combinations of them have been used as a random number generator component in cryptosystems. Generally, neural networks take certain data as input and generate the desired result as output. In this sense, neural networks are predictable. However, if over-fitting occurs, a neural network will show unpredictable result for input data which was not used for training. This property is widely used in most pseudo-random number generators based on neural networks [3]. So neural networks have unpredictable behaviour under certain conditions.

Recurrent neural networks, also known as RNN, are more suitable for generating random numbers. These networks are called feedback networks. In such networks neurons exchange information with each other. Thus, «memory» is implemented in the network. The examples of recurrent networks for generating pseudo-random numbers are Hopfield network, spiking neural networks (SNN), multi-layer perceptron neural networks (MLPNN).

The main idea when using Hopfield network as PRNG is to find a way for the network to never converge. The ability to converge in such networks is strongly related to the network architecture, the network initial condition, the updating rule mode. Thus, by changing these parameters, it is possible to achieve network non-convergence.

In multi-layer perceptron the over-fitting property is the basis of pseudo-random number generators. These random-number generators have a powerful ability to generate real random numbers. Combinations of such networks with a hash function represent better result with randomness test.

Spiking neural networks are much more complicated networks. They are called highly connected recurrent neural networks. In such networks the flow of information can spread both from and to neurons. Spiking neural networks emulate such a property of the brain as plasticity. Spiking neural networks have three types of plasticity – spike-timing dependent plasticity (STDP), anti-spike-timing dependent plasticity (Anti-STDP), intrinsic plasticity (IP). Neural networks with Anti-STDP are well-suited for a random number generator.

In conclusion, it can be said that neural networks are suitable for random number generators because of their features such as complexity, parallel ability, non-linearity, unpredictability. Also, different combinations of neural networks with other technologies have been used as a random number generator component in cryptosystems.

References:

1. Knuth, D. E. The art of computer programming, Volume 2, 3rd Ed.: Semi-numerical algorithm / D. E. Knuth – Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

2. Gevorkyan, M. N. Pseudo-random number generator based on neural network / V. N. Gevorkyan, A. V. Demidova, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastianov, I. M. Gostev, – GRID 2018, p.568-572, Dubna, Moscow region, Russia, September 10 - 14, 2018.

3. Karras, D. A. Overfitting in multilayer perceptron as a mechanism for (pseudo) random number generation in the design of secure electronic commerce systems, in Information Systems for Enhanced Public Safety and Security / D. A. Karras, V. Zorkadis – IEEE/AFCEA, EUROCOM, Munich, 2000, pp. 345 - 349.

4. Tirdad, K. Developing pseudo random number generator based on neural networks and neurofuzzy systems, dissertation – Ryerson University, 2010.