2020 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)

Selecting Architecture and Parameters of Deep Neural Networks for Computer Attack Classification

O.S. Amosov
Laboratory of Intellectual Control Systems and Modelling,V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences,
Moscow, Russia
osa18@yandex.ru

S.G. Amosova
Laboratory of Cyber-physical Systems,
V.A. Trapeznikov Institute of Control Sciences of Russian Academy of Sciences,
Moscow, Russia
amosovasg@yandex.ru

D.S. Magola
Network and system administration Department, Komsomolsk-on-Amur state university,
Komsomolsk-on-Amur, Russia,
dmagola@list.ru

***Abstract*— The article states a problem of a multiclass network classification of computer attacks. To solve it, the authors consider an option of applying deep neural networks. For this research, the authors selected the architecture of a deep neural network based on the strategy combining a set of convolution and recurrent LSTM layers. The research suggests optimizing the neural network parameters on the basis of a genetic algorithm. The authors present the simulation results that show an opportunity to solve the task of network classification in real time.**

Keywords—deep neural network; architecture; parameter optimization; convolutional layer; recurrent layer, long short-term memory; network classification; computer attacks.

I. INTRODUCTION

Relevance. At present the application of deep neural networks (DNN) showed positive results for the recognition and classification of persons, technical objects, dynamic situations, information safety, time series forecast [1-4]. However, no rigorous theory of design and structural & parametric DNN optimization exists. But there are particular approaches existing for smart systems design, including those for neural networks [5-9].

One of DNN building strategies is the combination of speed and lightness of convolution networks with the sensitivity towards the order of recurrent networks: convolution network use for the preliminary processing of data before transferring it into the recurrent network [10, 11].

Such strategy was selected for choosing the architecture of deep neural networks to classify computer attacks while the genetic algorithm is used for parameter optimization.

II. NETWORK CLASSIFICATION PROBLEM STATEMENT

Following [12], provide a statement of multiclass classification problem. The object shall be hereinafter understood as network attack or anomaly.

Given: a set Ω, storing the description of objects $\omega \in \Omega$ preset by the properties $x_i, i = \overline{1,n}$, the combination of which for the object ω is presented by vector descriptions $\mathbf{x} = \mathbf{\Phi}(\omega) = (x_1(\omega), x_2(\omega), ..., x_n(\omega))^T$; a set of classes $\mathbf{B} = \{\beta_1, \beta_2, ..., \beta_c\}$, c –number of classes; apriori information presented by a training set $\mathbf{D} = \{(\mathbf{x}^j, \beta^j)\}$, $j = \overline{1,L}$, preset by the Table each j-th line of which contains the object vector description $\mathbf{x} = \mathbf{\Phi}(\omega)$ and the class mark β_k, $k = \overline{1,c}$. Note that the training set is characterized by unknown display $^*\mathbf{F} := \Omega \to \mathbf{B}$.

By the incoming fragments $\mathbf{I}_t$ of a continuous network traffic $\mathbf{V} = (\mathbf{I}_1, ..., \mathbf{I}_t, ..., \mathbf{I}_\tau)$ and apriori information, set by the training set $\mathbf{D} = \{(\mathbf{x}^j, \beta^j)\}$, $j = \overline{1,L}$ for the deep leaning of a neural network with a teacher, it is required to resolve the problem of object recognition: detecting the objects ω in the form of the evaluation of properties $\tilde{\mathbf{x}}$ with the help of neural networks implementing the $\mathbf{F}_1 : \mathbf{I}_t \to \tilde{\mathbf{x}}$ detection and classifying them with the use of detecting $\mathbf{F}_2 : \tilde{\mathbf{x}} \to \beta_k$, $k = \overline{1,c}$ in compliance with a set criterion $\mathrm{P}(\tilde{\mathbf{x}})$ minimizing the probability of classification error.

Therefore, it is necessary to find the display of $\mathbf{F} : \mathbf{I}_t \to \beta_k$, $k = \overline{1,c}$, at which $\mathbf{F}$–is a set of functions and neural network algorithms $\mathbf{f}_i$, $i = \overline{1,N_f}$.

III. SOLUTION OF THE CLASSIFICATION PROBLEM STATEMENT

The classification problem should be resolved by the following DNN-based algorithm:

1. Select the fragment $\mathbf{I}_t$ of the size $w^{\mathbf{I}_t} \times h^{\mathbf{I}_t}$ out of a continuous network traffic $\mathbf{V} = (\mathbf{I}_1, ..., \mathbf{I}_t, ..., \mathbf{I}_\tau)$, where t – number of the current fragment.

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

2. The transformation of a fragment $\mathbf{I}_t$ with the size $w^{\mathbf{I}_t} \times h^{\mathbf{I}_t}$ into a fragment $\mathbf{I}_t^{'}$ with a size $k^{\mathbf{I}_t^{'}} \times s^{\mathbf{I}_t^{'}} \times h^{\mathbf{I}_t}$ where $k^{\mathbf{I}_t^{'}} \times s^{\mathbf{I}_t^{'}}$ contains all data $w^{\mathbf{I}_t}$. In other words, the initial two-dimensional tensor is transformed into a three-dimensional.

3. The extraction of informative features **R** with the use of a pre-trained deep neural network (NN) $\mathbf{\Phi}^{NN} : \mathbf{R} \to \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ – the data of the fragment $\mathbf{I}_t$ transferred into the space of NN features. It should be mentioned the authors consider a deep neural network with convolution and/or recurrent layers: one of the architectures which will be in detail discussed in the section of DNN architecture selection in Figure 2.

4. Relating the feature vector to one of the classes $\mathbf{f}_i : \tilde{\mathbf{x}} \to \mathbf{p}_{\tilde{\mathbf{x}}}$, where $\mathbf{p}_{\tilde{\mathbf{x}}}$ – vector containing the classification probability.

A. DNN architecture selection

Convolution neural networks are particularly suitable for resolving the problems of image recognition due to their capability of parameter convolution, feature extraction from local input samples and obtaining efficient and modular data representation. However, all these features can be applied for the processing of time sequences as time can be considered as a spatial dimension like a two-dimensional image height or width [13]. Unidimensional convolution layers can recognize local samples in the sequence [11]. This capability of 1D convolutional networks can be used for the processing of time series of network traffic to classify the network statuses, including the definition of its normal status and statuses with abnormal activities.

However, convolutional networks are not sensitive towards the sequence of time intervals as they have no memory in contrast to recurrent networks [11]. The "classical" layers of a recurrent network process the sequence by means of iterating over its elements and preserving the statuses obtained at the processing of previous elements. In many cases, the processing of the next sequence element requires the information not on the previous element, but on the element/elements from more distant past. "Classical" recurrent networks can partially solve this problem also but this requires the increase in the number of recurrent layers; here the problem of gradient damping occurs: as the number of layers grow the network is gradually becoming not capable of learning [11]. To solve the problem, other approaches are also applied: LSTM layer (Long Short-Term Memory) on the basis of the algorithm of the a long short-term memory [2, 14] and the GRU layer (Gated Recurrent Unit) [14, 15].

The strategy some authors [10] stick to is the combination of speed and lightness of convolution networks with the sensitivity towards the order of recurrent networks: convolution network use for the preliminary processing of data before transferring it into the recurrent network. This technique may be justified in the case when the sequence includes several thousand intervals or more, i.e., in the case that cannot be processed by a convoltional network. A convolutional network will transform a long sequence into a shorter one consisting of high-level features after which they will be delivered to the recurrent network input (Fig. 1) [11]. The coordination of these layers is provided lower.

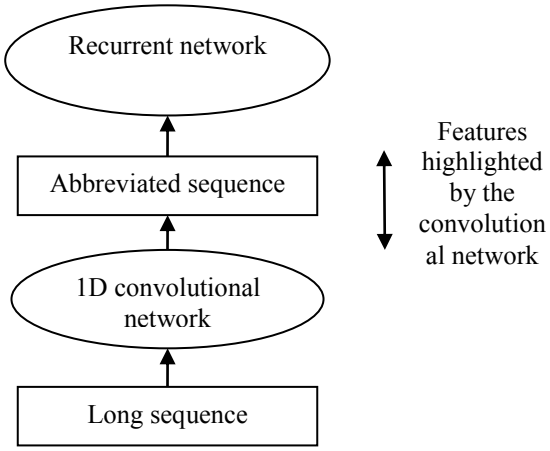

Fig. 1. The integration of 1D convolutional and recurrent networks

Select the following architecture given in Figure 2 as a DNN:

1) The network input, presented in the form of 3D tensor transformed from a two-dimensional tensor, comes to the input of the convolutional layer which transforms the sequence vector of the size $C \times N \times S$ into the sequence vector of the size $C \times (N * S)$. The layer outputs present a map of output features on the basis of the transformed input coming to the convolutional layer input as well as the minimum size of the processing package forwarded to the convolutional layer input.

2) Two sequential filter blocks (with the change of filters and filter number):

2a) The layer of convolution with the core **K** in the layer *l* with a linear function of activation σ^{linear}

$$\mathbf{h}_{conv}^{l} = \sigma^{linear}(\mathbf{h}^{l-1} \cdot \mathbf{K} + b_{conv}^{l}), \quad (1)$$

where $\mathbf{h}_{conv}^{l}$ – convolutional layer output, $\mathbf{h}^{l-1}$ – previous layer output, b_{conv}^{l} – shift coefficient.

2Б) Normalization layer:

$$\mathbf{h}_{bn}^{l} = a \cdot \frac{\mathbf{h}^{l-1} - \frac{1}{m}\sum_{i=1}^{m}\mathbf{h}^{l-1}}{\sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{h}^{l-1} - \frac{1}{m}\sum_{i=1}^{m}\mathbf{h}^{l-1}\right)^{2} + e}} + b, \quad (2)$$

where $\mathbf{h}^{l-1}$ – previous layer output, m – number of parameters processed by the previous layer, *b*– displacement factor, *a* – scale factor, *e*– constant value for numerical stability.

2в) Activation layer ReLU:

$$\mathbf{h}_{activ}^{l} = \sigma^{\mathrm{Re}LU}(\mathbf{h}^{l-1}). \quad (3)$$

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

Function ReLU – linear rectification or cutting-off of a negative part of a scalar value $\sigma^{ReLU}(\mathbf{z}) = \max(0, \mathbf{z})$.

2г) Pooling layer: taking the maximum (mean) value of the processed information block, i.e., generalizing information from the previous layer: $\mathbf{h}^l_{polling} = \max(\mathbf{h}^{l-1})$.

3) The convolutional layer conducts operations reverse to the convolutional layer: transformation of the sequence vector of the size $C \times (N * S)$ into the sequence of the size $C \times N \times S$.

4) Convolutional layer transforming spatial dimensions into one channel.

5) LSTM recurrent layer. The layer input receives the "shortened" sequence of high level features obtained from the previous convolution layer.

The LSTM network uses special mechanisms: forgetting filter (*f*), input (*i*) and output (*o*) filters. As opposed to the recurrent block that simply calculates the weighted total of input signals and uses a non-linear activation function each *j*-th LSTM block supports the SD card c_t^j at the point of time *t*. The output of the h_t^j LSTM block is calculated as follows:

$$h_t^j = o_t^j \tanh(c_t^j), \tag{4}$$

where tanh – designation of the hyperbolic tangent function, o_t^j – output filter that modulates the scope of memory and is calculated as follows:

$$o_t^j = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + b_o)^j, \tag{5}$$

where $\mathbf{x}$ – input sequence, σ – activation function, U – trainable parameter, b – displacement.

The memory cells c_t^j are renewed by means of partial forgetting of the existing memory and addition of new memory $\tilde{c}_t^j$:

$$\begin{aligned} c_t^j &= f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j, \\ \tilde{c}_t^j &= \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1})^j. \end{aligned} \tag{6}$$

The level of forgetting the existing memory is modulated by the forgetting filter f_t^j. The level of adding new content to the memory cell is modulated by the input filter i_t^j. The filters are calculated as follows:

$$\begin{aligned} f_t^j &= \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + b_f)^j, \\ i_t^j &= \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + b_i)^j. \end{aligned} \tag{7}$$

6) The noise elimination layer, or a hidden layer, in which all the previous layer elements are equal to 0 with the probability 0 - 100% (by default 50%). It is used for retraining prevention.

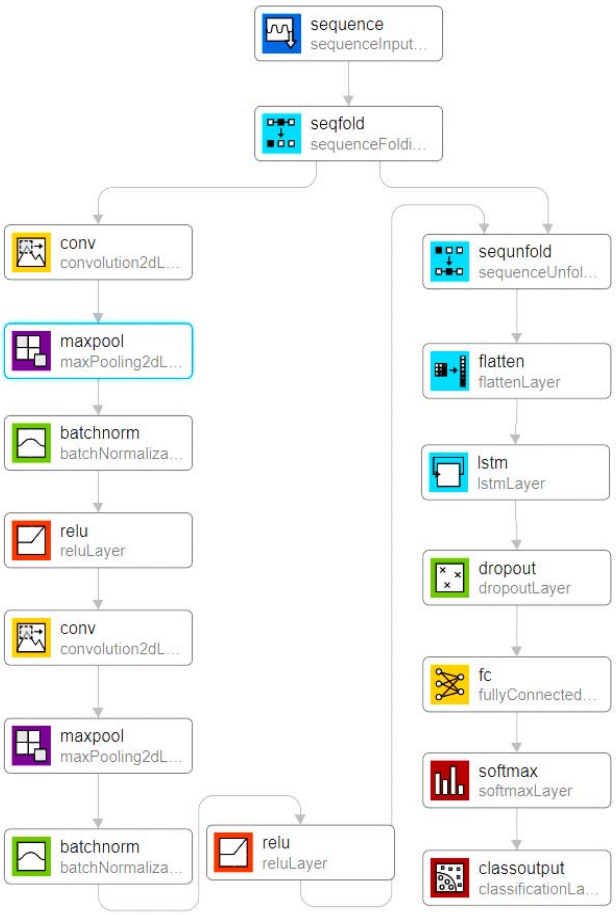

Fig. 2. Neural network architecture in the form of blocks by means of the tool Deep Network Designer of the MATLAB system

7) To forecast class marks, the network is calculated with a completely connected layer with the number of neurons equal to the number of classes, the layer *softmax*, that defines the probabilities of sequence belonging to the class and the classification output layer which sets the corresponding class marks in compliance with the output of the layer *softmax*. *Softmax* transforms the previous layer output into the new vector with the same size where each coordinate of the obtained vector will be a digit in the interval [0, 1] and be interpreted as a probability that a corresponding object belongs to the class while the coordinate total equals 1.

IV. Simulation Results

For experimental simulation, the authors used a generally accessible data base KDD [16]. The data base contains virtually 5 million (4 891 469) network status instances classified by 22 types on the basis of 41 features. All the features are not equivalent in terms of information. In compliance with [17] only 12 features contain 99% of information for classification, while in compliance with [18] to identify and classify 9 of 22 attacks, 29 features are sufficient, however, this circumstance was not taken into account at the obtaining the results provided below. The initial data base contains both numerical and symbolic data.

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

For processing all the symbolic features were transformed into numerical ones.

The attack base is divided into 4 categories:

DOS – denial of service: back, land, Neptune, pod, smurf, teardrop.

U2R – attacks intended for obtaining network administrator privileges by the authorized user: buffer_overflo, loadmodule, perl, rootkit.

R2L – attacks aimed at obtaining the access by a non-authorized user: ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster.

Probe – network port probing: ipsweep, nmap, portsweep, satan.

During the analysis of the number of references for each of attacks contained in the BD one can draw the conclusion that only the reference normal network statuses and 5 attacks can be used for the application of neural network technology for the training sample: ipsweep (12,481 – number of references), Neptune (1,072,917), portsweep (10,413), satan (15,892), smurf (2,807,886). To obtain the results describe below, the authors use three reference network statuses: normal status and 2 attacks: Neptune, smurf. This is explained by the fact that these very reference statuses make more than 99.2% of a total data base. Therefore, for the purpose of simulation the data base of three network statuses which contains the matrix *4,853,584 X 42* where each record should define the status of the computer network by the 41th feature while the last column contains the class number.

A. LTSM network

The authors consider various options of the combinations of the training and testing sample volume. Simulation was conducted in MATLAB 2019 at a PC with the following parameters Intel Core i7-4710HQ, 4 cores by 2.5 GHz, RAM 6 Gb, GPU GeForce GTX 850M, 64-digit operational system Windows. The authors used a standard LSTM network (Fig. 3). Simulation results are shown in Table 1.

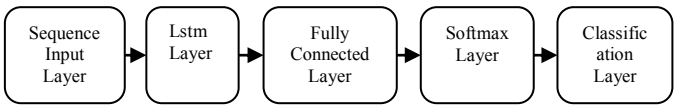

Fig. 3. LSTM network structure

TABLE I. LSTM NETWORK SIMULATION RESULTS.

No.	Training sample size	Test sample size	Transformed tensor size	Training sample precision, %	Test sample precision, %	Training time, min
1	3,000 x 41	3,000 x 41	150 x 20 x 41	100	100	0.7
2	12,000 x 41	3,000 x 41	600 x 20 x 41	100	100	1.5
3	120,000 x 41	30,000 x 41	6,000 x 20 x 41	100	100	16
4	1,200,000 x 41	300,000 x 41	60,000 x 20 x 41	100	99.29	75
5	300,000 x 41	75,000 x 41	600 x 500 x 41	100	100	1.05

The LTSM layer had 100 neurons for the experiments. To optimize machine costs and obtain the result for a shorter period of time, the information blocks from a two-dimensional tensor were transformed into three-dimensional. Without such transformation the training time increases by times; for instance, for the experiment No.3 the training time exceeded 6 hours.

B. DNN with convolution and recurrent layers

The authors simulated DNN on the basis of a set of convolutional layers and a recurrent LSTM layer. The architecture of this network is provided in Figure 2. Simulation was conducted in MATLAB 2019 at a PC with the following parameters Intel Core i7-4710HQ, 4 cores by 2.5 GHz, RAM 6 Gb, GPU GeForce GTX 850M, 64-digit operational system Windows. The LTSM layer had 100 neurons for the experiments. Simulation results are shown in Table 2. To optimize machine costs and obtain the result for a shorter period of time, the information blocks from a two-dimensional tensor were transformed into three-dimensional. Without such transformation the training time increases by times; for instance, for the experiment with the training sample 45,000 x 41 the training time equaled 83 minutes.

TABLE II. CNN-LSTM NETWORK SIMULATION RESULTS.

No.	Training sample size	Test sample size	Transformed tensor size	Training sample precision, %	Test sample precision, %	Training time, min
1	150,000 x 41	30,000 x 41	300 x 500 x 41	100	100	7.25
2	300,000 x 41	75,000 x 41	3000 x 100 x 41	100	99.8	13.45
3	900,000 x 41	225,000 x 41	1,800 x 500 x 41	100	98.67	35.1
4	300,000 x 41	75,000 x 41	600 x 500 x 41	100	100	8.1

As it is seen from the tables above, the application of an additional convolutional layer increases the training time in general without precision improvement. This can be explained by the fact that the sequence, presented for training, is not long and contains only 41 features (and thousands examples) which is processed by the recurrent network without additional convolutional processing with a high precision degree. However, in reality the data for the classifier from the networking devices of the computer network will be sent continuously including thousands of intervals, that is why the strategy of joint use of convolutional and recurrent layers can be more reasonable in a live situation according to the authors.

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

C. *Optimization of DNN parameters on the basis of the genetic algorithm*

The quality and time of neural network training directly depends on its structure, i.e., on the number of layers, neurons in the layers, weighting coefficients, parameters of activation functions, etc. Sometimes too cumbersome and/or non-optimal network structure results in the same (or worse) indicators as compared with those indicators at its optimal structure; however, it requires a significant amount of calculational and time resources which is the criterion of inefficient operation. By this reason the search for the optimal parameters of a neural network to solve a specific applied problem remains the relevant problem. It is also desirable that this search is conducted automatically. Convolutional neural networks are not an exception. To optimize DNN parameters, this section uses a genetic algorithm.

Genetic algorithms simulate the natural evolution processes and belong to the search evolution methods. Using the genetic algorithm, one could obtain the solution corresponding to the global optimum or close to it [10]. They are based on the main evolution concept: survival, or selection of the best individuals, breeding and mutation. The genetic algorithm represents the following object:

$$\Gamma A(P^0, r, l, sl, Fu, cr, m, ot), \tag{8}$$

where ΓA - genetic algorithm, P^0 – initial population, r – number of population elements, l – length of the bit string used for solution coding, sl – selection operator, Fu – fitness function (utility function) defining the "suitability" of solution , cr – crossing-over operator defining the opportunity to obtain a new solution, m – mutation operator, ot – selection operator [19]. One of the genetic algorithm schemes with the self-configuration elements is presented in Fig.4.

A genetic algorithm was used to optimize the following DNN parameters with convolutional and recurrent layers (Fig.2): number of neurons in the LSTM layer, block size in the pooling layer, the probability layer in the noise elimination, number of filters and the size of these filters in convolutional layers.

Due to the fact that the genetic algorithm searches for the optimal parameters by means of randomizing, combining and variation of the required parameters, the selection of the neural network structure on the basis of a genetic algorithm is the procedure demanding significant computational capabilities. In this connection the following values were used as genetic algorithm parameters: the number of generations - 30 or less and the population number - 50 or less. Simulation was conducted in MATLAB 2019 at a PC with the following parameters Intel Core i7-4710HQ, 4 cores by 2.5 GHz, RAM 6 Gb, GPU GeForce GTX 850M, 64-digit operational system Windows. Depending on the initial parameters, the genetic algorithm operation takes from 1-2 to 12 hours.

Table 3 presents some values obtained during the search for DNN parameters with the genetic algorithm. Figure 5 demonstrates the change of the penalty function value at the genetic algorithm operation.

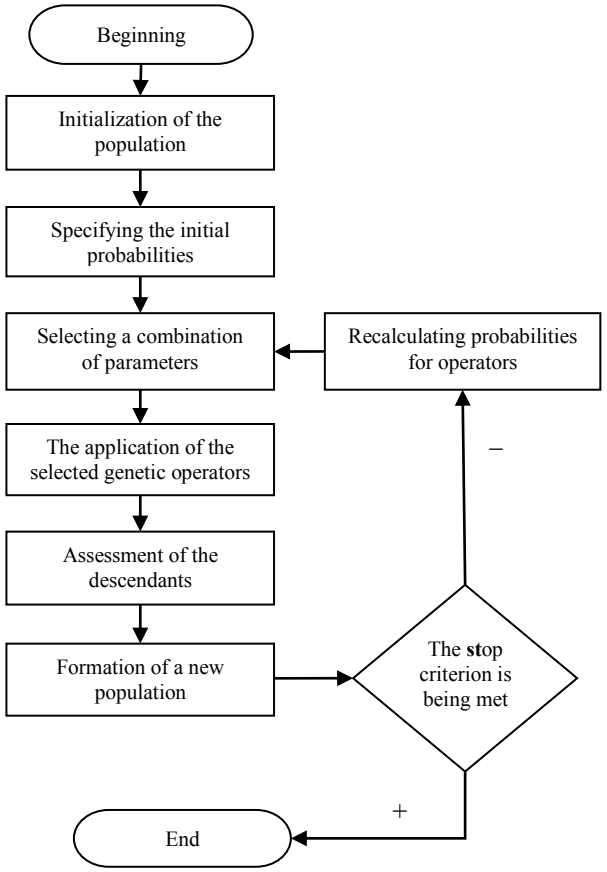

Fig. 4. Genetic algorithm flow diagram

TABLE III. DNN PARAMETERS OBTAINED BY THE GENETIC ALGORITHM.

No.	Number of neurons in the LSTM layer	Block size in the pooling layer	P-value in the noise elimination value, %	Filter number in the convolutional layer	Filter size in the convolutional layer	Precision, %
1	40	5	89	10	10	33
2	70	5	74	7	6	66
3	100	5	52	9	4	67
4	60	4	94	10	4	67
5	50	7	49	4	5	33
6	60	5	87	6	4	67
7	80	6	36	9	6	66
8	40	9	2	5	6	33
9	50	7	49	4	5	100
10	50	4	25	6	6	100

978-1-7281-6951-4/20/$31.00 © 2020 IEEE

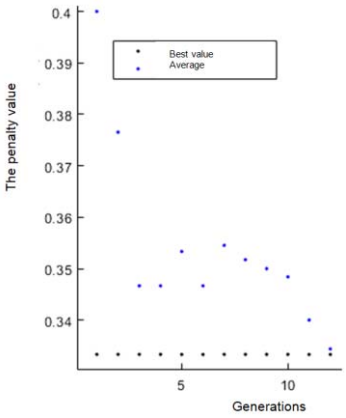

Fig. 5. Search for solution by means of the genetic algorithm

During the use of the genetic algorithm to optimize the DNN parameters the following has been noticed (which can be also seen from Table 3):

1. The mean number of neurons resulted in the best precision.

2. The probability in the elimination layer shall not be higher than 50%.

3. The values defining the filter size and number should be commensurate.

Therefore, at the availability of calculational and time resources, the use of genetic algorithms to optimize DNN parameters allows obtaining a more high-quality model at the training stage. Theoretically, this model allows online simulating the data which reflect the real situation in maximum detail; it its turn, it will assist the managing body in taking correct and timely decisions. As an alternative option for the genetic algorithm, the authors consider the opportunity of ant colony optimizations [20] or swarm intelligence in general [15].

V. Conclusion

The article states a problem of a multiclass network classification of computer attacks. The authors consider the opportunity of DNN use for solving a set problem.

The authors select the architecture of a deep neural network based on the strategy combining a set of convolution and recurrent LSTM layers. The research suggests optimizing the parameters of the selected deep neural network on the basis of a genetic algorithm.

The authors present the simulation results that show an opportunity to solve the problem of network classification for the reasonable amount of time and limited machine resources.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient Based Learning Applied to Document Recognition,” IEEE Intelligent Signal Processing, 1998, pp. 306-351.

[2] S. Hochreiter, J. Schmidhuber, “Long-Short Term Memory,” Neural Computation, 1997, Vol. 9, No. 8, pp. 1735-1780.

[3] O.S. Amosov and S.G. Amosova, “The neural network method for detection and recognition of moving objects in trajectory tracking tasks according to the video stream,” In 2019 26th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS), pages 1– 4, May 2019.doi: 10.23919 / ICINS.2019.8769340.

[4] O.S. Amosova, S.G. Amosova, S.V. Zhiganov, Yu.S. Ivanov, F.F. Pashchenko, “Computational Method for Recognizing Situations and Objects in the Frames of a Continuous Video Stream Using Deep Neural Networks for Access Control Systems,” Journal of Computer and Systems Sciences International, 2020, Vol. 59, No. 5, pp. 712-727.

[5] O.S. Amosov, S.G. Amosova, D.S. Magola, “Identification of information recourses threats based on intelligent technologies, fractal and wavelet analysis,” IEEE International Conference on Applied System Invention (ICASI), Chiba, Japan, 13-17 April 2018, pp. 528-531.

[6] O.S. Amosov, S.G. Baena, “The hierarchical approach to designing the Intelligent Information and Telecommunication System for Higher Educational Institution Security,” Proceedings of 2017 20th IEEE International Conference on Soft Computing and Measurements, SCM 2017, pp. 116-119.

[7] O.S. Amosov and S.G. Baena, “Decomposition Synthetic Approach for Optimum Nonlinear Estimation,” IFAC - Papers OnLine, 2015, vol. 48 (11), pp. 819-824.

[8] O.S. Amosov, Y.S. Ivanov, and S.V. Zhiganov, “Human localization in the video stream using the algorithm based on growing neural gas and fuzzy inference,” XII Intelligent Systems Symposium, INTELS'16, Procedia Computer Science, 2017, no 103 , pp. 403-409.

[9] M.D. Mesarovich and Y. Takakhara, General Systems Theory: Mathematical Foundations, Academic Press New York, San Francisco, London, 1975.

[10] Methods of classical and modern theory of automatic control: Textbook in 5 vols .; 2nd ed., Rev. and add. V.5: Methods of modern theory of automatic control, Ed. K.A. Pupkova, N.D. Egupova, M .: Publishing house of MSTU im. N.E. Bauman, 2004, 784 p.

[11] F. Scholle, “Deep Learning in Python”, SPb: Peter, 2018, 400 p.

[12] O.S. Amosov, S.G. Amosova, Y.S. Ivanov, S.V. Zhiganov, “Using the deep neural networks for normal and abnormal situation recognition in the automatic access monitoring and control system of vehicles,” Neural Computing and Applications, 2020, DOI: 10.1007 / s00521-020-05170-5.

[13] S. Mallat, “A wavelet tour of signal processing”, Academic press, 1999.

[14] D.V. Budylsky, “GRU and LSTM: modern recurrent neural networks,” Young Scientist, 2015, No. 15, pp. 51-54, URL https://moluch.ru/archive/95/21426/.

[15] J. Chung, C. Gulcehre, K.H. Cho, Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” 2014. arXiv: 1412.3555.

[16] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[17] R.A. Markov, V.V. Bukhtoyarov, A.M. Popov, “Research of neural network technologies for identifying information security incidents”, Young Scientist 2015, no. 23, pp. 55-60.

[18] M.P. Komar, “A neural network method for identifying computer attacks”, Optoelectronic information and energy technologies 2010, no. 2, pp. 105-109.

[19] The MathWorks, Inc. (CITM Exhibitor), MATLAB Documentation, https://docs.exponenta.ru/gads/ga.html.

[20] S.D. Shtovba, “Ant algorithms”, Exponenta Pro., Mathematics in applications, 2003, No. 4, pp. 70-75.

[21] G. Beni, J. Wang, “Swarm Intelligence in Cellular Robotic Systems,” Proceed. NATO Advanced Work-shop on Robots and Biological Systems, Tuscany, Italy, June 26-30, 1989.

978-1-7281-6951-4/20/$31.00 © 2020 IEEE