Neural network approach to 5G digital modulation recognition

Bohdan Kotyk[1,†], Denys Bakhtiiarov[1,2,*,†], Oleksandr Lavrynenko[1,†], Bohdan Chumachenko[1,†], Veniamin Antonov[1,†], Vladyslav Fesenko[1,†] and Volodymyr Chupryn[1,†]

[1] *National Aviation University, Liubomyra Huzara Ave. 1, Kyiv, 03058, Ukraine*

[2] *State Scientific and Research Institute of Cybersecurity Technologies and Information Protection, Maksym Zalizniak Str., 3/6, Kyiv, 03142, Ukraine*

Abstract

In digital communication, correct identification of the modulation types used is key to ensure reliable processing of received signals. This work presents results for the task of digital modulation type identification, assuming perfect receiver synchronization has already been achieved. A multilayer perceptron has been created and adjusted with the help of the Adam optimization algorithm towards lowering classification errors along with shared cumulants, which appeared to be great informative features in modulation type determination. Also, after results was collected, the analysis was carried out on how hidden layers affect the performance of the neural network; it was found that a three-layer perceptron gives excellent accuracy. Results show that at a signal-to-noise ratio (SNR) of 5 dB, recognition accuracy is about 99%. This methodology gives a safe pathway for correct modulation classification and future research to improve the model's capability to estimate SNR and cumulant selection refining for better classification. In future work it is planned that model will be able to proceed uncertain input parameters to further enhance the system adaptability and effectiveness in real life scenarios.

Keywords

machine learning, neural network, 5G, high order cumulants, digital modulation, ReLU, Softmax, Adam, backpropagation

1. Introduction

Currently, there are multiple data mining approaches that offer efficient solutions to problems regarding automatic digital modulation recognition. Among them are decision tree construction, machine learning, and artificial neural networks. The most widespread, though, is the method of classifying and identifying objects with the use of an artificial neural network. ANN is a computerized emulation of biological neural systems in the living body's natural brain. It illustrates one of many human-like intelligences - learning with the complete absorption of given facts, processing, drawing conclusions, and selecting appropriate decisions like human's brain [1–4].

Artificial neural networks operate by sharing the load of data processing among elementary local units-neurons. These units relate to each other via special links called synaptic connections. The information to be remembered is passed inside the network using weights assigned to these connections. Increasing functionality In ANNs does not happening through programming as is the case with digital computers; it occurs via training the artificial neural network. Consequently,

CH&CMiGIN'24: Third International Conference on Cyber Hygiene & Conflict Management in Global Information Networks, January 24–27, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

3351920@stud.nau.edu.ua (B. Kotyk); bakhtiiaroff@tks.nau.edu.ua (D. Bakhtiiarov); oleksandrlavrynenko@tks.nau.edu.ua (O. Lavrynenko); bohdan.chumachenko@npp.nau.edu.ua (B. Chumachenko); veniamin.antonov@npp.nau.edu.ua (V. Antonov); fes_vlad@ukr.net (V. Fesenko); volodymyr.chupryn@npp.nau.edu.ua (V. Chupryn)

0009-0005-0821-9108 (B. Kotyk); 0000-0003-3298-4641 (D. Bakhtiiarov); 0000-0002-3285-7565 (O. Lavrynenko); 0000-0002-0354-2206 (B. Chumachenko); 0000-0003-2244-262X (V. Antonov); 0000-0002-8933-9291 (V. Fesenko); 0000-0001-9412-7413 (V. Chupryn)

training a neural network is seen as an incremental activity that enables the network to acquire pattern recognition capabilities with respect to data by modifying the weights of synaptic connections among its neurons [1].

One of the most well-known types of ANNs is a multilayer neural network, which is a multilayer perceptron (MLP), shown in Figure 1, which has several layers, each of which consists of many neurons.

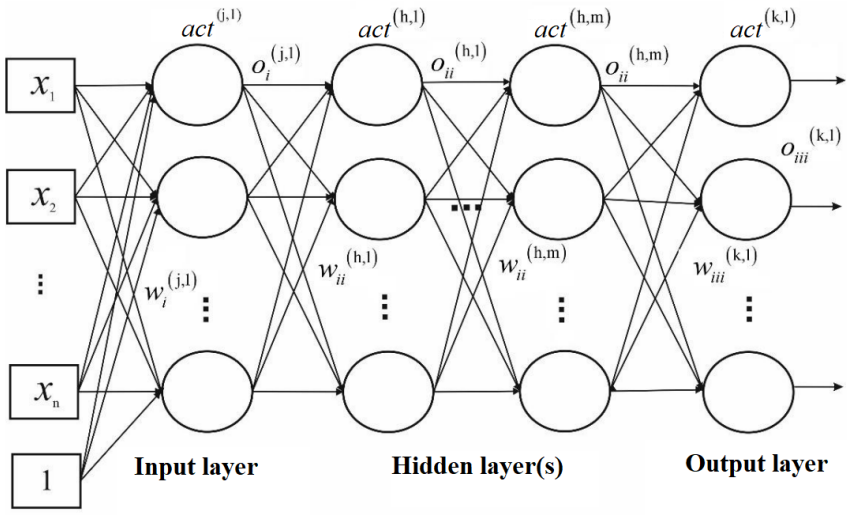

Figure 1: Structure of a multilayer perceptron.

A neural network consists of several or many layers, and therefore a large number of neurons. A neuron is a communication device that has input and output connections. It's function consists of two parts: the first is to calculate the discriminant function *df*, and the second is activation function $act^{(j,i)}$, as shown in Figure 2 [3].

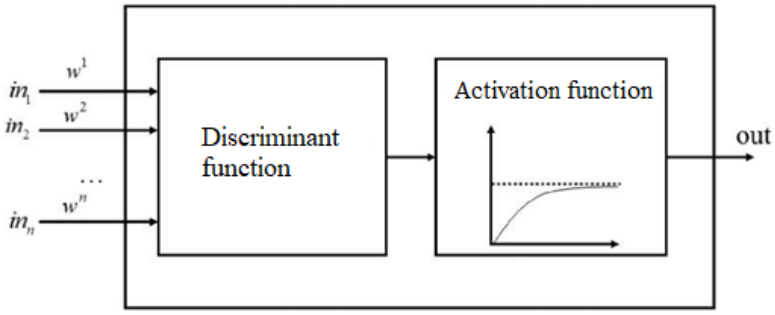

Figure 2: Structure of neuron.

The following values are used in the figure: *in* - input information features, *out* - output values of the neuron, *w*' - weight coefficients (synapse), *n* - number of input information features.

2. Basic technological stages

For our neural model to be successful in determining modulation type we need to choose its parameters, such as: activation and classification functions, backpropagation algorithm and information features for determining modulation to teach our model [4].

2.1. Activation function

The discriminant function *df* is a weighted sum of the input signals in the input layer, this sum for a neuron is expressed as [5]:

$$df_j = act^{(j,1)} = w_0^{(j,1)} + \sum_{i=1}^{n} w_i^{(j,1)} x_i \quad (1)$$

where $w^{(j,1)} = \left(w_0^{(j,1)}, w_1^{(j,1)}, \ldots, w_n^{(j,1)}\right), j = \overrightarrow{(1, N_1)}$ – row vector of synaptic connections for the j-th neuron, N_1 – number of neurons in the input layer, n – number of input features, x_i – row vector of the i-th input.

The activation function in a neural network greatly influences the output of a neuron given its input. The activation function allows for making a neuron's decision to activate and propagate information or to stay quiescent. There are many different types of activation functions that can be applied to neural networks, each with its own pros and cons [5]. The following functions have been used:

1. Linear Activation Function: It simply returns the input value without any changes. The function looks like this [6]:
$$f(x) = x. \quad (2)$$
2. Sigmoid activation function. Function formula:
$$f(x) = 1/(1 + e^{(-x)}). \quad (3)$$
Converts any number to a value between 0 and 1, making it useful for calculating probabilities.
3. Hyperbolic tangent (tanh): Function formula:
$$f(x) = (e^x - e^{(-x)})/(e^x + e^{(-x)}). \quad (4)$$
This is another S-shaped function. It converts numbers to values in the range [-1, 1].
4. ReLU (Rectified Linear Unit): Function formula:
$$f(x) = max(0, x). \quad (5)$$

In this article, the activation function used was “ReLU”. A simple nonlinear function, ReLU transforms the input signal by setting all negative values to zero and keeping positive values unchanged. In this case, the output of each neuron is defined as follows [6, 7]:

$$o^{(j,1)} = \text{ReLU}(df_j) = max(0, df_j), \quad (6)$$

In the hidden layer, each neuron processes a weighted sum of its’ inputs, which can be represented by the following expression [7]:

$$df_h = act^{(h,m)} = \sum_{ii=1}^{N_m} w_{ii}^{(h,m)} o_{ii}^{(h,m-1)}, \quad (7)$$

$$o^{(h,m)} = \text{ReLU}(df_h) = max(0, df_h). \quad (8)$$

where $w_{ii}^{(h,m)}$ – row vector of synaptic connections at the input of the h-th neuron of the m-th hidden layer, m – number of hidden layers, N_m – the number of neurons in the m-th hidden layer.

In the output layer, each neuron processes a weighted sum of its inputs:

$$df_k = act^{(k,1)} = \sum_{iii=1}^{N_k} w_{iii}^{(k,1)} o_{iii}^{(k,m)}, \quad (9)$$

where $w_{iii}^{(k,1)}$ is row vector of synaptic connections at the input of the k-th neuron, N_k – number of neurons in output layer.

2.2. Classification function

To determine the type of modulation, a classification function called the Softmax function is used. It transforms a set of input values into a set of positive numbers whose sum is 1 [8]. Thus, each number in the output vector represents the probability of the corresponding class. In this case, the Softmax function is represented as follows [9]:

$$o^{(k,1)} = f(df_k) = e^{df_k} / \sum_{k=1}^{N_k} e^{df_k}, \quad (10)$$

where $o^{(k,1)}$ is the prediction of the k-th neuron of the output layer.

2.3. Backpropagation algorithm

The neural network training is based on the backpropagation algorithm. In training process, the neural network first makes estimates based on input data and then compares those estimates with the correct answers to derive error [10].

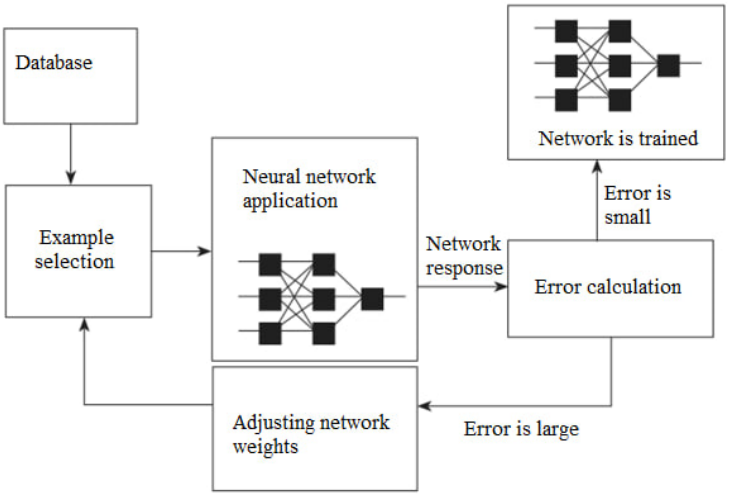

Figure 3: The process of training a neural network.

This comparison leads to the calculation of error, which is subsequently propagated back through the network to adjust weights to minimize this error. The utility of this error function can be expressed as follows [10]:

$$E(\vec{W}) = 0.5 \sum_{k=1}^{N_k} \left(\mu^{(k,1)} - \nu^{(k,1)}\right)^2, \tag{11}$$

where $\mu^{(k,1)}$, $\nu^{(k,1)}$ - the desired and predicted state of the neural network outputs of the k-th neuron of the output layer, N_k – number of neurons in output layer

Now, many methods exist to minimize the error function but four are most frequently used: Gradient method, Adagrad method, RMSProp method, and Adam method which excel in the pace algorithm evolution pace and more common in the literature.

1. Gradient descent method.

The concept of gradient descent is to perpetually modify the network weights in the direction opposed to the loss gradient [11]. The gradient indicates the direction of the highest increase of the function, so and move in the opposite direction helps to lessen the function. The formula applies for an update of weights and done once per iteration of gradient descent

$$w_i(t+1) = w_i(t) + \Delta w_i, \tag{12}$$

$$\Delta w_i = -\lambda \cdot \frac{\partial E(\vec{W})}{\partial w_i}, \tag{13}$$

where $w_i(t)$ and $w_i(t+1)$ are the previous and updated values of the weight coefficient of the i-th neuron, λ is the learning rate [12].

One big drawback of this method is that local minima and saddle points are found within the loss function. If the loss function has many local minima or saddle points, gradient descent might get stuck at one of these places and won't be able to reach the global minimum.

2. Adagrad method.

Adagrad adjusts the global learning rate for each parameter based on the gradient changes for that particular parameter. It reduces the learning rate for parameters that receive frequent updates and raises the learning rate for one that receive infrequent updates. This contrast with gradient method as update formula of Adagrad includes sum of squared gradient G_t in denominator. If a parameter has associations with a chain of normally active neurons, it reset frequently; hence, the overall sum accumulates quickly. Update formula can then be expressed as:

$$w_i(t+1) = w_i(t) - \frac{\lambda}{\sqrt{G_t + \epsilon}} \cdot \frac{\partial E(\vec{W})}{\partial w_i}, \quad (14)$$

where ε is a small value to prevent division by zero (typically around 10^{-8}).

Because of its adaptive nature Adagrad is less prone to manual errors than standard gradient descent. The accumulation of squared gradients in G is not bounded so it can make a very small denominator in the update formula and thus updates small. This can cause the algorithm to stop too early leading to a poor fit. Moreover, Adagrad does not have an explicit notion of moment as other optimization methods do like SGD with moment or Adam [13]. This could lead to early stoppage of algorithm when it's not properly trained yet.

3. RMSProp method.

This is a heuristic optimization method proposed by Geoff Hinton. It was conceived as an improvement on Adagrad to solve its gradient accumulation problem. Instead of storing all past squared gradients, as Adagrad does, RMSProp uses a running average of past squared gradients $E[g^2]_{t-1}$ that decays exponentially. This makes it robust to the learning rate decay problem. The exponentially decaying running average at time t is defined as [14]:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)\left[\frac{\partial E(\vec{W})}{\partial w_i}\right]^2, \quad (15)$$

where γ is the conservation coefficient in the range from zero to one. The closer γ is to one, the larger the accumulation window and the more intense the smoothing of the square of the gradient, i.e., a moving average is determined that decreases exponentially. In practice, the value y = 0.9 is usually used.

Then the update formula for RMSProp looks like this:

$$w_i(t+1) = w_i(t) - \frac{\lambda}{\sqrt{E[g^2]_t + \epsilon}} \cdot \left[\frac{\partial E(\vec{W})}{\partial w_i}\right]^2. \quad (16)$$

RMSProp fixes the problem of Adagrad's slow learning rate by taking a moving average of the squared gradient, which decays exponentially. Like Adagrad, however, RMSProp does not have a formal notion of "moment," which can hurt its performance when collinear trends are present [15].

4. Adam method.

Currently, the most effective approach to optimize the neural network training process is the Adam method, proposed in the mid-2010s. It combines the concepts of RMSProp and SGD with momentum. The Adam method differs from other methods in that we do not store the Δwi value, but instead the average gradient value, using two stores: one for the moving average of the gradient values (e.g., SGD with momentum), and the other for the moving average of the squared gradient values (e.g., RMSProp). The working principles of these two accumulators are described by the following formulas [16]:

$$m_t = \delta_1 m_{t-1} + (1-\delta_1)\frac{\partial E(\vec{W})}{\partial w_i}, \quad (17)$$

$$v_t = \delta_2 v_{t-1} + (1-\delta_2)\left[\frac{\partial E(\vec{W})}{\partial w_i}\right]^2, \quad (18)$$

where δ_1 and δ_2 are the parameters for exponentially weighted averages with weights of 0.9 and 0.999, respectively. One important difference is the initial value of m_t and v_t, since if they are initially set to zero, they will take a long time to accumulate. To overcome this problem, special adjustments are used for them. These adjustments are defined as follows [17]:

$$m_t = \frac{m_t}{1-\delta_1^t}, \hat{v}_t = \frac{v_t}{1-\delta_2^t}, \quad (19)$$

Therefore, the update formula is:

$$w_i(t+1) = w_i(t) - \frac{\lambda}{\sqrt{\hat{v}_t + \epsilon}} m_t. \quad (20)$$

Figure 4 shows the results of the neural network simulation with the four error minimization methods listed above; this figure allows us to estimate the learning rate associated with each method. The figure demonstrates that the Adam method has a high degree of accuracy in recognizing various types of digital manipulation (with a limited number of training cycles), it also has greater stability in the network training process. As a result, the Adam method was used to minimize the error rate as much as possible.

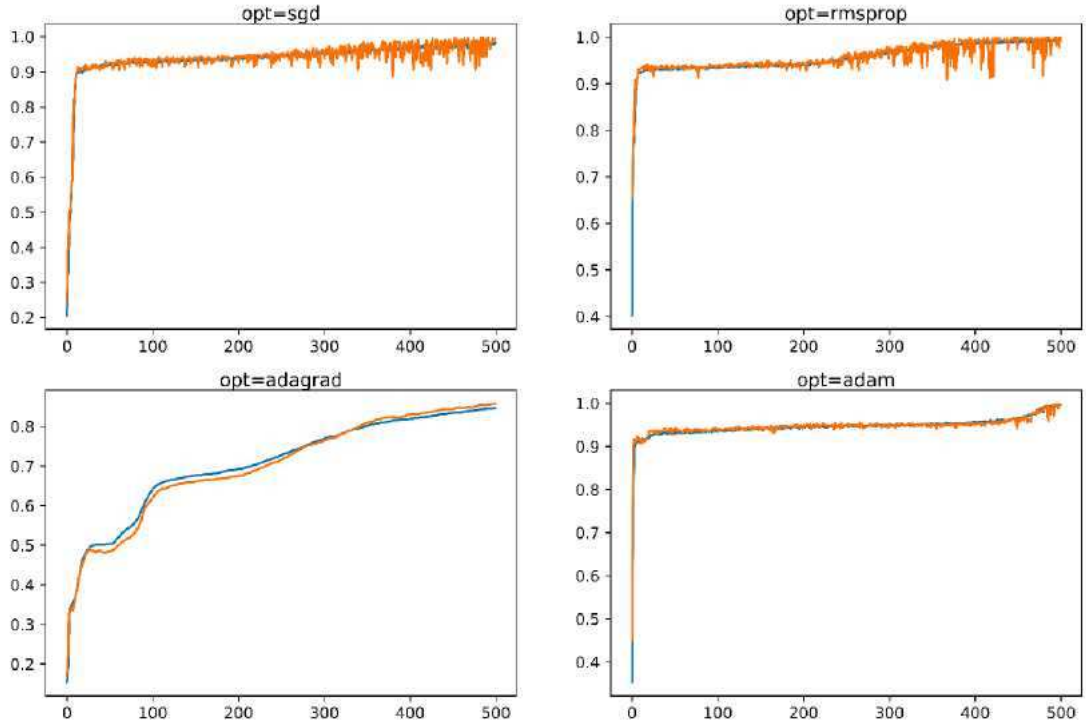

Figure 4: Comparison of learning speed between Gradient, Adagrad, RMSProp and Adam learning methods.

2.4. Information features for determining modulation

As defining characteristics for training neural model to distinguish modulations were chosen cumulants of two-dimension random processes and mixed moments that corelate to them.

Cumulants are insensitive to the addition of Gaussian white noise. In practice, the carrier frequency at reception may not be accurately determined because of, among other things, natural instabilities in the transmitter and receiver oscillator frequencies, Doppler shift, and timing error. Since they are statistical measures of the signal, high order cumulants are therefore less susceptible to these forms of distortion.

Cumulants (semi-invariants) are special combinations of moments that have useful properties such as additivity for independent random variables, where moment is usually defined as the mathematical expectation of the product of the process values at different moments in time for a stochastic stationary process. So, the moments of distribution or moments of a random variable φ are called integrals [18–21]:

$$E_n^{\phi} = \int_{-\infty}^{\infty} x^n W_{\phi}(x) dx, \tag{21}$$

where $W_{\phi}(x)$ is probability distribution density, n - moment order number.

The cumulant of a random process is the expansion coefficient of the logarithm of the characteristic function of a random process into a Taylor power series:

$$ln\,\theta\,(x) = \sum_{n=1}^{\infty} \frac{C_n (ix)^n}{n!}, \tag{22}$$

where n - is the cumulant order.

There is a relationship between the cumulant C_n and the moment E_n of a random variable, but the generating functions of the cumulants are difficult to calculate if we are not familiar with the exact distribution of the signal. However, the formulas for their relationship under the same distribution are known [1], we can use the expressions of cumulants features of two-dimensional random processes $C_{n,m}$, through their mixed moments $E_{n,m}$ up to the 9th order, from $C_{2,0}$:

$$C_{2,0} = E_{2,0}, \tag{23}$$

to $C_{5,4}$:

$$\begin{aligned} C_{5,4} = {} & E_{5,4} - 10E_{4,4}E_{2,0} - 20E_{4,3}E_{1,1} - 6E_{5,2}E_{0,2} - 10E_{4,2}E_{3,0} - 40E_{3,3}E_{2,1} - \\ & -30E_{4,2}E_{1,2} - 3E_{5,1}E_{0,3} - E_{5,1}E_{0,2} - 5E_{1,4}E_{4,0} - 40E_{2,3}E_{3,1} - 60E_{3,2}E_{2,2} - \\ & -20E_{4,1}E_{1,3} - E_{5,0}E_{0,4} + 30E_{1,4}E_{2,0}^2 + 60E_{2,3}E_{1,1}E_{2,0} + 180E_{2,4}E_{0,2}E_{2,0} + \\ & +240E_{1,3}E_{1,2}E_{2,0} + 120E_{3,2}E_{2,0}E_{1,1} + 111E_{4,1}E_{1,1}E_{0,2} + 4E_{5,0}E_{0,2} + \\ & +10E_{4,1}E_{1,1} + 2E_{5,0}E_{0,2} + 300E_{2,2}E_{1,2}E_{2,0} + 60E_{2,2}E_{1,1}E_{2,0} + \\ & +270E_{1,2}^2E_{2,0} + 720E_{1,1}E_{1,2}E_{2,0} - 360E_{1,2}E_{2,0}E_{0,2} - 720E_{1,1}E_{1,2}. \end{aligned} \tag{24}$$

Cumulants are complex numbers, and the main differences in their values for different types of digital modulation can be seen in the real components of these numbers. For brevity, we will use the term "cumulants" in what follows, referring only to their real parts.

The initial data for calculating moments and cumulants are the complex signal $r_k(t) = I_k(t) + iQ_k(t)$ and its complex conjugate $\overline{r_k(t)} = I_k(t) - iQ_k(t)$, where $I_k(t)$ - in-phase component , $iQ_k(t)$ - quadrature component.

Tables 1 and 2 show the values of the real part of the cumulant, selected as identification features, in various types of modulation with a signal-to-noise ratio of 5 dB, up to the ninth order.

Table 1

Cumulant Values at SNR=5 dB from 2nd to 6th Order

	C20	C11	C30	C21	C31	C22	C50	C41	C32	C60
GMSK	0.004	1.439	0.002	-0.014	-0.012	-0.012	-0.007	-0.002	-0.034	-0.089
QAM-8	0.61	1.243	-0.001	-0.079	-0.836	-0.936	-0.012	-0.002	0.134	3.379
QAM-16	0.002	1.237	0.001	-0.057	-0.006	-0.624	0.051	0.035	-0.006	1.147
QAM-64	0.003	1.255	0.001	-0.049	0.008	-0.502	0.061	-0.004	-0.015	2.157
APSK-16	0.005	1.108	-0.002	-0.03	0.001	-0.029	0.034	-0.003	-0.016	2.318
APSK-32	0.003	0.991	0.001	-0.019	-0.001	-0.018	0.032	-0.002	-0.013	1.212
BPSK	-0.987	1.267	0.003	-0.028	-0.002	-0.028	-0.03	0.001	0.012	3.445
QPSK	0.007	1.269	0.002	-0.01	0.001	-0.012	0.021	-0.002	0.014	0.259
PSK-8	0.016	1.348	0.003	-0.024	-0.025	-0.025	0.028	0.008	0.035	0.148
FSK-2	0.002	1.391	0.003	-0.009	-0.01	-0.009	0.037	-0.004	0.013	0.213

Table 2

Cumulant Values at SNR=5 dB from 6th to 9th Order

	C42	C70	C61	C43	C80	C62	C54	C90	C71	C63
GMSK	0.03	-4.391	0.258	-0.05	-6.89	0.21	-0.47	1.345	-1.33	108
QAM-8	4.74	-56.6	1.359	6.287	-55.8	1.425	-51.2	62.02	6.87	1010
QAM-16	0.04	-14.8	1.142	-5.47	-18.4	0.996	-0.38	5.025	-8.15	3042
QAM-64	0.02	-0.162	0.159	-7.92	0.153	0.196	-0.04	0.328	-5.68	1388
APSK-16	0.001	0.164	0.814	-0.02	-0.05	0.255	0.161	0.322	0.12	509
APSK-32	-0.02	-267.2	-0.04	-0.04	-0.26	0.143	0.131	0.34	0.087	269
BPSK	-0.12	-1.494	0.34	-0.04	0.388	0.36	0.261	0.753	-0.76	2466
QPSK	0.12	-39.3	-0.5	0.256	0.111	0.107	0.156	7.567	0.162	988
PSK-8	0.013	-2.276	0.138	0.014	-0.37	0.11	1.491	1.884	0.56	1046
FSK-2	0.013	-0.308	0.156	0.006	0.214	0.105	0.156	1.006	9.426	1013

Table 3
Characteristics of Neural Network

Neural Network Parameters	Value
Number of neurons in the input layer	10
Number of neurons in each hidden layer	20
Number of hidden layers	10
Activation function in input and hidden layers	ReLU
Activation function in the output layer	Softmax
Maximum number of epochs	500
Early stopping value	30

3. Process and result of neural model training

In a multilayer perceptron, hidden layers have great significance in both the functioning of the network and training; these are placed between the input and output layers. The neurons in these layers receive data from previous layers, process that data, and send it to other layers. Hidden layers are for training purposes and gather features from the input data to be further processed in the output layer for classification or prediction. This section determines how the number of layers impacts the accuracy in recognizing various types of digital manipulations.

Multilayer neural networks were implemented in Python using Colab Notepad. Multilayer neural networks were implemented in Python using Colab Notepad with algorithm displayed on Figure 5.

A dataset was prepared for the classification of digital modulations comprising 10,000 different signals (1,000 signals for each type of modulation), where 7,200 are for training, 1,800 for validation, and 1,000 for testing purposes. The results of the simulations are represented in Figure 7. It also can be viewed in Figure 6 that the level of identification of types of digital modulation can be estimated by the quantity of layers.

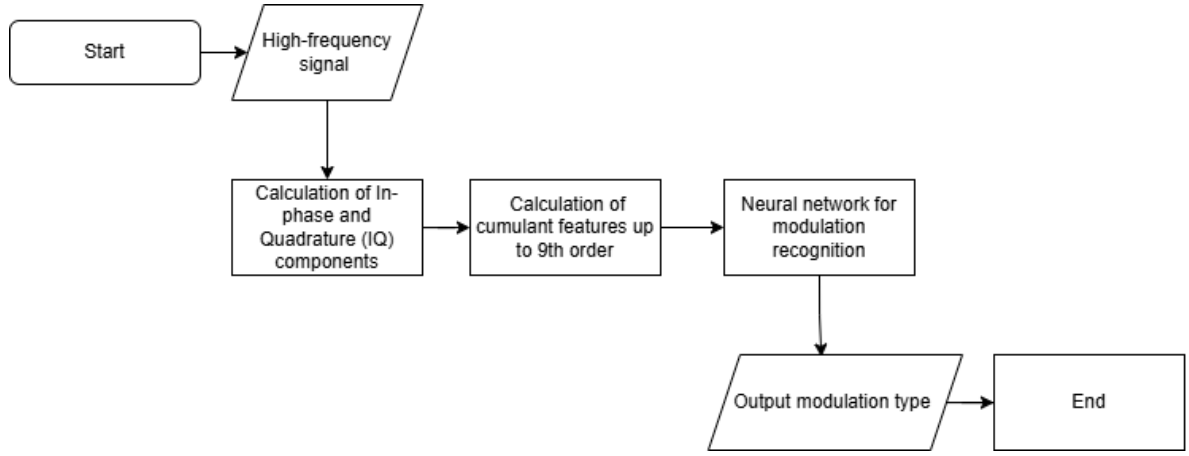

Figure 5: Algorithm for recognizing types of digital modulation.

Figure 7 show the identification of digital modulation at a signal-to-noise ratio of 5 dB for different ANN structures. The numbers are presented in the form of a table, the rows and columns of which correspond to the signal modulation type and the algorithm solution. The cells contain the results that determine the modulation types. For example, at single-layer model: When identifying QAM-8 signals, all 100 signals involved in the computer experiment were correctly identified. When identifying QAM-16 signals, 99 signals were correctly identified, but 1 signal was misidentified as QAM-64.

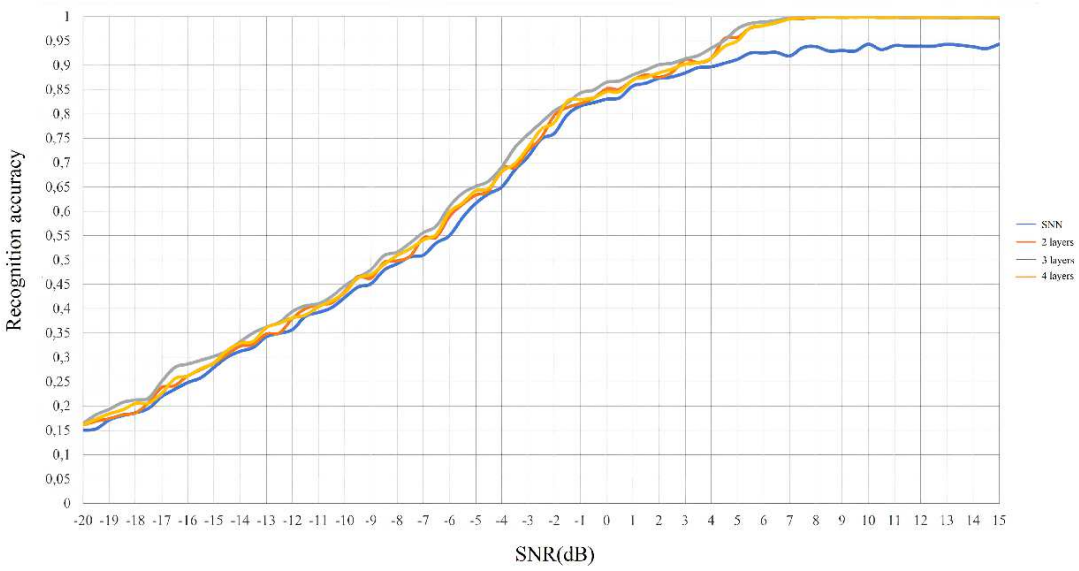

Figure 6: Dependence of recognition accuracy on the signal-to-noise ratio.

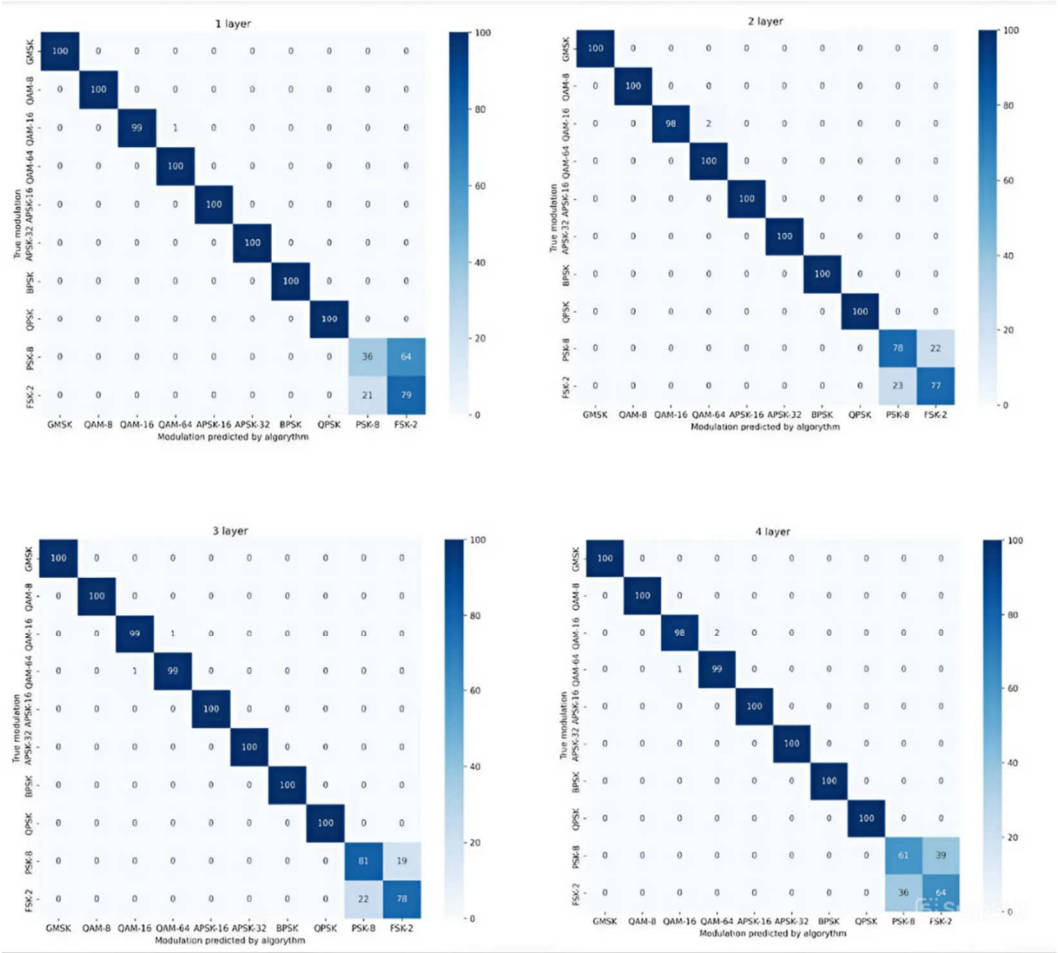

Figure 7: Result of recognizing types of digital modulation using hidden layers in quantity from one to four.

4. Conclusions

This article presents a study of digital modulation type recognition for ideal receiver synchronization, i.e., with a known frequency and initial phase of the received signal. It is shown why shared cumulants and moments are excellent to be used as information features for determining

modulation types. A multilayer perceptron structure is created that uses Adam as error minimization function. Influence of the number of hidden layers of the neural system on the accuracy of digital modulation type recognition is investigated. It is shown that a perceptron with three hidden layers recognizes digital signal modulation types with excellent accuracy. For instance, with an SDN of 5 dB, the modulation detection accuracy was approximately 0.99.

In future works, it is possible to train this model to determine SDN and further improve its efficiency by choosing specific cumulants that best allow determining the modulation type and SDN. It is also planned to improve the technique for determining the modulation type when the values of the input parameters are not known exactly.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] O. Holubnychyi, O. Lavrynenko, D. Bakhtiiarov, Well-Adapted to Bounded Norms Predictive Model for Aviation Sensor Systems, in: I. Ostroumov, M. Zaliskyi, (Eds.), Proceedings of the International Workshop on Advances in Civil Aviation Systems Development. ACASD 2023, volume 736 of Lecture Notes in Networks and Systems, Springer, Cham, 2023, pp. 179–193. doi: 10.1007/978-3-031-38082-2_14.

[2] B. Jdid, W. H. Lim, I. Dayoub, K. Hassan, M. R. B. Mohamed Juhari, Robust Automatic Modulation Recognition Through Joint Contribution of Hand-Crafted and Contextual Features, IEEE Access 9 (2021) 104530–104546. doi: 10.1109/ACCESS.2021.3099222.

[3] D. Bakhtiiarov, O. Lavrynenko, N. Lishchynovska, I. Basiuk, T. Prykhodko, Methods For Assessment And Forecasting Of Electromagnetic Radiation Levels In Urban Environments. Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska 11(1) (2021) 24–27. doi: 10.35784/iapgos.2430.

[4] R. S. Odarchenko, S. O. Gnatyuk, T. O. Zhmurko, O. P. Tkalich, Improved method of routing in UAV network, in: Proceedings of International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD), IEEE, Kyiv, Ukraine, 2015, pp. 294–297. doi: 10.1109/APUAVD.2015.7346624.

[5] M. Zaliskyi, R. Odarchenko, S. Gnatyuk, Y. Petrova, A. Chaplits, Method of traffic monitoring for DDoS attacks detection in e-health systems and networks, CEUR Workshop Proceedings 2255 (2018) 193–204. URL: https://ceur-ws.org/Vol-2255/paper18.pdf.

[6] V. Kharchenko, I. Chyrka, Detection of airplanes on the ground using YOLO neural network, in: Proceedings of 17th International Conference on Mathematical Methods in Electromagnetic Theory (MMET), IEEE, Kyiv, Ukraine, 2018, pp. 294–297. doi: 10.1109/MMET.2018.8460392.

[7] Y. Averyanova, et al., UAS cyber security hazards analysis and approach to qualitative assessment, In: S. Shukla, A. Unal, J. Varghese Kureethara, D.K. Mishra, D.S. Han (Eds.), Data science and security, volume 290 of Lecture Notes in Networks and Systems, Springer, Singapore, 2021, pp. 258–265. doi: 10.1007/978-981-16-4486-3_28.

[8] O. Tachinina, O. Lysenko, I. Alekseeva, V. Novikov, Mathematical modeling of motion of iron bird target node of security data management system sensors, CEUR Workshop Proceedings 2711 (2020) 482–491. URL: https://ceur-ws.org/Vol-2711/paper37.pdf.

[9] O. I. Lysenko, S. V. Valuiskyi, O. M. Tachinina, S. L. Danylyuk, A method of control by telecommunication airsystems for wireless AD HOC networks optimization, in: Proceedings of 2015 IEEE International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD), IEEE, Kyiv, UKraine, 2015, pp. 182–185. doi: 10.1109/APUAVD.2015.7346594.

[10] M. Zaliskyi, et al., Heteroskedasticity analysis during operational data processing of radio electronic systems, in: S. Shukla, A. Unal, J. Varghese Kureethara, D.K. Mishra, D.S. Han (Eds.),

Data science and security, volume 290 of Lecture Notes in Networks and Systems, Springer, Singapore, 2021, pp. 168–175. doi: 10.1007/978-981-16-4486-3_18.
[11] J. S. Al-Azzeh, M. Al Hadidi, R. S. Odarchenko, S. Gnatyuk, Z. Shevchuk, Z. Hu, Analysis of self-similar traffic models in computer networks, International Review on Modelling and Simulations 10(5) (2017) 328–336. doi: 10.15866/iremos.v10i5.12009.
[12] O. Tachinina, O. Lysenko, I. Romanchenko, V. Novikov, I. Sushyn, Using Krotov's functions for the prompt synthesis trajectory of intelligent info-communication robot, in: M. Nechyporuk, V. Pavlikov, D. Krytskyi, (Eds.), Information Technologies in the Design of Aerospace Engineering. Studies in Systems, Decision and Control, volume 507, Springer, Cham, 2024. doi: 10.1007/978-3-031-43579-9_6.
[13] V. Tkachuk, Y. Yechkalo, S. Semerikov, M. Kislova, Y. Hladyr, Using mobile ICT for online learning during COVID-19 lockdown, Communications in Computer and Information Science, 1308 (2021) 46–67. doi: 10.1007/978-3-030-77592-6_3.
[14] O. Solomentsev, M. Zaliskyi, T. Herasymenko, O. Kozhokhina, Y. Petrova, Efficiency of operational data processing for radio electronic equipment, Aviation 23 (3) (2020) 71-77. doi: 10.3846/aviation.2019.11849.
[15] O. Lavrynenko, R. Odarchenko, G. Konakhovych, A. Taranenko, D. Bakhtiiarov, T. Dyka, Method of Semantic Coding of Speech Signals based on Empirical Wavelet Transform, in: Proceedings of 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies (AICT), IEEE, Lviv, Ukraine, 2021, pp. 18–22. doi: 10.1109/AICT52120.2021.9628985.
[16] O. M. Tachinina, O. I. Lysenko, I. V. Alekseeva, Algorithm for operational optimization of two-stage hypersonic unmanned aerial vehicle branching path, in: Proceedings of 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), IEEE, Kiev, Ukraine, 2018, pp. 11–15. doi: 10.1109/MSNMC.2018.8576319.
[17] M. Zaliskyi, S. Migel, A. Osipchuk, D. Bakhtiiarov, Correlation Method of Dangerous Objects Detection for Aviation Security Systems, CEUR Workshop Proceedings 3421 (2023) 1–11. URL: https://ceur-ws.org/Vol-3421/paper1.pdf.
[18] O. Lavrynenko, D. Bakhtiiarov, V. Kurushkin, S. Zavhorodnii, V. Antonov, P. Stanko, A method for extracting the semantic features of speech signal recognition based on empirical wavelet transform, Radioelectronic and Computer Systems 3 (2023) 101–124. doi: 10.32620/reks.2023.3.09.
[19] N. S. Kuzmenko, I. V. Ostroumov, K. Marais, An accuracy and availability estimation of aircraft positioning by navigational aids, in: Proceedings of 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), IEEE, Kiev, Ukraine, 2018, pp. 36-40. doi: 10.1109/MSNMC.2018.8576276.
[20] O. Solomentsev, M. Zaliskyi, O. Kozhokhina, T. Herasymenko, Efficiency of data processing for UAV operation system, in: Proceedings of 4th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD), IEEE, Kiev, Ukraine, 2017, pp. 27–31. doi: 10.1109/APUAVD.2017.8308769.
[21] O. Sushchenko, et al., Airborne sensor for measuring components of terrestrial magnetic field, in: Proceedings of IEEE 41st International Conference on Electronics and Nanotechnology (ELNANO), IEEE, Kyiv, Ukraine, 2022, pp. 687–691. doi: 10.1109/ELNANO54667.2022.9926760.