International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Design a New Neural Network Architecture Using a Layer of Neurons

Ibrahim Altarawni [a], Khalid Altarawneh [b], Obada Alhabashneh [c],Nadeem El-adaileh [d] , Mohammad Almajali [E] , Alaa Harasees [F]
Aqaba University of Technology, Aqaba, Jordan,
Mutah University, Karak, Jordan,
Mutah University, Karak, Jordan,
Mutah University, Karak, Jordan,
Mutah University, Karak, Jordan,
Aqaba University of Technology, Aqaba, Jordan

Abstract

Nowadays there are many different models of artificial neural networks. The difference between these models lies in the learning methods only, that is, in the rules for changing the parameters of the algorithm tuning, with or without links and side comments. While studying the general framework of network models, we see that the rules for obtaining the result and the mechanism for calculating the error can differ. For example, a multilayer realization might produce a threshold function when used as a classifier, or a linear function if used as an internal typeface.

Where this research came to discuss the possibilities of the standard representation of some models of artificial neural networks, which clarify and treat some of the characteristics of that representation. Which can be considered an essential element in the process of typical representation of these networks, where a new proposal is made during this representation by using a “layer” of neurons, in other words, using a group of neurons that work in parallel and perform the same functions for which they were set .

Keywords: model, layer, neuron, artificial, research, function, element, network.

1- Introduction:

Currently, a large number of various paradigms of artificial neural networks are k nown. However, the difference between paradigms often lies only in learning methods, that is, in the rules for changing the tuning parameters of the algorithm, and the presence or absence of lateral and feedback links. Within the framework of network paradigms, the rules for obtaining the result and calculating the error can differ. For example, a multilayer perceptron may output a threshold function when used as a classifier, or a linear function if used as an interpolator.

If we conditionally divide neural networks into “simple” and “complex”, then the architecture of “simple” networks is built based on several structural components. Such components are a processing element (neuron), a layer or a group of processing elements operating in parallel, and a set of layers that form the actual neural network. In some types of networks, the rules for signaling between neurons of a layer or layers of the network play a special role. "Complex" neural networks can usually be thought of as a combination of several "simple" ones.

The limited number of structural components of neural networks leads to a natural question about the possibility of a uniform description of various architectures. To answer this question, it is necessary to determine what elements will be required to describe networks, what properties such elements should have, and determine the rules for using these elements. It is obvious that a uniform representation of various types of neural networks should allow not only implementing existing paradigms, but also providing the means to obtain "new types" of networks.

One of the possible, and in our opinion, the most attractive ways of a uniform description of neural networks is their representation in the form of modular structures. This paper proposes a method for representing neural networks in the form of modular structures, and discusses the features of such a representation.

Almost all widely known and applied in practice neural networks can be used as classifiers. Therefore, without loss of generality, the idea of a modular representation can be considered on the example of a classification problem. For greater certainty, we will talk about neural networks trained “with a teacher”. That is, about such networks in which the required output is known for each image from the training set. The latter is important in the sense that it not only makes it possible to concretize the analysis, but also implies the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

presence of an additional module in the network representation.

The possibilities of modular representation are considered on the example of the most famous feed-forward networks, such as multilayer perceptron (MLP), radial basis functions (RBF) and support vector machines (SVM) [1, 2]. Recurrent, associative [3], and associative-projective networks [4] are considered as examples of a modular representation of networks with feedback and lateral connections. In conclusion, a trivial example of building a "new type" network is given.

2. Modular representation of feed-forward networks:

Most neural networks are based on the representation of a neuron as a processor element with adjustable parameters, for example, in MLP. By analogy with a biological neuron, a processing element has an arbitrary number of inputs and one output, and each input is associated with a customizable parameter called the connection weight. In some networks, such as SVMs, processor elements are not always explicitly distinguished, however, such networks can also be described using "neurons".

Let the vector X describe the image supplied to the inputs of the neuron. Denote the output of the neuron through y, and then the functioning of the neuron in general terms can be represented as follows:

$$y = F(G(,X)),$$

(1)

Where V - is the vector of connection weights, F- is the transfer function of the neuron, and G is some “weighting” function. Depending on whether a neuron model with or without a threshold is used, the size of the weight vector can be equal to or greater than the number of neuron inputs. There are neural networks for which more than one tuning parameter per connection can be used. An example of such a network is discussed below.

The "weighting" function is a scalar function of a vector argument. The type of "weighting" function determines the properties of the neuron in the space of inputs. For example, in the most common neuron model, the weighting function is defined as the scalar product of the input vector and the weight vector, given the threshold:

$$S_P(v,x) = v_o + (v',X) = v_o + \sum_{i=1}^{N} v,x\ ,$$

(2)

Where N is the number of neuron inputs and 0 , v is the threshold. The index p - "perceptron" in the name of the function was introduced for the convenience of referring to various "weighting" functions. The result of calculating function (2) is proportional to the distance to the hyper plane defined by the normal vector 'V { v ,...,v } = 1 N and offset 0, v in the space of neuron inputs.

For neurons using the weighting function (2), the most commonly used transfer function is the sigmoid

$$S_P(t) = \frac{I}{I + e^{-\propto t}}$$

In many other, practically important cases, the result of the weighting function is proportional to the distance to some point in the input space. For example, in RBF networks, the weighting function is represented as the Euclidean distance to the reference point, the coordinates of which can be considered as link weights

$$S_r(v,x) = (v_o,X) = v_o + \|v - x\| = v_o + \sqrt{\sum_{i=1}^{N} (v_1 - x_1)^2}$$

(3)

Where the index r means "radial" function. The threshold 0 v in function (3), by analogy with function (2), is introduced as a weighting factor, and in this case, it has the meaning of the "range" of the transfer function. In RBF networks, Gaussian-type functions are usually used, that is, we can consider a transfer function of the form

$$F_r(t) = e^{-\propto t}$$

It should be added that in networks with lateral connections, for example, in networks of associative memory, an additional “setting” input of the neuron u is sometimes considered, while before recalculating the network, y = u is taken and this input is usually not used in further calculations.

Temporarily abstracting from the learning algorithms, let us try to highlight what common in the operation (recalculation) of the selected is feed-forward networks, that is, in MLP, RBF and SVM. Without loss of generality, we can consider the classification problem into two classes. In this case, the neural network has one output. We introduce the following notation: X is the vector at the input of the network and z is the output of the entire network. We will use subscripts to number inputs or neurons in one layer, and upper ones to designate network layers.

Consider a two-layer perceptron. Let y be the output of the first layer neuron. At the output of the first layer, we get the vector { ,..., } 1 M Y = y y , where M is the number of neurons in the first layer. Then the operation of the two-layer perceptron is described by the expression

$$Z = FP^2\ \left(GP^2(V^2 - Y)\right) = FP^2(v_o^1 + \sum_{i=1}^{M} v_1^2.Fp^i\left(Gp^i(v_i^1.X)\right)))$$

In the two-class classifier mode, the output of the perceptron (the desired class) is determined by comparison with some threshold T . The input vector will be assigned to the first of two classes if z < T , otherwise - to the second one. Therefore,

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

the non-linearity at the output in this case is unnecessary, and for the output neuron, you can use the linear transfer function Fp(t) = t . Therefore, the operation of a two-layer perceptron can be described by the expression

$$z = v_0^2 + \sum_{i=1}^{m} \quad v_1^2 . F_p(S_P(S_P(\, v_1' . X)) \tag{4}$$

RBF and SVM networks are based on the use of "support vectors", that is, points in the input space chosen in a certain way. Let M reference points be chosen, then the operation of the RBF network is described by the expression.

$$z = v_i + \sum_{i=1}^{m} \quad v_i . F_r(S_r(v_i . X)) \tag{5}$$

And the operation of the SVM network

$$z = v_0 + \sum_{i=1}^{m} \quad v_i . K(v_i . X)) \tag{6}$$

Where K is the so-called kernel function and has a wide range of representations.

However, it turns out that the kernel functions used in practice can be represented in the form (1), while one of the functions (2) or (3) will be used as the weighting function. An example is the Gaussian kernel:

$$K(v, x) = e^{\varphi \| v - x \|^2}$$

Which exactly matches one of the RBF implementations, or a polynomial kernel:

$$K(v, x) = (1 + (v . x))^d$$

Which in the case of d = 1 gives a neuron of the "perceptron" type.

Comparison of expressions (4) - (6) shows that in the sense of work (direct recalculation or examination) SVM is a generalization of RBF and a two-layer perceptron. However, RBF and SVM remain different neural network paradigms because they use different rules for generating a set of support vectors and setting the weights of the output neuron.

Returning to the original task, namely, to the representation of neural networks in the form of modular structures, we see that the direct propagation networks under consideration can be described using two layers (modules) with the following rules for the operation of module elements:

$$y = F\big(G(v', x)\big) and \;\; z = F_p(S_p(v^2, y)) \tag{7}$$

In principle, the networks under consideration allow a modular representation in the form of sets of neurons, and each of these networks requires no more than two types of processor elements. There are several approaches to describing neural networks in the form of sets of processor elements, for example, descriptions using graphs [5]. However, there are associative networks that require taking into account the cooperative behavior of neurons within the layer.

Therefore, it is proposed to consider the layer as the minimum element of the modular representation.

It is easy to check that, using the weighting function (3) and taking into account the cooperative behavior, the modular representation (7) describes such recognition methods as the method of potential functions, as well as the methods of nearest and k-nearest neighbors.

The first advantages of the modular representation of feed-forward networks stem from the principle of solving the classification problem using such networks. RBF and SVM networks are so-called classifiers with input space transformation. The essence of using these classifiers is that they find such a transformation of the input space $\Omega \rightarrow R$ that in the space R the classes become linearly separable, and a linear perceptron (linear separating rule) is used to determine the boundary between the classes.

It is proved that MLP and RBF are universal approximators. That is, if the boundary between classes exists, then a classifier can be constructed that exactly describes such a boundary [6–8]. However, there are two fundamental problems with the use of classifiers with the transformation of the original feature space. The first is the determination of the most appropriate transformation functions (in the case of RBF, these are the Fr functions) and the second, directly related to the first, is the number of support vectors needed to achieve linear reparability. There is a contradiction between the requirement of linear reparability and a limited number of support vectors.

To resolve this contradiction (reduce the number of support vectors), it is possible to use a compromise solution when, after transforming the space, in our case, using the "RBF module", not a linear, but, for example, a two-layer perceptron with a small number of neurons in the first (hidden) layer is used . Then one cannot require linear reparability in the new space.

Nevertheless, the modular representation makes sense if it allows you to describe the learning process of neural networks. MLP and RBF learning algorithms are presented in

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

[1], and SVM in [2]. Without going into a detailed description, we note the following: in the networks under consideration, the algorithms for changing the tuning parameters belong to the layer, although they require knowledge of the error at the output of the entire network. Therefore, in order to represent a neural network with a modular structure, it is necessary to determine what properties the module (layer) should have and what functions the modular network as a whole should support, regardless of the types of modules included in it.Let us define the properties that a layer and a modular network should have. In general, a fully functional layer should have the following properties:

1) Architecture - the layer has inputs, outputs and a certain number of neurons (processor elements) working in parallel. Here it should be noted that the parallelism of the work of neurons in the layer refers to the "external" observer and means only that at each moment of time the outputs of all neurons in the layer are determined.

2) Work - a direct recalculation algorithm, that is, the rules for obtaining a result at the outputs for the current values at the inputs.

3) Training - an algorithm for changing the settings and determining the error at each input by an error signal or known errors at each of the outputs of the module (layer), and also, which is very important, the rules for removing or adding inputs or outputs according to the number specified from the outside and / or according to the internal algorithm of the layer.

Consider the purpose and use of the functions of the layer (module). First, it should be noted that not all features are required. Depending on the type and purpose, the module may not contain neurons and may not have a learning algorithm, that is, it may not contain tuning parameters. In addition, the ability to determine the error at each input is a property of the learning algorithm of this module and determines the limitations in building a modular architecture.

Therefore, for example, if it is supposed to use an MLP layer with learning by error back propagation, then the subsequent module must determine the error for each of its inputs, since the learning algorithms of the MLP layer use the value of this error. For some implementations of the RBF layer in training mode, it is sufficient to simply know that there is an error on a given input.

In the case of SVM modeling, there are two possibilities for organizing training in a modular representation: both the layer can receive a signal about the presence of an error at the output and the corresponding reference vector (neuron) will be deleted, or obtain information about the numbers of deleted inputs of the subsequent module. In this case, the corresponding outputs (support vectors) will be removed from the layer. The second option is universal, since the modular representation similarly implements the algorithms for removing unnecessary links (pruning), which are used in many feed-forward networks.

The presence of rules for removing inputs and outputs in modules ensures the implementation of mechanisms for removing unnecessary connections (tuning parameters) and neurons (processor elements). The simplest algorithm is to remove links that have zero weight.

In the example under consideration, where the second layer of each of the networks contains one neuron, deleting the connection means deleting the input of this layer, which means that the outputs of the corresponding neurons of the first layer do not affect the result of the network, and the corresponding neurons of the first layer can be deleted. The described mechanism corresponds to the last stage of SVM training.

The mechanism for adding inputs and outputs makes it possible to implement network options that either involve the creation of new support vectors by including a new part of the training set or based on specific heuristics. It is important that the number of inputs or outputs of a module affects only the number of external connections of neighboring modules but does not require knowledge of either the operation algorithm or the learning algorithm of adjacent modules. In addition, the number of inputs and outputs of the entire modular network remains unchanged.

Let the modules have the listed properties, and then it is not difficult to single out the functions that should be provided by the modular network as a whole. Moreover, it is obvious that the functions of the network should not imply “knowledge” about which particular learning or work algorithm implements a particular module.

Then the modular network should provide:

1) Architecture - inputs, outputs and a number of modules (layers) connected by directional connections. The direction of the links specifies the direct recalculation of the network.

2) Work - the sequence of recalculation of modules and the rule for generating the output of the entire network.

3) Training - obtaining information about an error in the network, the rules for propagating an error through the network, the sequence of recalculating modules during

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

training, and also ensuring the coordination of the number of connections in case of a change in the number of inputs or outputs for modules.

In addition to the above functions, for modeling some types of networks, rules for automatically adding or removing modules of a given type can be introduced.

3. Modular representation of networks with feedback and lateral connections:

Most networks with feedback and lateral connections have a natural representation in the form of modular

Structures. The most common networks with feedback and lateral connections are recurrent, associative and associative-projective networks.

Recurrent networks are most widely used in time series analysis, in sentence parsing problems, and others, usually where the data represents a certain sequence. Recurrent networks are built based on a multilayer perceptron and include all types of feedback. There are four types of recurrent networks, which are shown in Fig. 1. Networks differ only in the way they form feedback loops and usually have a two-layer architecture.

The Frasconi-Gori-Soda recurrent network [9] is shown in fig. 1a and is a network with local feedback. In this architecture, the output of each neuron in the hidden layer is connected to the input of the same neuron. In the architecture proposed by Narendra and Parthasarathy [10] (Fig. 1b), the entire network is covered by feedback, that is, the output of each neuron of the last layer is connected to each of the neurons of the first hidden layer. Elman [11] (Fig. 1c) considered the use of a feedback network covering only the hidden layer. That is, the output of each neuron of the hidden layer is connected to the inputs of all neurons of the same layer. Williams and Zipler [12] (Fig. 1d) considered the possibility of using a "fully connected" recurrent network, in which the output of each neuron of the network is connected to the inputs of all neurons on each layer.In all considered recurrent networks, feedback acts as a delay buffer for one cycle, that is, in addition to the input vector, the state of the outputs of the corresponding neurons at the previous recognition cycle is fed to the network neurons. There are modifications of recurrent networks in which the delay buffer performs a more complex function, for example, a delay with accumulation, when the current state of the neurons is added to the contents of the buffer, taken with a certain coefficient. The new state of the buffer is determined by the formula

$$B_{i+1} = \propto By_i + (1-\propto)B_i. \quad (8)$$

Where B is the state of the buffer, Y are the outputs of neurons at cycle t, and 0 1 < < α is the coefficient, which is often called the moment.

Often, the input of recurrent networks is not a separate element of the sequence, but a subsequence of a certain length. In this case, a conventional multilayer perceptron can also be used to recognize sequences, and the delay buffer is used outside the network to form the input vector. On fig. Figure 2 shows examples of using a multilayer perceptron (Fig. 2a) and a recurrent network (Fig. 2b) for sequence analysis.

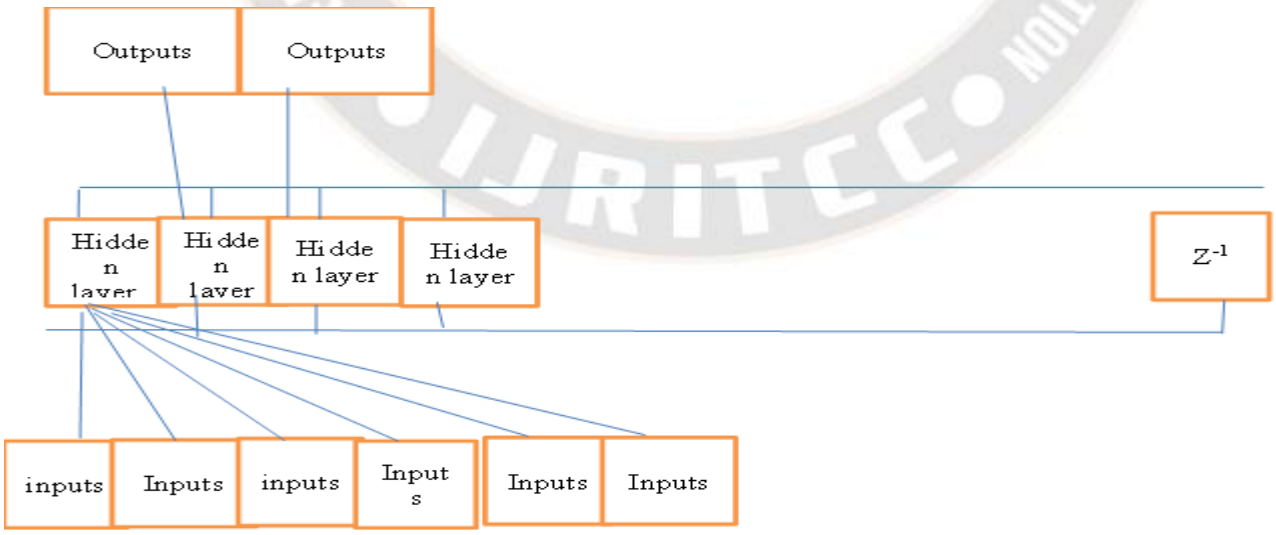

Figure 1-a

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

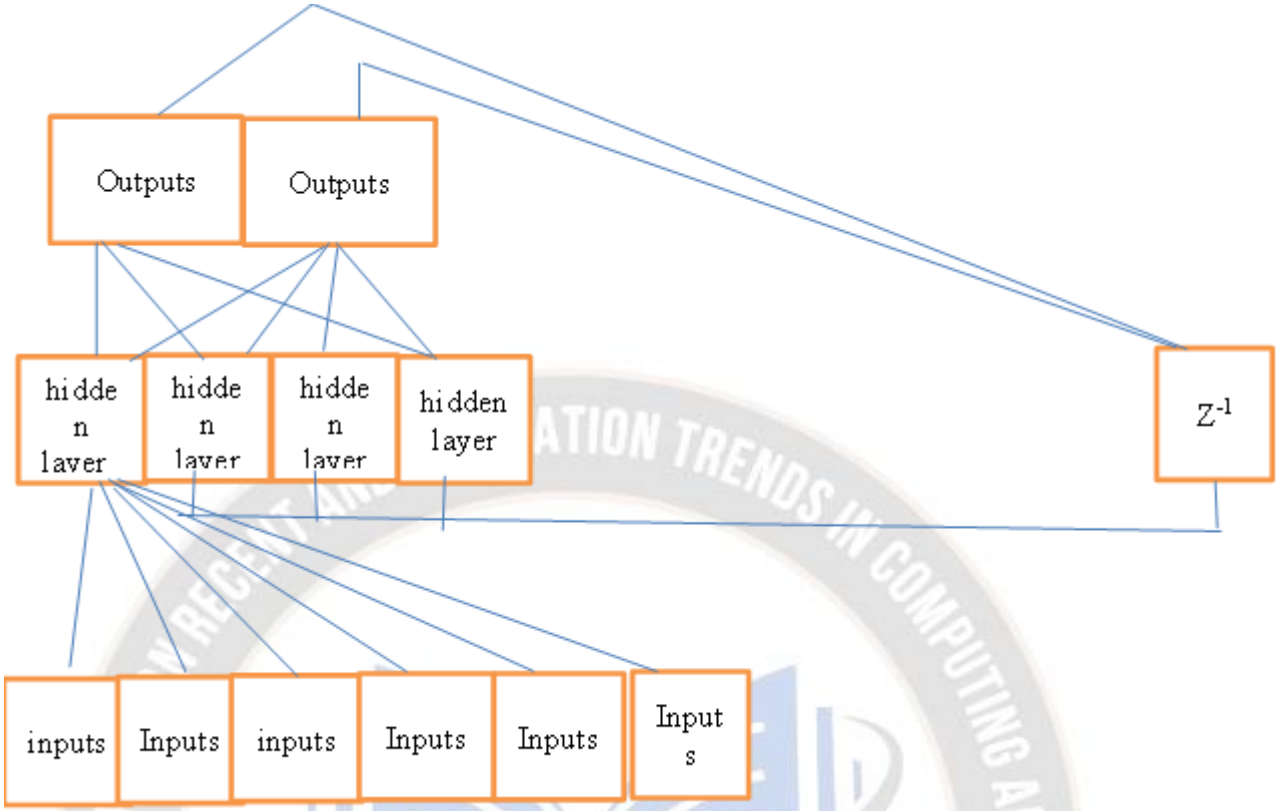

Figure 1- b

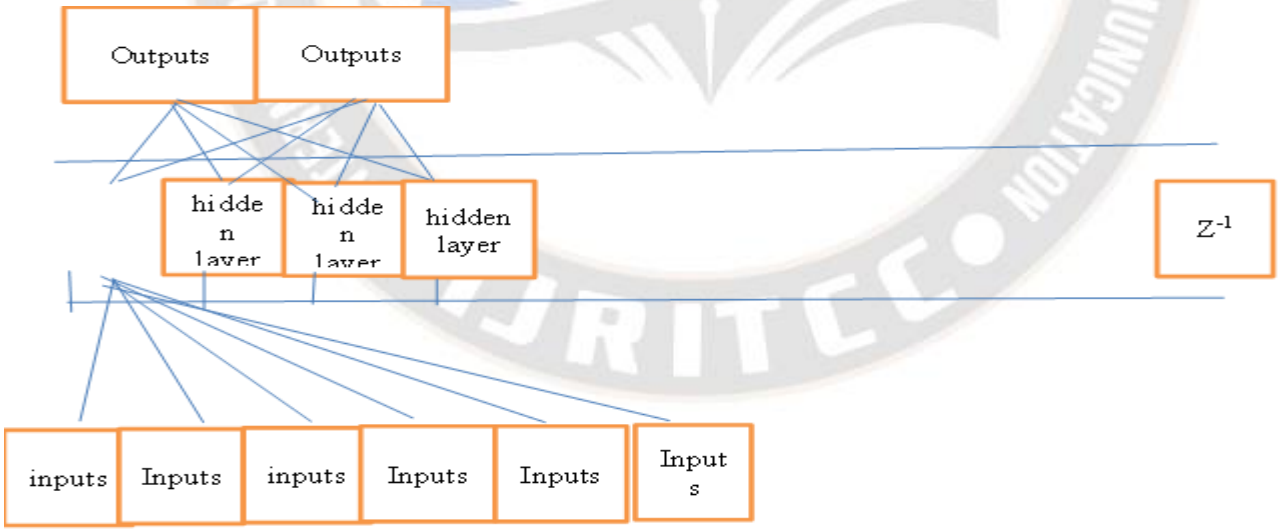

Figure 1- c

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
Article Received: *25 July 2023* ***Revised:*** *12 September 2023* ***Accepted:*** *30 September 2023*

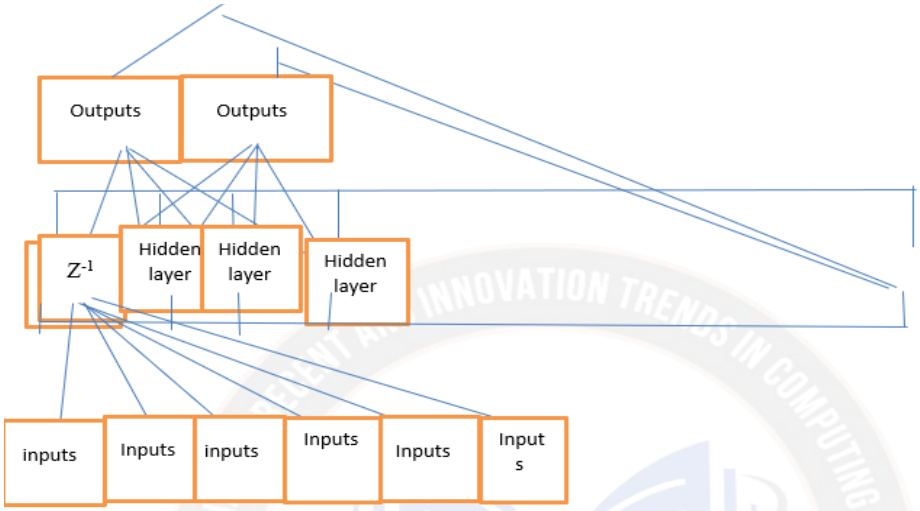

Figure 1- d.

Figure (1-a, 1-b, 1-c, 1-d): Examples of recurrent networks based on a multilayer perceptron.

Directly from Fig. 2 shows that, using only two types of layers - MLP and delay layer (Delay), it is possible to build both considered networks. An example of building these networks using modules that implement the corresponding layer is shown in fig. 3. Other architectures of recurrent networks also appear in an obvious way.

Hopfield-type associative networks contain one layer of neurons with lateral connections [8 -11]. In terms of functioning, the main difference between associative networks and networks of direct distribution and recurrent networks is not so much the type of connections as the process of network convergence. Network convergence is the process of successive recalculation of the network several times for the same total input action, which allows the output vector of the network to shift to the center of the attractor. In other words, after convergence, one of the previously stored vectors appears at the output of the associative layer.

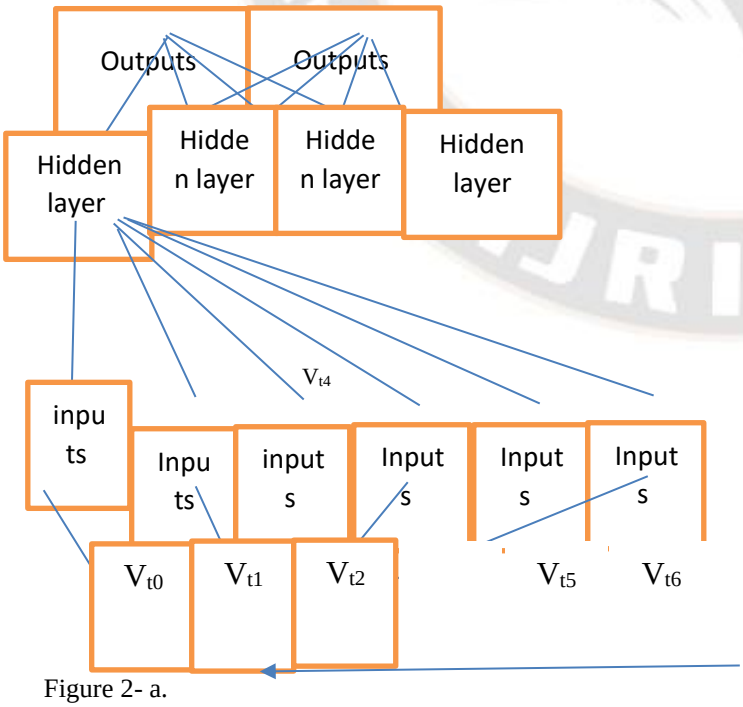

Figure 2- a.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

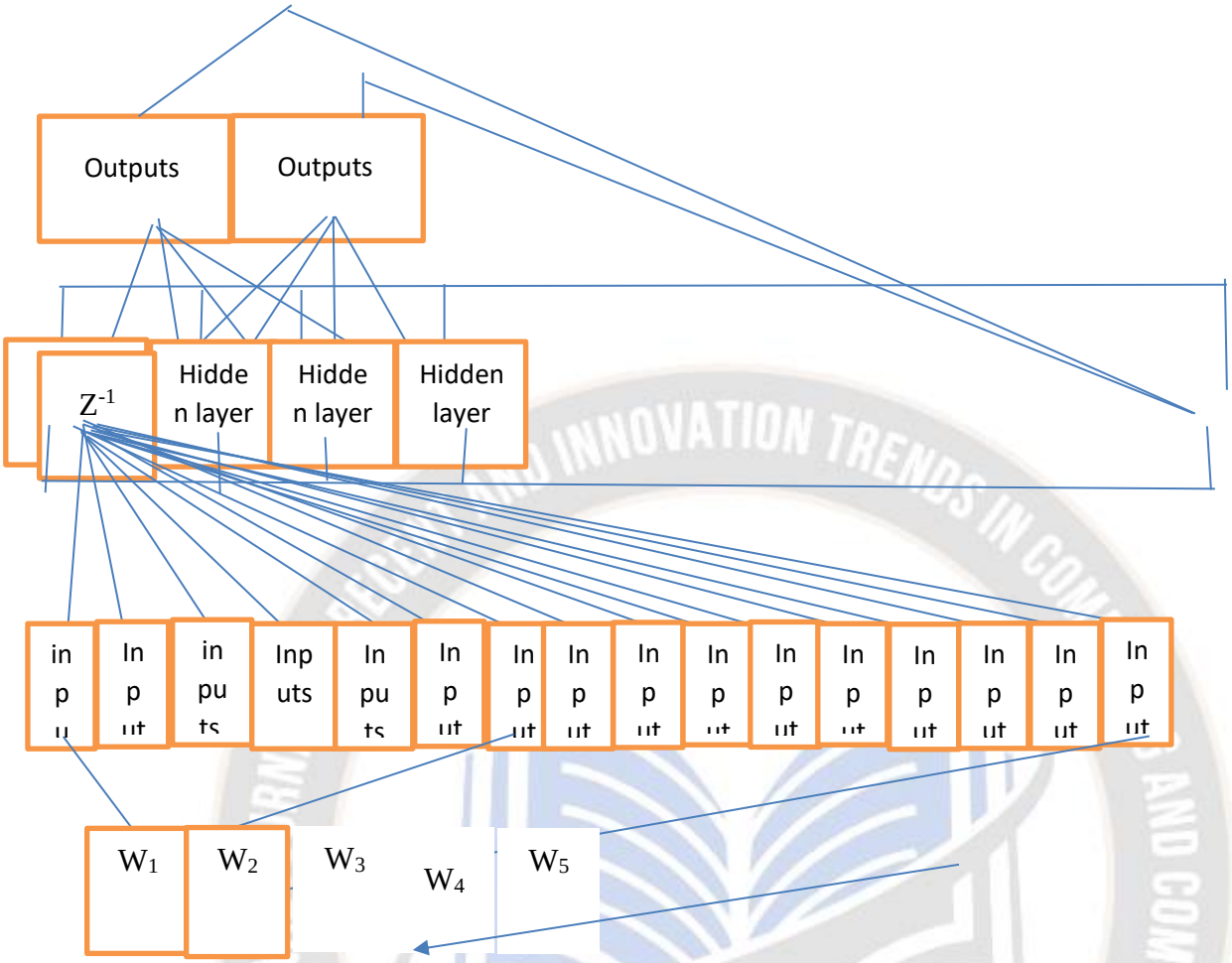

Figure 2- b

Figure 2 (a, b)- An example of using networks for time series analysis.

We said above that for the functioning of a modular neural network, we need some data flow control algorithm (router), which does not depend on the number or types of modules included in the network. One of the requirements for the router is the ability to automatically detect the presence of cycles in the "designed" modular network and the ability to set the parameters for working with each of the cycles. The parameters of the cycle, in fact, are the number of recalculations of modules within the cycle per one input vector or a sign of stopping the recalculation. Here the input vector has the meaning of the vector given to the input of the cycle, and not the entire network. Cycles in modular networks are considered in more detail in [12 -15].

In Hopfield-type associative networks, the algorithm of neuron operation is the same as the algorithm used in MLP. It turns out that if we can set the sign of the end of the cycle count, then in the modular representation, in order to implement this type of associative networks, we do not need to introduce an additional type of layer. If in fig. If we remove the output layer of neurons, we get an implementation of associative memory, where for one input vector; recalculation in the MLP-delay-MLP cycle is performed until the activity of neurons at the output of the MLP layer stops changing.

In the case of associative-projective networks [16- 20], the modular representation is generally natural, since in networks of this type not only the functions of each of the layers are defined (in these networks there are several types of layers that differ in the recalculation algorithm and functions performed), but almost arbitrary data flows between layers.

4. Using the Modular View:

The advantages of modularizing recurrent and social neural networks, as well as feed-forward networks, follow from the way the architecture is built in a modular implementation. Since the network for solving a specific problem is no longer implemented by a single algorithm, but is presented as a unified set of algorithms related exclusively to the layer, then, having the means to dynamically (or interactively) change the type of layer used, arbitrarily combine layers and set the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

direction of data flows, we get the opportunity to quite simply "design" a neural network that most successfully solves a specific applied problem.

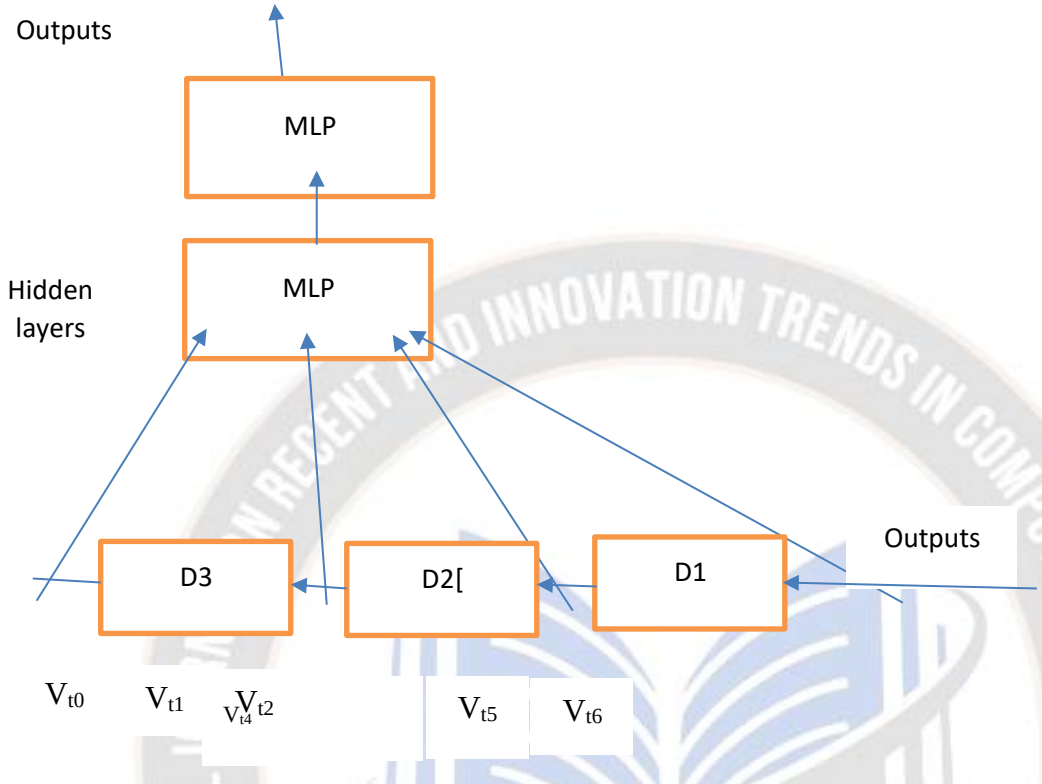

Figure 3-a

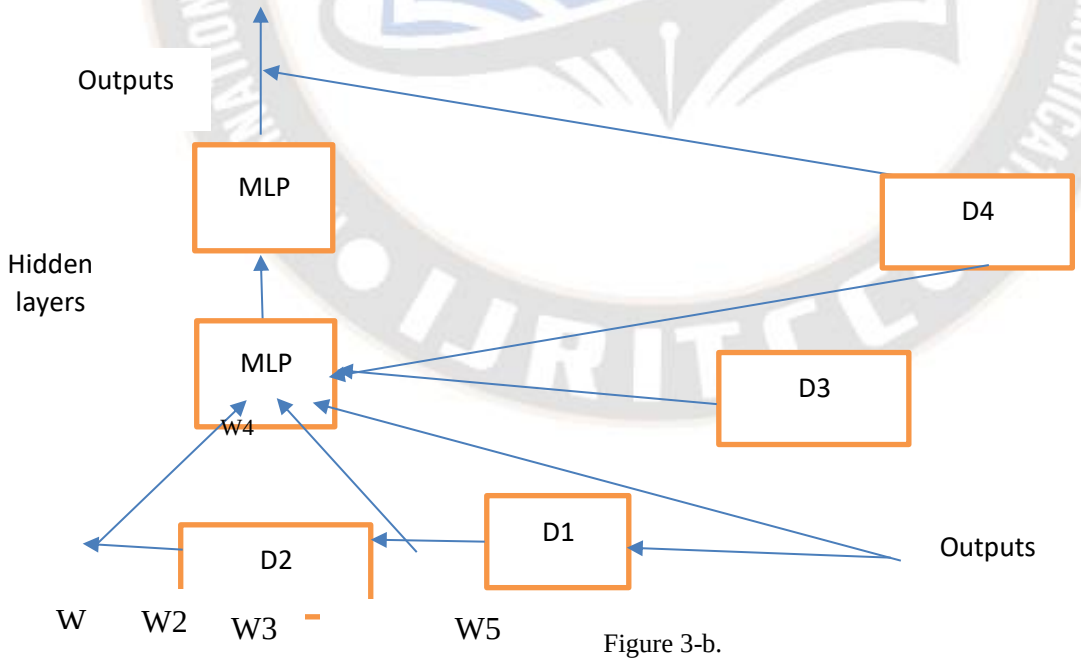

Figure 3-b.

Figures 3(a ,b): Example of Modular Networks for Time Series Analysis.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

A tool that allows you to interactively design a neural network from modules for solving applied problems is CAD INN [20].

Consider a model problem illustrating the advantages of a modular representation. As an example, take the classification problem known as XOR. The problem is formulated as follows: four points with coordinates (0.0), (1.1), (0.1) and (1.0) are given on the plane. The first two points belong to one class, the third and fourth to another. At one time, Minsky and Papert showed that a single-layer perceptron does not solve this problem, and it has become a classic for testing neural networks. It is known that the minimal architecture of MLP (Fig. 4a), solving this problem, contains two neurons and five connections (weights or adjustable parameters).

However, if instead of using "classical" neural networks, we put two RBF layers in series, then we need only two neurons and three connections to solve the problem (Fig. 4b). In this case, the links are three adjustable parameters that are the coordinates of the centers of the support vectors. In our case, the transfer function can be practically any, in particular, linear. Any of the four points can be chosen as a reference vector in the first layer. Let a point with coordinates (0,0) and a linear transfer function be chosen. Then at the output of the neuron of the first layer for 4 input vectors (coordinates of points) we will have three points now in one-dimensional space (on the line), with coordinates (0), (2) and (1). Both points of the second class at the output of the layer are indistinguishable. Taking as a reference vector for the second RBF layer a point with coordinate (1) and a linear transfer function, as well as a threshold of 0.1 (3), we obtain a modular network that solves the problem.

The "new type" architecture shown in Fig. 4b quite fully illustrates the potential of the proposed approach to modeling neural networks [21. 22].

Known neural network paradigms do not provide the ability, for example, to use RBF neurons or layers sequentially. This is due to the problems of constructing learning algorithms for such a network. However, when solving applied problems, by examining data using known types of networks and learning algorithms, it is possible to determine what kind of transformation of the input feature space "simplifies" the solution. By isolating individual layers in trained networks and combining them with layers of other types, we get a powerful tool for solving classification problems, and more broadly - recognition, of arbitrary complexity.

5. Conclusions:

Through the foregoing, we see that this research analyzes several different models of neural networks used on a wide range of applications. Accordingly, it becomes clear that the main component of the structures of these networks is a layer of neurons.

It has been established that if we consider layers as independent elements of the architecture, the number of different layers required to model the studied neural network models is less than the number of models themselves.

Based on the analysis carried out, it is proposed to consider the layers of neurons as units or elements of the architecture of modular networks. The requirements for the properties of the layers and the functioning of the modular network as a whole are formulated. A typical example shows that the representation of neural networks in the form of unit layers allows not only to model well-known models, but also to build structures that have not been studied before.

Note that no set of studied units gives a viable neural network. The use of modules in the construction of neural networks of various architectures requires the introduction of certain rules and restrictions. The rules for using modules when building neural network architectures will be discussed in subsequent publications.

Of particular interest are the problems of training modular networks with an arbitrary composition of modules. In an arbitrary network, some modules can be trained with a teacher, some without, and some modules may not require training at all. The latter type includes the principal component method, which allows one to reduce the dimension of the original feature space.

Most developers train modules separately and obtain a general solution to the problem as a combination of solutions from individual networks. However, cooperative learning is of great interest, when the results at the outputs of some modules are interpreted as signals for learning others. We see the continuation of theoretical research in the field of modular neural networks, in particular, the study of cooperative learning methods.

References:

1- Safwan Al Salaimeh, BILJANA STOJAN ILIC, Management of telecommunication operator services in Serbia – Case study Eastern Serbia, WSEAS TRANSACTIONS on BUSINESS and ECONOMICS, VOL. 9, 2022.
2- Prof. Safwan Al Salaimeh, Dr. Amer Abu Jassar, Dr. Mohammad Salim Al Hababsah, Improved

Algorithm For Creating An Optimized Network Diagram, Test Engineering and Management, Volume 83, Issue Jan – Apr. , 2021

3- Vardan Mkrttchian and Safwan Al Salaimeh, About Performance of the Computer Education System to Present in Natural Language, chapter of book "Advances in Engineering Research" vol 40, 2020.

4- 1. End to end learning for self-driving cars / M. Bojarski [et al.] // arXiv preprint arXiv:1604.07316. – 2016.

5- Smadi, T. A., & Zureiqat, M. A. (2017). High-Speed Small-Purpose Parallel Hybrid Architecture of Summator for Calculation Back 3x in Eighth Coding. Eastern European Scientific Journal, (3), 19-31.

6- Quantization and training of neural networks for efficient integerarithmetic-only inference / B. Jacob [et al.] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2018. – P. 2704–2713.

7- Hussein, A. L., Trad, E., & Al Smadi, T. (2018). Proactive algorithm dynamic mobile structure of Routing protocols of ad hoc networks. IJCSNS, 18(10), 86.

8- 5.Low-bit quantization of neural networks for efficient inference / Y. Choukroun [et al.] // 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). IEEE. – 2019. – P. 3009–3018.

9- Haq: Hardware-aware automated quantization with mixed precision / K. Wang [et al.] // Proceedings of the IEEE conference on computer vision and pattern recognition. – 2019. – P. 8612–8620.

10- Quantized convolutional neural networks for mobile devices / J. Wu [et al.] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. – 2016. – P. 4820–4828.

11- Krishnamoorthi R. Quantizing deep convolutional networks for efficient inference: A whitepaper: arXiv preprint arXiv:1806.08342. – 2018.

12- Mobilenets: Efficient convolutional neural networks for mobile vision applications / A.G. Howard [et al.] // arXiv preprint arXiv:1704.04861. – 2017.

13- A. Smadi, H. A. Al Issa, E. Trad, and K. A. A. Smadi, "Artificial Intelligence for Speech Recognition Based on Neural Networks," Journal of Signal and Information Processing, vol. 06, no. 02, pp. 66–72, 2015, doi: 10.4236/jsip.2015.62006.

14- Safwan Al Salaimeh, Khaldoun Albesuol, Khaled Batiha, Combined method of routing multimedia data in computer networks, International Journal of Advanced Research in Computer Science, vol.2, No.5, sep. – oct., 2011, India.

15- Al Salaimeh safwan, Hanna jabber, 2008, Weights Adjustment Neural Networks. European Journal of Scientific Research (EJSR) / Vol.21 No.2, pp.314-318, (2008), UK.

16- Safwan Al Salaimeh, Information Technologies of the Distributed Applications Design, Leonardo Journal of Science, Issue 10, Jan – June, 2007.

17- A. S. Takialddin, O. I. Al-Agha, and K. A. Alsmadi, "Overview of Model Free Adaptive (MFA) Control Technology," IAES International Journal of Artificial Intelligence (IJ-AI), vol. 7, no. 4, p. 165, Oct. 2018, doi: 10.11591/ijai.v7.i4.pp165-169.

18- T. Al Smadi, "Application of Fuzzy Logic to Cognitive Wireless Communications," Journal of advanced Sciences and Engineering Technologies, Jan. 2019, doi: 10.32441/jaset.02.01.03.

19- A. S. Takialddin, K. Al Smadi, and O. O. AL-Smadi, "High-Speed for Data Transmission in GSM Networks Based on Cognitive Radio," American Journal of Engineering and Applied Sciences, vol. 10, no. 1, pp. 69–77, Jan. 2017, doi: 10.3844/ajeassp.2017.69.77.

20- Galinskaya A.A. Architecture and training of modular classifiers for applied problems // Mathematical machines and systems. - 2003. - No. 2. - P. 77 - 86.

21- Distel R. Graph Theory. - Novosibirsk: Publishing House of the Institute of Mathematics, 2002. - 336 p.

22- Al Smadi, K. A., & Rababah, M. A. A. (2018). Analytical Survey: Speech Recognition Methods Used In Voice Recognition Techniques. Journal of Advanced Sciences and Engineering Technologies, 1(2), 1-8.