UDC 681.3.07:004.8
DOI: 10.15587/1729-4061.2025.328928

MLP-KAN: IMPLEMENTATION OF THE KOLMOGOROV-ARNOLD LAYER IN A MULTILAYER PERCEPTRON

Oleg Galchonkov
Corresponding author
PhD, Associate Professor*
E-mail: o.n.galchenkov@gmail.com
Oleksii Baranov
Software Engineer
Oracle Corporation
Oracle World Headquarters
Oracle Way, 2300, Austin, USA, 78741
Oleh Maslov
Doctor of Technical Sciences, Associate Professor
Department of Physics**
Mykola Babych
PhD, Associate Professor*
Illia Baskov
Senior Lecturer*
*Department of Information Systems**
**Institute of Computer Systems
Odesa Polytechnic National University
Shevchenko ave., 1, Odesa, Ukraine, 65044

The object of this study is neural networks used for categorizing objects in images. The task addressed in the work is to identify options for building a multilayer perceptron architecture that apply the Kolmogorov-Arnold layer and are characterized by the best ratio of classification quality and computational effort.

The paper proposes a modification to the multilayer perceptron (MLP) by replacing the first hidden layer with a Kolmogorov-Arnold layer. This allowed the use of the approximating properties of neurons and learning activation functions simultaneously. A feature of the designed MLP-KAN neural network, unlike the classical KAN network, is the use of only one activation function for each of the input elements. The training of activation functions is carried out on the basis of invariant radial basis functions, which are composed using learning weight coefficients. Such construction of the MLP-KAN neural network architecture allowed the use of typical libraries and optimizers for its training. In this case, unlike known analogs, there is no slowdown in the learning speed.

Experimental studies on the handwritten digit dataset (MNIST) have shown that MLP-KAN could provide higher classification quality with less computational effort. In particular, to obtain classification quality comparable to MLP, with the appropriate parameter setting, MLP-KAN requires 3.63 times less computational effort than MLP. This makes it possible to significantly improve the efficiency of image object classification devices built on microprocessors operating under an autonomous mode as part of robotic systems

Keywords: multilayer perceptron, neural network, Kolmogorov-Arnold network, weight coefficients, radial basis functions

Received 18.02.2025
Received in revised form 01.04.2025
Accepted 24.04.2025
Published 25.06.2025

How to Cite: *Galchonkov, O., Baranov, O., Maslov, O., Babych, M., Baskov, I. (2025). MLP-KAN: implementation of the Kolmogorov-Arnold layer in a multilayer perceptron. Eastern-European Journal of Enterprise Technologies, 3 (4 (135)), 34–41. https://doi.org/10.15587/1729-4061.2025.328928*

1. Introduction

The multilayer perceptron (MLP) is the basic building block for almost all neural networks. It has been proven to be a universal approximator [1], which is the theoretical justification for its use in a wide range of problems. However, in practical tasks there are always limitations on the amount of computing resources and the bit depth used. Therefore, a large number of neural network architectures have been developed with a much better ratio of performance quality to the required amount of computation. But at the output of such neural networks, there is usually always an MLP, which forms the resulting signal.

In [2], an alternative architecture (Komogorov-Arnold KAN networks) was proposed, which is also a universal approximator [3]. Naturally, the quality of approximation of any function and the required amount of computation depend on the type of this function and the approximation method used. MLP uses multi-input adders with trainable weight coefficients at the input and a nonlinear function at the output, called neurons. The nonlinear output function (activation function) is the same for all neurons in a layer in MLP. All activation functions in MLP do not change during training. In KAN, trainable nonlinear functions are used at the adder inputs instead of trainable weights, and there are no activation functions at the adder outputs.

In [2], the higher efficiency of using KAN, compared to MLP, was demonstrated for a number of problems. Examples of such problems are, for example, approximation of Bessel functions and solving partial differential equations. However, these problems are characterized by specific functions to be approximated. For them, the approximation used in KAN is more efficient. In [4], it was shown that using KAN in reinforcement learning problems allows one to obtain training quality comparable to using MLP, with significantly less computation. The study reported in [5] showed that for a wide range of machine learning problems, using MLP compared to KAN allows one to obtain better results with less computation. The exception is the symbolic representation of formulas. At the same time, simultaneous integration of MLP and KAN in the transformer architecture [6] allows one to obtain high results not only in the approximation of complex functions but also in image classification. Therefore, the development and study of a modified MLP architecture using KAN seems relevant. This could make it possible to combine the approximation capabilities of both architectures.

2. Literature review and problem statement

The use of dynamically adjustable activation functions in KAN makes it possible to demonstrate higher performance than MLP in problems where such signal approximation is more efficient. In [7], a modification of KAN for time series prediction is proposed. The two-layer structure of the neural network uses splines to analyze temporal patterns between successive data samples. It is shown that such an architecture can have a smaller number of trainable functions compared to the number of weight coefficients in MLP. For time series prediction, two architecture variants are proposed in [7]: T-KAN and MT-KAN. The first is designed to detect concept drift in time series. For this purpose, symbolic regression is used, which determines nonlinear relationships between past and current forecasts. This situation is typical for dynamically changing environments. The second architecture is designed to improve the accuracy of current forecasting of multidimensional time series. However, training such neural networks takes significantly more time. In this regard, in [7] it is recommended to use pure MLP for problems where time dependences change dynamically and fast adjustment to input signal parameters is required. In [8] a modification to the recurrent neural network is proposed by replacing the MLP at the output with KAN in the task of forecasting solar radiation and outdoor temperature. Experimental studies on real data have shown an improvement in the quality of forecasting with a smaller amount of calculations. However, the efficiency of using KAN in this problem is explained by the fact that for this type of signals, spline approximation is more effective than approximation in MLP.

In [9], a hybrid architecture for hyperspectral image classification is proposed by analogy with [10]. Unlike [10], which uses convolutional neural networks and image transformers, [9] proposes to sequentially use three-dimensional, two-dimensional, and one-dimensional KAN layers. That made it possible to obtain comparable classification characteristics with less computation. However, this architecture is specifically aimed at categorizing hyperspectral images and is redundant for simpler datasets. In [11], the KANDiff architecture is proposed to improve the quality of hyperspectral images, which uses a KAN and a diffusion neural network together. In it, a KAN is used at the input to merge hyperspectral and multispectral images, which makes it possible to eliminate differences in modal information. However, the image generated by the diffusion model contains residual noise. To suppress this noise and enhance the fusion effect, a special MergeCNN module is used, which provides a smoother and more accurate result. Experimental studies on publicly available datasets have shown improved image quality when solving problems of landing site search and resource exploration in the field of deep space exploration. However, such an architecture is not designed for categorizing objects in images and is redundant for it.

Using MLP in global satellite positioning systems of the visible range [12] makes it possible to increase the accuracy of positioning of moving objects. However, in the infrared range, images have significantly worse characteristics. To solve this problem, it was proposed in [13] to use a KAN layer at the output of the YOLOv.10 program. In addition, an additional neural network Swin Transformer is used in the system architecture, the main task of which is to extract global features from infrared images of small and medium-sized ships. Trainable activation functions in the KAN layer and the shifted window mechanism implemented in Swin Transformer together led to an increase in the ability of the model to focus on global features. Experimental studies have shown a significant increase in the accuracy of ship positioning at sea with less computation. However, this architectural solution is implemented specifically for the YOLOv.10 program and cannot be extended to the general case of the task of categorizing objects in images.

In [14], a modification to the convolutional network was proposed by completely replacing the MLP at the output with a three-layer KAN in the problem of intrusion detection in Internet of Things (IoT) systems. As a result, the modified convolutional network on test datasets showed higher performance with less computation compared to conventional convolutional networks, recurrent neural networks, and an autoencoder. Improvement was achieved for both binary and multi-classification tasks in terms of accuracy, confidence, and recall. However, the KAN in such an architecture received specific input signals and the effectiveness of replacing MLP with KAN in the general case is unclear. In [15], KAN layers were integrated directly into the convolutional neural network architecture, and the output was a traditional MLP. Experimental studies on the Fashion MNIST dataset showed 1.3 % worse results compared to a conventional convolutional network with less computation. However, it is not clear from the paper whether it is possible to obtain higher results using a modified neural network. In [16], a convolutional neural network was modified by introducing new layers Bottleneck Convolutional Kolmogorov-Arnold and Self KAGNtention. The first type of layers involves using a special design of an encoder, functional experts, and a decoder in parallel with the convolutional layers. The second type of layers uses a block similar to the KAN layer in parallel with the convolutional layers and consisting of an orthogonal basis decomposition, weighting with adjustable coefficients and an adder. This layer is a certain equivalent of the self-attention mechanism. This construction of the neural network makes it possible, with some types of architecture configuration, to slightly increase the quality of object classification in images, compared to a conventional convolutional neural network. However, this was achieved at the expense of a very large increase in the volume of calculations from several times to tens of times. It should be noted that, in addition to a significant increase in the volume of calculations, such an architectural solution is intended for modifying only convolutional networks and cannot be used to modify MLPs.

The MLP-Mixer neural network shows high results in categorizing objects in images. Its architecture is based on the repeated use of MLP in processing the matrix derivative of the original image by rows and columns. In [17], a modification to this architecture using KAN layers was proposed. Despite the fact that the results obtained were somewhat better than those of MLP-Mixer, they were significantly lower than those of RES-Net-18.

In [18], a modification of the U-Net network was proposed, which consists of using a combination of KAN layers and convolutional layers at the output of the encoding part and at the input of the decoding part of the U-Net network. Using this network to process synthetic aperture radar data in a flood monitoring task makes it possible to better capture complex nonlinear relationships and global features with less computation. However, such a combination of KAN layers and convolutional layers is specially designed for the U-Net architecture and cannot be directly transferred to other neural network architectures.

Thus, a large number of neural network modifications using KAN layers have been devised at present. However, all of them involve either a complete replacement of MLP with KAN, or the integration of KAN into a complex architecture. Moreover, a significant improvement in performance is observed mainly only in applications with specific input signals. Therefore, the issue of combining the approximating properties of MLP and KAN in the problems of categorizing objects in images requires further research.

It should be noted that when using KAN in any architecture, the technique of implementing the trained activation functions is of great importance. In a fundamental work [2], splines were used. However, in practical implementation, it is convenient to use a number of functions for approximation that are most suitable for this. In [19], 18 types of different polynomials were studied using the example of the task of recognizing handwritten letters from the MNIST data set. Polynomials created on the basis of various principles were specially considered, in particular, orthogonal polynomials, hypergeometric polynomials, combinatorial polynomials. For the used MNIST data set, the best results were obtained using Gottlieb polynomials. It is noted that for other data sets, a similar study should be carried out to obtain the best results. Replacing MLP with KAN showed the performance of all options but the classification quality was significantly lower than the best known results for MNIST. It should be noted that the MNIST dataset is general and has no specific features, so it is advisable to use universal functions for KAN to categorize objects from it. In [20], it is shown that radial basis functions (RBFs) have universal approximation characteristics and are one of the best replacement options for splines. In the paper, a new class of neural networks with radial basis functions in which smoothing factors are reduced by shifts is considered. With some restrictions on the activation function, these networks are able to approximate any continuous multidimensional function on any compact subset of d-dimensional Euclidean space. For a finite number of radial functions used, conditions are obtained that guarantee a given accuracy. This is best suited for use in a KAN layer for constructing trainable activation functions. However, in [20], the construction of KAN layers was not considered.

In [21], the problem of approximation of functions using neural networks using radial basis functions was solved. However, high results were achieved due to an additional clustering algorithm used to determine the best choice for the centers and width of the radial basis functions. In the work, approximation with shifted radial basis functions was also used. However, control over the centers of basis functions is a rather complex computational task and cannot be solved by typical optimizers used in training neural networks. If we add control over the width of the basis functions, then such an architecture falls into the category of theoretical and cannot find wide application in practice. In [22], an improved version of neural networks with radial basis functions is proposed, but the improvement was achieved due to additional processing of input signals with exponential functions. The radial basis functions themselves were not configured in this architecture and the approximation of the activation functions in the form of a KAN layer was not performed. In [23], neural networks with radial basis functions were used in the problem of pattern recognition. Experimental studies showed an improvement in the quality of recognition when the actual noise distribution differs from the Gaussian one. However, the improvement in performance was obtained not by learning the activation functions but by using a surrogate likelihood function. Some analog of the KAN architecture using radial basis functions is proposed in [24]. However, the main focus of this architecture is on removing and adding nodes in the hidden layer, rather than learning the activation functions. The contribution of each node to the output signal is determined using the normalization concept. In addition, a simple algorithm for controlling the width of the basis functions is proposed in the work. And the recursive least squares method is used to find the connecting weights. The results of the experimental study of the proposed architecture showed high quality approximation of nonlinear functions and forecasting the behavior of dynamic systems. However, such architecture does not seem promising for a wide range of practical tasks since there was no full tuning of activation functions, only adding and removing nodes in hidden layers depending on their contribution to the resulting characteristics. The most complete training of activation functions is implemented in [25]. The input signals from the output of the first stage of the ensemble classifier are transformed by radial basis functions. Then they are weighted by the trained weighting coefficients and summed up. Thus, the output values of the trained activation functions are obtained at the output of the adders. These values are fed to the output fully connected layer. It should be noted that the possibility of combining activation functions was not investigated in the work. In addition, the integration of two different approximation methods was not fully used since only a simple fully connected layer of 10 neurons was used. The KAN layer using radial basis functions is implemented there in the second stage of the ensemble classifier with stacking. Specific signals from the first stage are fed to the second stage. What results such an architecture will provide for general-type input signals is not investigated there.

All this allows us to state that it is advisable to design and investigate the architecture of a neural network that combines the approximating properties of MLP and KAN with input signals of a general type. The task to categorize objects in images from the MNIST set can be used as signals of this type.

3. The aim and objectives of the study

The aim of our work is to identify the effectiveness of combining MLP and KAN in the task to recognize objects in images from the MNIST dataset. This will make it possible to find in practice the options for the parameters of the combined neural network (MLP-KAN) that provide improved classification quality with less computation, compared to pure MLP.

To achieve the goal, the following tasks were set:

– to design the architecture of the MLP-KAN neural network;

– to study the dependence of classification quality on the number of Gaussian RBFs used to approximate one activation function;

– to investigate the dependence of classification quality and the amount of computation on the parameters of combining activation functions.

4. The study materials and methods

The object of our research is object classifiers on images from the MNIST dataset. The subject of research is the combined neural network MLP-KAN.

The main hypothesis of the study assumes that the joint use of two different approximation mechanisms could improve the resulting ratio of classification quality and the amount of classifier computations. The research was conducted on a general dataset. It is assumed that similar results could be obtained on other general datasets.

The dataset for research is MNIST [26] contains black and white images of handwritten digits, 28×28 pixels in size, 60,000 images for training and 10,000 for testing. The images belong to $c = 10$ classes. Examples of images are shown in Fig. 1.

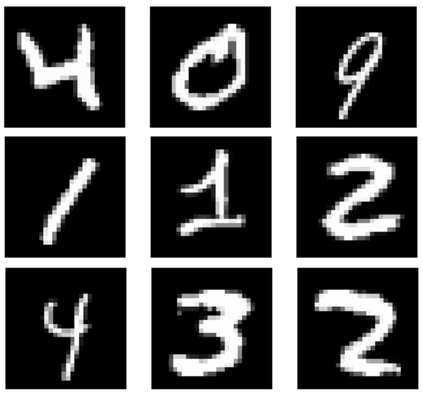


Fig. 1. Examples of images of digits from the MNIST dataset

The MLP with the following parameters was adopted as the basic multilayer perceptron:

1. Input vector, dimensionally – 784.
2. First hidden layer: 64 neurons, Relu activation function.
3. Second hidden layer: 64 neurons, Relu activation function.
4. Output layer: 10 neurons, softmax activation function.

The image of the basic MLP is shown in Fig. 2.

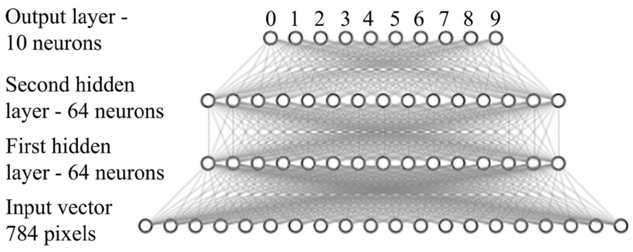

Output layer - 10 neurons
0 1 2 3 4 5 6 7 8 9
Second hidden layer - 64 neurons
First hidden layer - 64 neurons
Input vector 784 pixels

Fig. 2. Schematic of basic MLP

During training, the initial values were set as "glorot_uniform", the Adam optimizer. The training duration was 20 epochs. The best achieved classification quality was 0.9782. The number of trainable weight coefficients was 55050.

The programs for conducting the study were written in Python using the Keras library and are available on GitHub [27]. Training was performed on training data for 20 epochs. The classification quality was assessed on a set of testing images and the maximum value was recorded. The classification quality was defined as the ratio of the number of correctly recognized objects in the images to the total number of test images (10000).

5. Results of investigating the effectiveness of combining MLP and KAN

5. 1. Design of the MLP-KAN neural network architecture

In its pure form, the KAN architecture layer [2] assumes that each of the q input signals is transformed using q trainable activation functions. As a result, q^2 transformation results are obtained, which are fed to q adders. The outputs of the adders are the outputs of the KAN layer.

To combine the approximating properties of MLP and KAN in the MLP architecture shown in Fig. 1, the first hidden layer was replaced with a KAN layer.

Considering the universal approximating properties of RBF [20] and the ease of their implementation in the form of a pre-calculated table of values, they were used as a basis for trainable activation functions. Based on the results reported in [20], the basis set of functions is formed by shifting their centers.

Gaussian RBFs take the following form

$$\varphi_i\left(x_k\right) = e^{-\left(\left(x_k - x_{\min} + i \times d\right)^2 / \left(2 \times s^2\right)\right)}, \tag{1}$$

where $i = 0,\ldots,(m-1)$ is the function number, m is the total number of RBFs used; x_k is the k-th pixel of the input signal, for the MNIST set $k = 0\ldots784$; $x_{\min}$ is the minimum value of the input signal pixels, for the MNIST set it is 0; $d = (x_{\max} - x_{\min})/(m-1)$ is the distance between the centers of adjacent RBFs; $x_{\max}$ is the maximum value of the input signal, for the MNIST set it is 255, and after the typical normalization of the input signals it is 1; $s = (x_{\max} - x_{\min})/8.45$ is the coefficient that specifies the RBF width.

The first operation in implementing the KAN layer is to calculate the m values for each pixel of the input signal x_k for the corresponding functions φ_i.

To implement an ideal KAN layer, for each pixel of the input image from MNIST it is necessary to form 784 trainable activation functions, each of which takes the form

$$F_k\left(x_k\right) = \sum_{i=0}^{m-1} \varphi_i\left(x_k\right) \times w_{ki}, \tag{2}$$

where F_k is the activation function for the k-th pixel of the input signal; x_k is the k-th pixel of the input signal; w_{ki} is the i-th trainable weight coefficient for forming the activation function for the k-th pixel of the input signal.

The w_{ki} weight coefficients can be operated in the same way as the MLP weight coefficients. This allows the entire MLP-KAN architecture to be trained uniformly using the Keras library.

However, implementing as many activation functions as are required for an ideal KAN layer (784*g for MNIST, where g is the number of nodes in the next layer) seems redundant. In order to ensure a small resulting amount of computation, only one activation function is calculated for each pixel.

Thus, for the MNIST set, where the number of pixels in the image is 784, after the transformation operation using the RBF functions, $784 \cdot m$ values are obtained.

A typical form of the φ_i functions for $m = 6$ is shown in Fig. 3.

To take into account the mutual influence of the values at the outputs of the activation functions, we shall adjust the weighting coefficients not for each activation function separately, but for 2, 4, 8, etc. activation functions at once. In this case, the output of the Dence layer, which forms a combined activation function for pixels with numbers from $n1$ to $n2$, takes the following form

$$F_{n1-n2} = \left(\begin{array}{l} \sum_{i=0}^{m-1} \varphi_i\left(x_{n1}\right) \cdot w_{n1i} + \\ + \sum_{i=0}^{m-1} \varphi_i\left(x_{n1+1}\right) \cdot w_{(n1+1)i} + \ldots + \\ + \sum_{i=0}^{m-1} \varphi_i\left(x_{n2}\right) \cdot w_{n2i} \end{array} \right). \quad (3)$$

An example of the architecture of a KAN layer with a combination of two activation functions using 6 RBF functions and one neuron in the Dence layer is shown in Fig. 4.

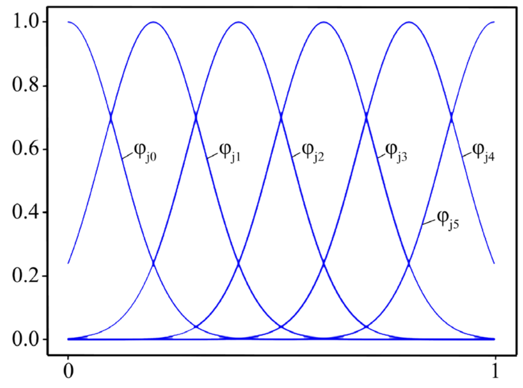


Fig. 3. Typical form of functions φ_i for $m = 6$

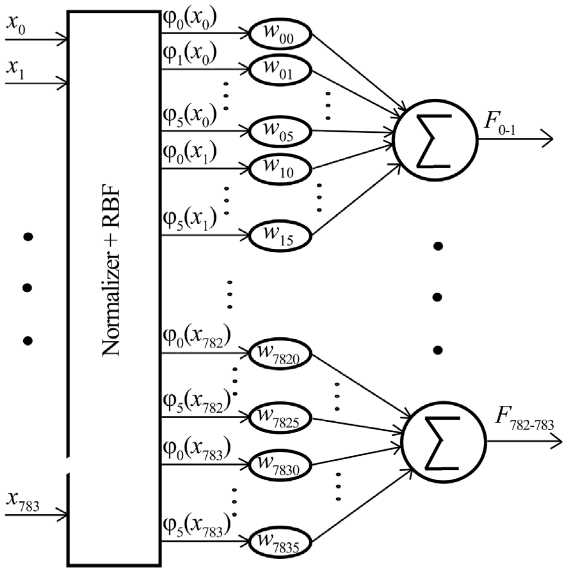


Fig. 4. Example of KAN layer with pooling of two activation functions

The input of the layer receives 784 pixels of the input image x_0, ..., x_{783}. Then they are normalized and transformed by RBF functions. This gives 784·6 values, which are multiplied by the weight coefficients w and added in the corresponding adders. The set of weight coefficients combined with the adder is a Dence layer, which does not use an activation function. Fig. 3 shows the combination of 2 activation functions. For this purpose, 392 Dence layers are used, each of which contains 1 neuron and has 12 inputs with the corresponding trainable weights. 382 output signals of the KAN layer are then fed to the second hidden layer of the MLP and from it to the output layer.

The proposed architecture of the KAN layer makes it possible to combine an arbitrary number of activation functions and have not only 1 neuron but also more in the Dence layers. The number of RBF functions can be varied for better classification quality of the corresponding types of input signals. Finding RBF values from input signals does not require performing arithmetic operations, and in many practical situations there is no need to normalize input signals. It is enough to first calculate and save tables of values, and then simply find the necessary values from the table. For example, for the MNIST set, pixel values are specified by 8 binary bits. Therefore, to find one RBF function, it is enough to have a table with 256 values in memory. If 6 RBF functions are used to approximate, then it is necessary to have 6 tables in memory, each with 256 values. Similarly, if this leads to an improvement in the quality of processing input signals, RBF functions can be easily replaced with any other type of basis functions while maintaining the principle of implementation in the form of tables.

5. 2. Investigating the dependence of classification quality on the number of Gaussian RBFs in MLP-KAN for the MNIST dataset

Considering that the number of possible combinations of parameters of the MLP-KAN neural network is large, the study was conducted for three different variants. The results are given in Tables 1–3. The following notations were used: k_o is the coefficient of combining activation functions in the KAN layer, k_n is the number of neurons in the Dence KAN layer, k_c is the number of neurons in the hidden layer of the MLP-KAN network (the 2nd hidden layer of MLP, which remained in the MLP-KAN structure), k_{RBF} is the number of RBF functions used.

Table 1

Classification quality and the number of trained weighting coefficients when $k_o = 2$, $k_n = 2$, $k_c = 64$

k_{RBF}	Quality of classification	Number of trained weights
3	0.9712	55594
4	0.9741	57162
5	0.9739	58730
6	0.9749	60298
7	0.9764	61866
8	0.9750	63434

Table 2

Classification quality and the number of trained weighting coefficients when $k_o = 4$, $k_n = 1$, $k_c = 64$

k_{RBF}	Quality of classification	Number of trained weights
3	0.9758	15610
4	0.9767	16394
5	0.9753	17178
6	0.9774	17962
7	0.9756	18746
8	0.9752	19530

Table 3

Classification quality and the number of trained weighting coefficients when $k_o = 8$, $k_n = 1$, $k_c = 64$

k_{RBF}	Quality of classification	Number of trained weights
2	0.9729	8554
3	0.9750	9338
4	0.9755	10122
5	0.9751	10906
6	0.9741	11960
7	0.9737	12474

As can be seen from Tables 1–3, in the first variant the best classification quality was obtained with the number of RBFs used $k_{RBF} = 7$, in the second variant – with $k_{RBF} = 6$, in the third variant – with $k_{RBF} = 4$. Therefore, for a better understanding of the dependences in the next chapter all calculations were performed for $k_{RBF} = 6$.

5. 3. Investigating the dependence of classification quality on the activation function pooling parameter in MLP-KAN for the MNIST dataset

The results of investigating the classification quality and the volume of calculations on the activation function pooling parameters are given in Table 4. The following notations are used in Table 4: the pooling coefficient – k_o, the number of neurons k_n in the Dence KAN layers (1 or 2) and the number of neurons k_c in the hidden layer (64 or 96). All results were obtained with the number of RBF functions used $k_{RBF} = 6$. In addition, Table 4 gives ratios of the number of trainable MLP coefficients (55050) and for MLP-KAN k_w, as well as ratio of the classification quality of MLP-KAN and MLP (0.9782) k_{qc}.

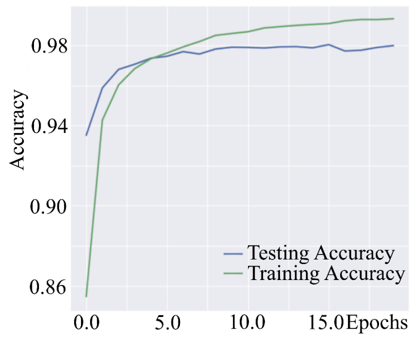
Table 4

Results of investigating classification quality and computation volume

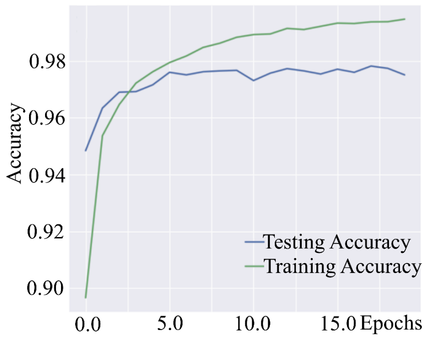
k_o	k_n	k_c	Number of trained weights	k_w	Quality of classification	k_{qc}
2	1	64	30506	1.8	0,9763	0.9981
2	1	96	43402	1.27	0.9784	1.0002
2	2	64	60298	0.91	0.9749	0.9966
2	2	96	87738	0.64	0.9776	0.9994
4	1	64	17962	3.06	0.9774	0.9992
4	1	96	24586	2.24	0.9793	1.0011
4	2	64	35210	1.56	0.9769	0.9987
4	2	96	48106	1.14	0.9805	1.0024
8	1	64	11690	4.7	0.9741	0.9958
8	1	96	15178	3.63	0.9782	1.0000
8	2	64	22666	2.43	0.9780	0.9998
8	2	96	29290	1.88	0.9782	1.0000
16	1	64	8554	6.44	0.9728	0.9945
16	1	96	10474	5.26	0.9744	0.9961
16	2	64	16394	3.36	0.9752	0.9969
16	2	96	19882	2.77	0.9786	1.0004

Table 4 shows that with the same classification quality of 0.9782 as MLP, MLP-KAN may have 3.63 times less computation volume. In addition, a higher classification quality of 0.24% can be achieved with 14% less computation volume. The best ratio of the number of trained coefficients and classification quality was obtained with the activation function pooling coefficient $k_o = 8$. Increasing the number of neurons in the Dence KAN layer with small k_o led to a deterioration in the classification quality, and with large k_o to an increase. Increasing the number of neurons in the hidden layer always led to an increase in the classification quality.

For comparison of the learning rate, Fig. 5 shows the learning curves for MLP and MLP-KAN.



a



b

Fig. 5. Learning curves: *a* — for MLP; *b* — for MLP-KAN

The vertical axis in Fig. 5 shows the classification accuracy (Accuracy), and the horizontal axis shows the number of training epochs (Epochs). Training was performed using the training data (Training Accuracy). Using the testing data, only the calculation of the result was performed with the weight coefficients achieved at that time. It is evident from Fig. 5 that the training speeds of MLP and MLP-KAN are almost identical. The maximum classification accuracy for MLP (Fig. 5, *a*) is 0.9782, for MLP-KAN (Fig. 5, *b*) – 0.9805.

6. Discussion of results based on investigating the ratio of classification quality and the volume of calculations of MLP-KAN

The proposed architecture of the KAN layer implementation differs from known ones by its significant flexibility. Firstly, the use of RBF (formula (1) and Fig. 3) is a universal but not limiting option. Instead of RBF, any other basis can be used if it is more suitable for the characteristics of the practical problem being solved. Secondly, the number of combined activation functions (formula (3) and Fig. 4) can be selected to obtain the best ratio of classification quality and the volume of calculations for a specific application. Thirdly, the implementation of the trained activation functions in the form of formulas (2) and (3) allows them to be easily implemented using traditional software libraries used to implement subsequent layers. Such a construction of the MLP-KAN architecture makes it possible to easily identify the most effective options for constructing the architecture when solving practical tasks.

The study of the efficiency of using different numbers of functions when approximating activation functions (Tables 1–3) revealed a complex nature of the dependence of classification quality on the number of functions for different parameters of the KAN layer. For the MNIST data set and using RBF as the average value to facilitate the interpretability of subsequent experiments, $k_{RBF} = 6$ was adopted. However, any practical application of MLP-KAN should include additional research to identify the best number of basis functions.

The results of our experimental studies (Table 4) show that by replacing the first hidden layer in MLP with a KAN layer, with an appropriate choice of parameters, it is possible to obtain a higher classification quality with a significantly smaller amount of computations. In particular, with the same classification quality of 0.9782, the amount of computations for MLP-KAN was 3.63 times less than that of MLP. With a higher classification quality of MLP-KAN (0.9786), the gain in computations was 2.77 times. In addition, it should be noted that the proposed modification of MLP-KAN is trained at the same speed as MLP (Fig. 5), in contrast to the slowdown reported in [7]. The results obtained are explained by the fact that the MLP-KAN neural network uses two types of approximation simultaneously – using weight coefficients in neurons and using trainable activation functions. It should be noted that the results were obtained on a general data set (MNIST), which allows us to hope that similar results could be obtained on other general data sets. An additional advantage of the proposed architecture of the MLP-KAN neural network is that it can be implemented using the widely used Keras library, in contrast to the special library proposed in [2].

A special feature of the MLP-KAN architecture is the use of joint processing of a small number of activation functions. The best results were achieved with joint processing of 8 activation functions. At the same time, the classical implementation of KAN proposed in [2] assumes joint processing in the nodes of the next layer of its activation functions from each node of the current layer. Such an MLP-KAN architecture, with an appropriate choice of parameters, provides a significantly smaller amount of calculations, compared to MLP, while maintaining the classification quality.

Thus, the proposed MLP-KAN architecture successfully solves the problem of joint use of the approximating properties of MLP and KAN for a general data set of MNIST. In this paper, we used the approximation of activation functions by radial basis functions. However, within the framework of the proposed architecture, any other approximating functions can be used if they better approximate the existing data set. There is no need to calculate these functions at each iteration. It is sufficient to initially calculate the tables of these functions with a sufficient number of samples and then simply select values from these tables.

The limitation of the proposed architecture is that it is based on MLP. MLP is used in various applications both independently and as an output element of almost all complex neural networks. However, there are known architectures that are more efficient than MLP, and it is of interest to develop the proposed approach to these architectures.

The disadvantage of the proposed architecture is a small increase in the quality of classification. Overcoming it is primarily associated with the use of more efficient architectures than MLP as base ones. The main difficulties in this direction are associated with the design of efficient architectures that simultaneously provide high quality of classification and a small amount of calculations.

7. Conclusions

1. The MLP-KAN neural network architecture has been designed, combining the approximating properties of MLP and KAN. The architecture is meant for the tasks to categorize objects in general-type images similar to MNIST.

2. Our study of the dependence of classification quality on the number of radial basis functions used showed that the best number depends on parameters of the entire neural network. The average best value for the MNIST data set is 6.

3. The results of experimental studies have demonstrated that with the same classification quality as MLP, MLP-KAN requires 3.63 times less computation when setting the appropriate parameters. MLP-KAN can provide higher classification quality than MLP but the gain in computation volume is reduced.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The research was conducted with financial support from the National Research Foundation of Ukraine, grant no. 131/0161.

Data availability

The manuscript has associated data in the data warehouse. Links are provided in the text of the paper.

Use of artificial intelligence

The authors used artificial intelligence technologies within acceptable limits to provide their own verified data, which is described in the research methodology section.

References

1. Hornik, K., Stinchcombe, M., White, H. (1989). Multilayer feedforward networks are universal approximators. Neural Networks, 2 (5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8
2. Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M. et al. (2024). KAN: Kolmogorov-Arnold Networks. arXiv. https://doi.org/10.48550/arXiv.2404.19756

3. Braun, J., Griebel, M. (2009). On a Constructive Proof of Kolmogorov's Superposition Theorem. Constructive Approximation, 30 (3), 653–675. https://doi.org/10.1007/s00365-009-9054-2
4. Guo, H., Li, F., Li, J., Liu, H. (2025). KAN v.s. MLP for Offline Reinforcement Learning. ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1–5. https://doi.org/10.1109/icassp49660.2025.10888327
5. Yu, R., Yu, W., Wang, X. (2024). KAN or MLP: A Fairer Comparison. arXiv. https://doi.org/10.48550/arXiv.2407.16674
6. He, Y., Xie, Y., Yuan, Z., Sun, L. (2024). MLP-KAN: Unifying Deep Representation and Function Learning. arXiv. https://doi.org/10.48550/arXiv.2410.03027
7. Xu, K., Chen, L., Wang, S. (2024). Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability. arXiv. https://doi.org/10.48550/arXiv.2406.02496
8. Gao, Y., Hu, Z., Chen, W.-A., Liu, M., Ruan, Y. (2025). A revolutionary neural network architecture with interpretability and flexibility based on Kolmogorov–Arnold for solar radiation and temperature forecasting. Applied Energy, 378, 124844. https://doi.org/10.1016/j.apenergy.2024.124844
9. Jamali, A., Roy, S. K., Hong, D., Lu, B., Ghamisi, P. (2024). How to Learn More? Exploring Kolmogorov-Arnold Networks for Hyperspectral Image Classification. Remote Sensing, 16 (21), 4015. https://doi.org/10.3390/rs16214015
10. Roy, S. K., Krishna, G., Dubey, S. R., Chaudhuri, B. B. (2020). HybridSN: Exploring 3-D–2-D CNN Feature Hierarchy for Hyperspectral Image Classification. IEEE Geoscience and Remote Sensing Letters, 17 (2), 277–281. https://doi.org/10.1109/lgrs.2019.2918719
11. Li, W., Li, L., Peng, M., Tao, R. (2025). KANDiff: Kolmogorov-Arnold Network and Diffusion Model-Based Network for Hyperspectral and Multispectral Image Fusion. Remote Sensing, 17 (1), 145. https://doi.org/10.3390/rs17010145
12. Liu, X., Tang, Z., Wei, J. (2025). Multi-Layer Perceptron Model Integrating Multi-Head Attention and Gating Mechanism for Global Navigation Satellite System Positioning Error Estimation. Remote Sensing, 17 (2), 301. https://doi.org/10.3390/rs17020301
13. Guo, L., Wang, Y., Guo, M., Zhou, X. (2024). YOLO-IRS: Infrared Ship Detection Algorithm Based on Self-Attention Mechanism and KAN in Complex Marine Background. Remote Sensing, 17 (1), 20. https://doi.org/10.3390/rs17010020
14. Abd Elaziz, M., Ahmed Fares, I., Aseeri, A. O. (2024). CKAN: Convolutional Kolmogorov-Arnold Networks Model for Intrusion Detection in IoT Environment. IEEE Access, 12, 134837–134851. https://doi.org/10.1109/access.2024.3462297
15. Bodner, A. D., Tepsich, A. S., Spolski, J. N., Pourteau, S. (2024). Convolutional Kolmogorov-Arnold Networks. arXiv. https://doi.org/10.48550/arXiv.2406.13155
16. Drokin, I. (2024). Kolmogorov-Arnold Convolutions: Design Principles and Empirical Studies. arXiv. https://doi.org/10.48550/arXiv.2407.01092
17. Cheon, M. (2024). Demonstrating the Efficacy of Kolmogorov-Arnold Networks in Vision Tasks. arXiv. https://doi.org/10.48550/arXiv.2406.14916
18. Wang, C., Zhang, X., Liu, L. (2025). FloodKAN: Integrating Kolmogorov-Arnold Networks for Efficient Flood Extent Extraction. Remote Sensing, 17 (4), 564. https://doi.org/10.3390/rs17040564
19. Seydi, S. T. (2024). Exploring the Potential of Polynomial Basis Functions in Kolmogorov-Arnold Networks: A Comparative Study of Different Groups of Polynomials. arXiv. https://doi.org/10.48550/arXiv.2406.02583
20. Ismayilova, A., Ismayilov, M. (2023). On the universal approximation property of radial basis function neural networks. Annals of Mathematics and Artificial Intelligence, 92 (3), 691–701. https://doi.org/10.1007/s10472-023-09901-x
21. He, Z.-R., Lin, Y.-T., Lee, S.-J., Wu, C.-H. (2018). A RBF Network Approach for Function Approximation. 2018 IEEE International Conference on Information and Automation (ICIA), 105–109. https://doi.org/10.1109/icinfa.2018.8812435
22. Panda, S., Panda, G. (2022). On the Development and Performance Evaluation of Improved Radial Basis Function Neural Networks. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 52 (6), 3873–3884. https://doi.org/10.1109/tsmc.2021.3076747
23. Seghouane, A.-K., Shokouhi, N. (2021). Adaptive Learning for Robust Radial Basis Function Networks. IEEE Transactions on Cybernetics, 51 (5), 2847–2856. https://doi.org/10.1109/tcyb.2019.2951811
24. Wu, C., Kong, X., Yang, Z. (2018). An Online Self-Adaption Learning Algorithm for Hyper Basis Function Neural Network. 2018 2nd IEEE Advanced Information Management,Communicates,Electronic and Automation Control Conference (IMCEC), 215–220. https://doi.org/10.1109/imcec.2018.8469684
25. Galchonkov, O., Baranov, O., Antoshchuk, S., Maslov, O., Babych, M. (2024). Development of a neural network with a layer of trainable activation functions for the second stage of the ensemble classifier with stacking. Eastern-European Journal of Enterprise Technologies, 5 (9 (131)), 6–13. https://doi.org/10.15587/1729-4061.2024.311778
26. LeCun, Y., Cortes, C. and Burges, C. J. C. (1998). The MNIST Database of Handwritten Digits. New York.
27. MLP-KAN. Available at: https://github.com/oleksii-m-baranov/MLP-KAN