One of the promising directions for improving the quality of object recognition in images and parallelizing calculations is the use of ensemble classifiers with stacking. A neural network at the second level makes it possible to achieve the resulting quality of classification, which is significantly higher than each of the networks of the first level separately. The classification quality of the entire ensemble classifier with stacking depends on the efficiency of the neural networks at the first stage, their number, and the quality of the classification of the neural network of the second stage. This paper proposes a neural network architecture for the second stage of the ensemble classifier, which combines the approximating properties of traditional neurons and learning activation functions. Gaussian Radial Basis Functions (RBFs) were chosen to implement the learned activation functions, which are summed with the learned weights. The experimental studies showed that when working with the CIFAR-10 data set, the best results are obtained when six RBFs are used. A comparison with the use of multilayer perceptron (MLP) in the second stage showed a reduction in classification errors by 0.45–1.9 % depending on the number of neural networks in the first stage. At the same time, the proposed neural network architecture for the second degree had 1.69–3.7 times less learning coefficients than MLP. This result is explained by the fact that the use of an output layer with ordinary neurons allowed us not to enter into the architecture many learning activation functions for each output signal of the first stage, but to limit ourselves to only one. Since the results were obtained on the CIFAR-10 universal data set, a similar effect could be obtained on a large number of similar practical data sets

Keywords: multilayer perceptron, neural network, ensemble classifier, weighting coefficients, classification of objects in images

UDC 681.3.07: 004.8
DOI: 10.15587/1729-4061.2024.311778

DEVELOPMENT OF A NEURAL NETWORK WITH A LAYER OF TRAINABLE ACTIVATION FUNCTIONS FOR THE SECOND STAGE OF THE ENSEMBLE CLASSIFIER WITH STACKING

Oleg Galchonkov
Corresponding author
PhD, Associate Professor*
E-mail: o.n.galchenkov@gmail.com
Oleksii Baranov
Software Engineer
Oracle Corporation
Oracle Way 2300, Austin, TX 78741, United States
Svetlana Antoshchuk
Doctor of Technical Sciences, Professor*
Oleh Maslov
Doctor of Technical Sciences, Associate Professor
Department of Physics**
Mykola Babych
PhD, Associate Professor*
*Department of Information Systems**
**Institute of Computer Systems
Odesa Polytechnic National University
Shevchenka ave., 1, Odesa, Ukraine, 65044

Received date 05.07.2024
Accepted date 17.09.2024
Published date 23.10.2024

How to Cite: *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*

1. Introduction

The accuracy of computer vision algorithms is enhanced by improving the architecture of neural networks, increasing their depth and processing width. However, very deep neural networks, due to the sequential processing of input signals, encounter problems with the stability of the learning process and low efficiency of parallelizing computational processes. This leads to the need to use supercomputers, which complicates the expansion of the area of their practical use. One of the promising approaches to overcoming these problems is the development of architectures with an increased processing width. This is most fully and consistently incorporated into the architecture of ensemble classifiers and, in particular, into ensemble classifiers with stacking [1]. This architecture assumes the presence of two processing stages. At the first stage, a number of conventional classifiers operate in parallel, each of which processes the input signal according to its own algorithm and produces a complete response on its classification. At the second stage, a neural network also operates, which accepts the output signals of the neural networks of the first stage as input and forms the final classification result. Such a construction of the classifier architecture opens up wide possibilities for practical implementation. Firstly, the first stage can use neural networks that are acceptable for a specific application in terms of classification quality, computation volume, training speed, and available training data volume. These networks can be trained independently of each other. Secondly, such a construction of the first stage is easier to implement on multiprocessor and multi-core computers, as well as on computing nodes in the cloud.

Despite the specific classification quality of the neural networks of the first stage, a further increase in the resulting classification quality is ensured by increasing the number of neural networks in the first stage. Naturally, the more efficient the networks used in the first stage, the higher

the quality of the resulting classification at the output of the second stage. The limitation of the number of networks in the first stage is the requirement for different architectures for processing input signals, ensuring that different networks make mistakes on different input signals. It should be noted that new processing architectures are constantly being developed, but their number is not very large at the moment. Therefore, it is also important to increase the efficiency of the neural network in the second stage. Due to the specificity of the input signals of the second stage, some complex architectures have no advantage over the multilayer perceptron (MLP), and it is usually used in practice [2]. MLP consists of neurons, which is a multi-input adder with adjustable weights for each of the inputs, the output of which is transformed by a constant activation function. In [3], it is proposed to adjust the activation functions. This provides new opportunities and flexibility in processing input signals. Therefore, the development of a neural network with trainable activation functions for the second stage of an ensemble classifier with stacking to improve the resulting classification quality and reduce the amount of computation is a relevant task.

2. Literature review and problem statement

The theoretical justification for the success of neural networks in a wide range of application areas is the universal approximation theorem [4]. The cited work strictly proves that a multilayer perceptron with one hidden layer, using an arbitrary nonlinear activation function in neurons, is a universal approximator, provided that the number of neurons in the hidden layer is large enough. However, the MLP architecture is not the best in terms of the number of neurons used. Numerous modern neural network architectures demonstrate higher efficiency compared to MLP with the same number of trainable coefficients.

In [3], it is proposed to replace the trainable weight coefficients with trainable nonlinear one-dimensional functions parameterized as a spline. This architecture is called KAN (Kolmogorov-Arnold Networks). KAN nodes use conventional adders, which are not followed by a nonlinear activation function as in MLP. This architecture, like MLP, is a universal approximator [5]. In [3], it is shown that for some problems, for example, for solving partial differential equations, such an architecture shows two orders of magnitude higher accuracy with two orders of magnitude fewer parameters. However, when constructing complex composite architectures typical of modern neural networks, the efficiency of using KAN requires additional research. In [6], KANs were used directly as surrogate models to replace the optimization function in a surrogate evolutionary algorithm. That made it possible to significantly reduce the dependence on optimization function estimates during the search process, thereby reducing optimization costs. However, the use of evolutionary algorithms for training neural networks is effective only for a narrow class of problems, where the optimization function can be described as a simple formula. Paper [7] describes an improved architecture of a convolutional neural network using KAN. New Bottleneck Convolutional Kolmogorov-Arnold and Self KAGNtention layers are introduced. It is shown that scaling convolutional networks in width allows obtaining better results compared to scaling in depth. All that made it possible to significantly improve the quality of classifying objects in images with a smaller number of training parameters. It should be noted that the architecture proposed in [7] showed high results in the problem of classifying objects in images, and the use of an ensemble classifier with stacking at the second stage requires additional research. A modification of the well-known MLP-Mixer architecture [8] using the KAN approach is aimed at solving the same problem, resulting in the KAN-Mixer architecture [9]. It shows higher results than MLP-Mixer, but lower than in [7]. In [10], a study was conducted on graph neural networks (GNNs), which are used for feature extraction by capturing dependences within graphs. MLPs and fixed activation functions were completely removed from the neural network. Replacing them with KANs showed a significant increase in classification quality. However, this was achieved only for a small number of classes and the training time of such a network increased by an order of magnitude compared to a regular GNN. A similar problem, but in the area of graph collaborative filtering (GCF), was studied in [11]. To speed up training, the Fourier transform was built into the model architecture. However, the KAN in this model is used as part of the feature transformation during message passing to the graph convolution network (GCN). This makes it difficult to determine the net contribution of the KAN to the final result, since the model also uses message dropout and node dropout strategies to improve the representation power and robustness of the model. Another area where KANs have shown both better quality of predicting time-varying signals and less computational effort is satellite traffic forecasting [12]. Trainable activation functions allow better prediction of complex signal shapes, compared to MLP. It should be noted that in [12] the pure KAN architecture was used, which does not add new architectural approaches for using KAN in other applications. In [13], the SigKAN architecture with trainable path signatures was proposed for forecasting time series using KAN. This improved the forecasting quality, but the use of path signatures in image classification requires additional research.

Thus, the use of trainable activation functions is promising in terms of providing additional capabilities of the neural network to improve the resulting quality of processing with a smaller number of trainable parameters. However, the issues of practical implementation of trainable activation functions and the efficiency of their use in various neural network architectures remain open. In the first work [3], which proposed the KAN architecture, splines were used to implement trainable activation functions. However, many subsequent studies propose replacing splines with other systems of functions that have proven efficient in various approximation problems to improve the efficiency of neural networks. Radial basis functions (RBF) have found the widest use in neural networks. In [14], universal approximation properties of neural networks with RBF are considered. In [15], it is shown that Gaussian RBFs approximate splines well. However, the issue of joint use of trainable activation functions and neurons remained unaddressed. In [16], the use of RBF as an activation function in the hidden

layers of a neural network improved the interpretability of internal processes. However, the work did not aim to train activation functions. The influence of the width and location of the RBF center on the characteristics of the neural network was studied. In [17], it was proposed to supplement a conventional neural network with RBF with a feature extractor. This increased the classification accuracy. RBFs in this architecture did not directly solve the problem of approximating activation functions. In [18], the problem of approximating functions was solved using neural networks using RBF. The architecture included a clustering algorithm that was used to determine the number of nodes in the hidden layer, as well as the centers and width of the RBF. That improved the quality of function approximation; however, this is a significantly limited solution compared to KAN. Transformation of input data using exponential functions versus processing unmodified data and then passing it to a conventional RBF neural network was investigated in [19]. However, the unmodified transformation was investigated, as opposed to the learnable function in KAN. In [20], an algorithm for training the RBF neural network architecture itself is proposed. Hidden layer nodes can be added and removed based on the nodes' contributions to the resulting characteristics. In addition, the width of the radial functions is adjusted using a special calculation method. It should be noted that such an architecture is mainly intended for research purposes due to the complexity of implementing the use of well-known high-level libraries for programming neural networks.

The study of the efficiency of replacing splines in KAN with various polynomial functions was reported in [21]. 18 different polynomials were tested for recognizing handwritten digits from the MNIST dataset. It should be noted that all polynomials allowed obtaining high results, which, however, were significantly lower than the best known results for MNIST. In addition, the efficiency of using these polynomials in the KAN architecture when processing more complex datasets remains unexplored. In [22], the advantage of using Wav-KAN when processing hyperspectral data is shown using satellite images taken in an extended spectral range. However, the efficiency of using wavelets to approximate activation functions in specific applications requires further study. In order to ensure efficient parallelization of processing on GPUs, the ReLU-KAN architecture is proposed in [23]. Splines are replaced with an approximation of activation functions using simple ReLU functions. This replacement allowed to increase the computation speed by 20 times in the studied examples of approximation of complex functions. However, this architecture has been studied only for the problem of approximation of nonlinear functions. In [7], the architecture of a convolutional neural network using KAN was modernized by introducing new layers Bottleneck Convolutional Kolmogorov-Arnold and Self KAGNtention. Comparison of various functions for approximation of activation functions in the problem of recognition of objects in images showed the advantage of Chebyshev polynomials. However, for use at the second stage of the ensemble classifier, such an architecture is redundant. Thus, the literature review showed the promise of using trainable activation functions in neural networks. However, the KAN architecture in its pure form demonstrates high results only in problems of approximation of complex functions. Processing problems such as image classification requires the introduction of additional elements into the architecture. Despite the fact that in [3] it was proposed to use splines to create trainable activation functions, the vast majority of works with KAN use Gaussian RBF. This is explained by their proximity to splines and lower computational costs. Therefore, it seems appropriate to first study the use of Gaussian RBF at the second stage of the ensemble classifier. It should be noted that both MLP and KAN are universal approximators, but their approximation mechanism is different. Therefore, it seems promising to study the effectiveness of a neural network for the second stage of the ensemble classifier with stacking, using the capabilities of MLP and KAN simultaneously.

3. The aim and objectives of the study

The aim of our work is to develop and study a neural network for the second stage of the ensemble classifier, containing a layer of trainable activation functions and MLP. This will make it possible to reduce the volume of calculations and improve the quality of classification by ensemble classifiers with stacking.

To achieve the goal, the following tasks were set:

– to design the architecture of a neural network for the second stage of the ensemble classifier with stacking, including both trainable activation functions and MLP;

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

– to compare the efficiency of using trainable activation functions in the neural network of the second stage of the classifier compared to using only MLP in it.

4. The study materials and methods

The object of research in this paper is ensemble classifiers with stacking. The subject of our research is a neural network at the second stage of the classifier. The dataset for research is CIFAR-10 [24]. This set contains color images, 32×32 pixels in size, 50,000 for training and 10,000 for testing. The images belong to c=10 classes. Examples of images are shown in Fig. 1 [24].

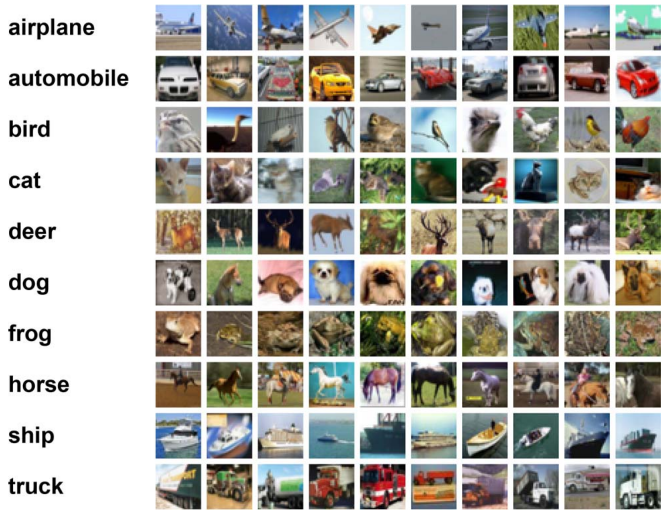

Fig. 1. Examples of images from the CIFAR-10 set

The first stage of the ensemble classifier used the same neural networks as in [2]: CCT [25], EANet [26], MLP-Mixer [8], Fnet [27], gMLP [28], and SwinTr [29]. These networks were trained for 50 epochs. The Xavier initialization [30] was used to initialize the weight coefficients. The software implementation of these networks in Python using the Keras library is given in [31]. The resulting classification quality of these networks after training for the CIFAR10 dataset and the number of trainable coefficients are given in Table 1 [2].

The neural network program with trainable activation functions is written in the Python programming language using the Keras library. Training was performed over 200 epochs using the output signals of pre-trained first-stage neural networks. Training was performed on training data. Classification quality was assessed on the testing data set and was defined as the ratio of the number of correctly recognized objects to the total number of images (10,000).

Table 1

Parameters of the first stage classifiers after training

Neural network	Quality of classification	Number of trainable weights
CCT	0.8021	408139
EANet	0.6788	355530
FNet	0.7572	582410
SwinTr	0.7128	151386
MLP-Mixer	0.7674	219658
gMLP	0.7405	862218

5. Results of investigating the efficiency of using a layer of trainable activation functions

5. 1. Construction of the neural network architecture for the second stage of the ensemble classifier with stacking

The KAN architecture proposed in paper [3] assumes that, by analogy with MLP, each input signal is transformed using q activation functions, the results are fed to q adders of the hidden layer. Since the use of MLP is proposed after the KAN layer in this paper, we shall adopt the following simplification. Each input signal of the second stage is transformed using m Gaussian RBF functions, the results are weighted by trainable weight coefficients and fed to the adder. Thus, only one trainable activation function is formed for each of the input signals of the second stage. From the outputs of the adders, the signals are fed to the output layer of the neural network, consisting of ordinary neurons with weight coefficients for the inputs, an adder and a softmax activation function.

The Gaussian RBFs used take the following form:

$$\varphi_{ji}\left(x_{jk}\right)=e^{-\left(\left(x_{jk}-x_{j\min}+i\times d\right)^2/\left(2\times s^2\right)\right)}, \quad (1)$$

where i=0, ..., (m–1) is the function number,

j=0, ..., (l–1) is the number of the first-stage neural network of the ensemble classifier,

x_{jk} is the k-th output signal of the j-th first-stage neural network,

l is the number of neural networks at the first stage of the ensemble classifier,

m is the number of RBF functions used,

$x_{j\min}$ is the minimum value of the output signal of the j-th first-stage neural network,

$d=(x_{\max}-x_{\min})/(m-1)$ is the distance between the centers of adjacent RBFs,

$x_{j\max}$ is the maximum value of the output signal of the j-th first-stage neural network,

$s=(x_{j\max}-x_{j\min})/8.45$ is the coefficient that specifies the RBF width.

Thus, for the outputs of each of the first-stage neural networks, its own set of m RBFs is used, the centers of which are uniformly distributed from $x_{j\min}$ to $x_{j\max}$ with interval d. A typical view of the RBF set for m=6 is shown in Fig. 2.

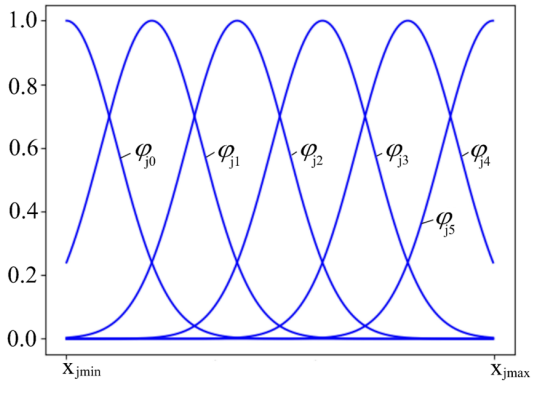

Fig. 2. Typical view of RBF set for m=6

The trainable activation function looks like this:

$$F_{jk}\left(x_{jk}\right)=\sum_{i=0}^{m-1}\varphi_{ji}\left(x_{jk}\right)\times w_{jki}, \quad (2)$$

where F_{jk} is the activation function for the k-th output signal of the j-th neural network of the first stage of the ensemble classifier,

x_{jk} is the k-th output signal of the j-th neural network of the first stage of the ensemble classifier,

w_{jki} is the i-th trainable weight coefficient for forming the activation function for the k-th output signal of the j-th neural network of the first stage of the ensemble classifier.

In accordance with formula (2), the element of the neural network architecture of the second stage of the ensemble classifier, which forms the trainable activation function, takes the form shown in Fig. 3.

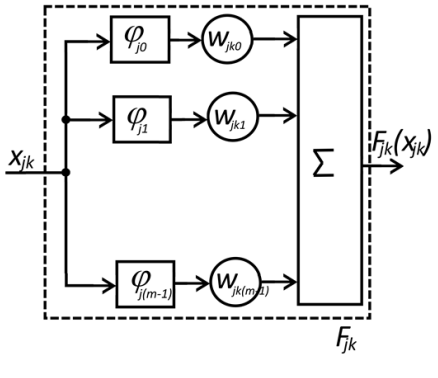

Fig. 3. Formation of a trainable activation function for x_{jk}

The architecture of the neural network of the second stage of the ensemble classifier is shown in Fig. 4.

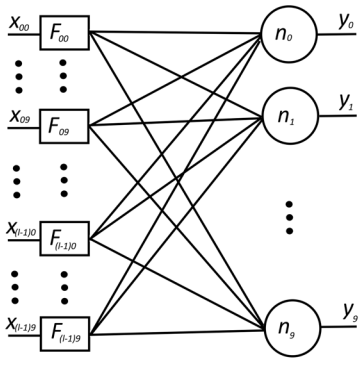

Fig. 4. Architecture of the neural network at the second stage of the ensemble classifier

The input signals are the output signals of the first neural network of the first stage x_{00}, ..., x_{09}, the second neural network x_{10}, ..., x_{19}, and so on up to the output signals of the l-th neural network of the first stage $x_{(l-1)0}$, ..., $x_{(l-1)9}$. Each of these signals is transformed by its own trainable activation function and is fed to each of the 10 neurons of the output layer n_0, ..., n_9. In each of the neurons, all their input signals are weighted by trainable weight coefficients, summed up, transformed by the softmax activation function, and fed to the outputs. The bias value, weighted by its trainable coefficient, is also fed to the summation in the neurons. The output signal of the neurons, which has the maximum value, is taken as the response of the neural network of the second stage and, accordingly, as the output signal of the entire ensemble classifier:

$$class(s) = \arg\left\{\max_p\left(y_p(s)\right)\right\}. \quad (3)$$

The architecture of the entire ensemble classifier is shown in Fig. 5. The input of the ensemble classifier is an image s, and the output is scalar values y_0, ..., y_9.

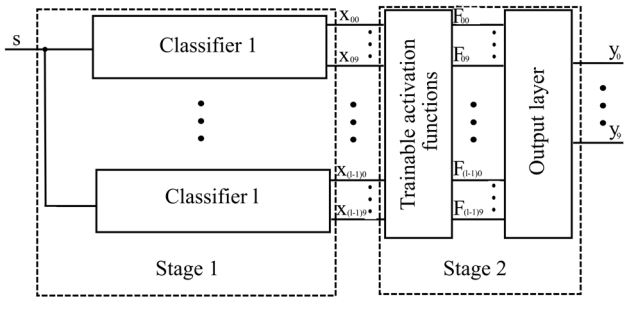

Fig. 5. General architecture of an ensemble classifier

The efficiency of using trainable activation functions in the second-stage neural network was compared with the results reported in [2].

In that work, the second-stage neural network contained 3 layers:

– input layer, dimensionality $c\times l$ (c is the number of classes in the data set, l is the number of neural networks at the first stage of the classifier);

– hidden layer, dimensionality $c\times(l-1)$, activation function Relu;

– output layer, dimensionally c, activation function softmax.

In our work, instead of the hidden layer of the second-stage neural network, a layer of trainable activation functions was used. The number of trainable coefficients in this layer is $l\times c\times m$. A comparative table for the number of trainable weight coefficients is given in Table 2.

Table 1 demonstrates that the neural network architecture developed in this subchapter for the second stage of the ensemble classifier provides a reduction in the number of trained coefficients at the second stage by 1.96–3.7 times, depending on the number of neural networks at the first stage.

Table 2

Number of adjustable weights in the second stage neural network

Number of classifiers in the first stage of the ensemble classifier l	2	3	4	5	6
Number of configurable weights in MLP [2]	630	830	1,540	2,450	3,560
Number of configurable weights in this work for $m=6$	321	481	641	801	961
Ratio of the number of weights	1.96	1.73	2.4	3.06	3.7

5. 2. Studying the dependence of classification quality on the number of Gaussian RBFs used for approximation

For the study, an ensemble classifier with three neural networks at the first stage CCT, EANet, and MLP-Mixer was used. The resulting classification quality for the entire ensemble classifier after training is given in Table 3. The best results were obtained for m=6, so this number of RBFs was used for further studies.

Table 3

Dependence of classification quality on the number of Gaussian RBFs used for approximation

Number of Gaussian RBFs used for approximation (m)	4	5	6	7	8
Quality of classification	0.8402	0.8423	0.8431	0.8430	0.8418

As an example, Fig. 6 shows the activation functions for the output signals of the CCT neural network after training the second stage of the ensemble classifier. The first stage of the classifier used the CCT, EANet, and MLP-Mixer neural networks. The number of RBFs used for approximation is m=6.

From the shape of the trained activation curves, it is clear that after training, only large positive values are fed to the output layer of the second-stage neural network, while negative and small positive values are reset to zero.

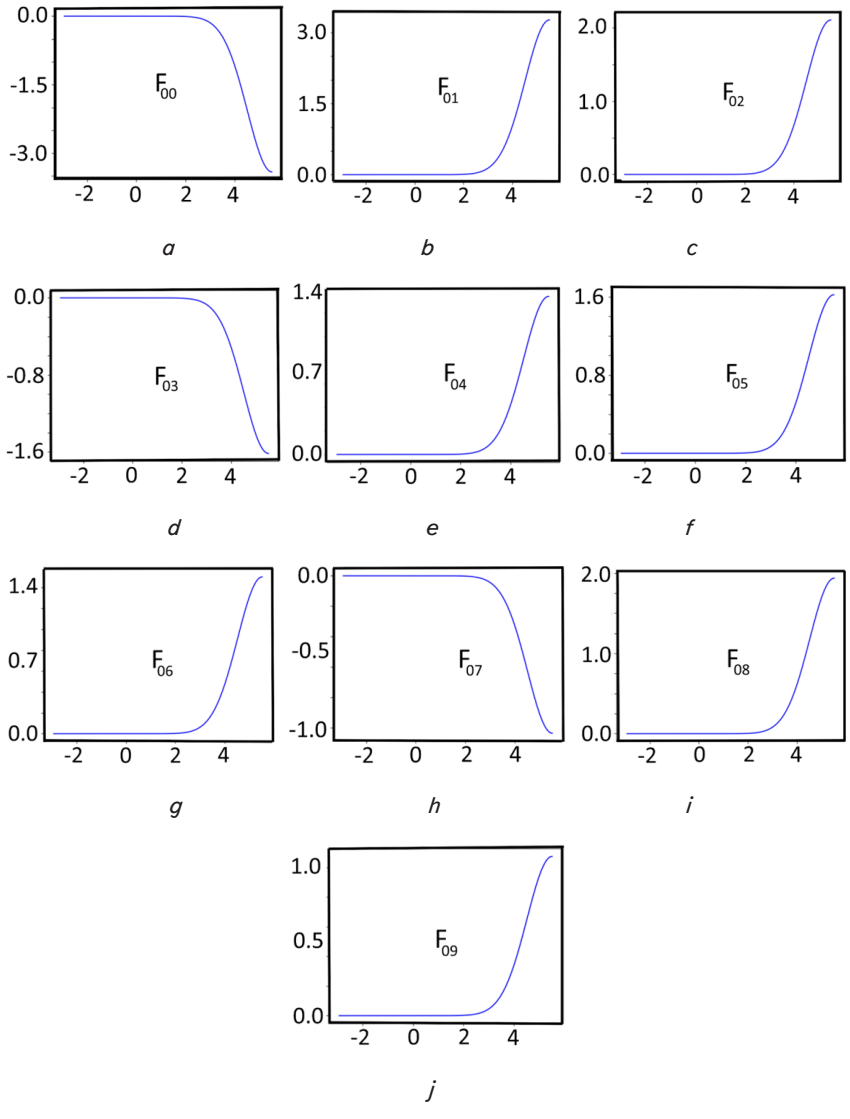

Fig. 6. Activation functions for the output signals of the CCT neural network after training the second stage of the ensemble classifier: *a* – function F_{00}; *b* – function F_{01}; *c* – function F_{02}; *d* – function F_{03}; *e* – function F_{04}; *f* – function F_{05}; *g* – function F_{06}; *h* – function F_{07}; *i* – function F_{08}; *j* – function F_{09}

5. 3. Comparing the efficiency of using trainable activation functions and pure MLP

A comparison of the efficiency of replacing the hidden layer of the MLP of the second stage of the classifier with a layer of trainable activation functions was carried out for the number of neural networks at the first stage from 3 to 6. The number of RBFs used in all cases was m=6. The results are given in Table 4. The indicators of MLP use are taken from paper [2].

The results in Table 4 show that replacing the hidden layer in MLP with a layer of trainable activation functions led to a slight decrease in classification errors from 0.45 % to 1.9 %. At the same time, in the second-stage neural network, according to Table 1, a significantly smaller number of trainable coefficients was used, differing from 1.69 to 3.7 times.

Table 4

Comparison of the efficiency of using trainable activation functions and pure MLP

Classifiers of the first stage	l	Classification quality when using MLP	Classification quality when using a layer of trained activation functions	Reducing errors when using a layer of trainable activation functions
CCT+EAT+MLP-Mixer	3	0.8401	0.8431	1.9 %
CCT+EAT+MLP-Mixer+FNet	4	0.8445	0.8452	0.45 %
CCT+EAT+MLP-Mixer+FNet+gMLP	5	0.8457	0.8467	0.65 %
CCT+EAT+MLP-Mixer+ +FNet+gMLP+SwinTr	6	0.8468	0.8477	0.59 %

6. Discussion of results of the study on using a layer of trainable activation functions in an ensemble classifier

The distinctive features of the proposed architecture of the second-stage neural network of the ensemble classifier with stacking are as follows. Replacing the hidden MLP layer used in [4] with a layer of trainable activation functions

allowed us to reduce the number of errors by 0.45 %–1.9 % for the same data set and neural networks at the first stage. At the same time, the number of trainable weight coefficients decreased by 1.69–3.7 times. Such a reduction in the number of trainable coefficients was achieved by the fact that, unlike in [3, 6, 7, 9–13, 15, 22, 23], each output signal of the first stage was processed by only one trainable activation function. In addition, the second-stage neural network contained not only trainable activation functions, but also an output layer with ordinary neurons. It should also be noted that the differences between the proposed architecture of the second-stage neural network and the architectures with trainable activation functions in the cited works made it possible to use the widely used Keras software library [31], rather than the specially developed Py Kan [3].

The positive result obtained in this work is a consequence of the simultaneous presence in the proposed architecture of the second-stage neural network of two different types of approximators that complement each other. This is a layer with trainable activation functions and an output layer with ordinary neurons.

The advantages of the proposed architecture of the second-stage neural network are in improving the quality of classification with a smaller number of trainable weight coefficients. An additional advantage is the ability to use the high-level Keras library.

The results given in Table 3 showed that for the used CIFAR-10 dataset and the study of the architecture of only the second stage of the classifier, the best results are provided by using 6 Gaussian RBFs for approximation. Obviously, for other datasets and other uses of the trainable neural network, it is necessary to conduct an additional study on the most suitable type of approximating functions and their number. The resulting activation functions shown in Fig. 6 confirmed the general advantage of using trainable activation functions – higher interpretability of the results of neural network training [3]. Fig. 6 clearly shows that the activation functions pass only significant output signals for further processing and cut off negative and small values. The results of Tables 1, 4 demonstrate that for all variants of ensemble classifiers considered in [2], the proposed architecture of the second-stage neural network provides higher classification quality with a smaller number of weight coefficients. It should be noted that positive results should be expected from the joint use of approximators of different types in other neural network architectures.

As limitations of our research, it can be noted that it was conducted for a specific data set and a specific place of the proposed architectural solutions in the general architecture of the ensemble classifier.

A disadvantage of the proposed architecture is a small increase in the quality of classification. However, overcoming this disadvantage is associated, first of all, with the development of more efficient architectures of neural networks that perform processing at the first stage of the ensemble classifier.

One of the possible directions for advancing our research is the construction of new architectures of neural networks containing trainable activation functions for the first stage of the ensemble classifier with stacking. This area is promising for further reducing the volume of calculations and improving the quality of classification for the entire ensemble classifier.

7. Conclusions

1. The architecture of the neural network for the second stage of the ensemble classifier has been designed and investigated. The architecture uses two types of approximators: trainable activation functions and regular neurons with weight coefficients, an adder, and a fixed activation function. Each output signal of the first stage of the classifier is processed by only one trainable activation function. It has been shown that such an architecture provides higher classification quality with a smaller number of trainable coefficients by 1.69–3.7 times, compared to MLP.

2. The study of the dependence of classification quality on the number of Gaussian RBFs used to approximate one activation function showed that for the CIFAR-10 dataset, the best is to use 6 RBFs with uniformly distributed centers in the range of input signal values. The study of the resulting activation functions after training showed their high interpretability, compared to MLP.

3. Comparison of the use of the proposed neural network architecture in comparison with MLP at the second stage of the ensemble classifier with the same classifiers at the first stage revealed a decrease in classification error by 0.45–1.9 %.

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 carried out 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.

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. Rokach, L. (2019). Ensemble Learning. Series in Machine Perception and Artificial Intelligence. https://doi.org/10.1142/11325
2. Galchonkov, O., Babych, M., Zasidko, A., Poberezhnyi, S. (2022). Using a neural network in the second stage of the ensemble classifier to improve the quality of classification of objects in images. Eastern-European Journal of Enterprise Technologies, 3 (9 (117)), 15–21. https://doi.org/10.15587/1729-4061.2022.258187

3. Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Solja i , M. et al. (2024). KAN: Kolmogorov-Arnold Networks. arXiv. https://doi.org/10.48550/arXiv.2404.19756
4. 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
5. 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
6. Hao, H., Zhang, X., Li, B., Zhou, A. (2024). A First Look at Kolmogorov-Arnold Networks in Surrogate-assisted Evolutionary Algorithms. arXiv. https://doi.org/10.48550/arXiv.2405.16494
7. Drokin, I. (2024). Kolmogorov-Arnold Convolutions: Design Principles and Empirical Studies. arXiv. https://doi.org/10.48550/arXiv.2407.01092
8. Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T. et al. (2021). MLP-Mixer: An all-MLP Architecture for Vision. arXiv. https://doi.org/10.48550/arXiv.2105.01601
9. Cheon, M. (2024). Demonstrating the efficacy of Kolmogorov-Arnold Networks in vision tasks. arXiv. https://doi.org/10.48550/arXiv.2406.14916
10. Zhang, F., Zhang, X. (2024). GraphKAN: Enhancing Feature Extraction with Graph Kolmogorov Arnold Networks. arXiv. https://doi.org/10.48550/arXiv.2406.13597
11. Xu, J., Chen, Z., Li, J., Yang, S., Wang, W., Hu, X. et al. (2024). FourierKAN-GCF: Fourier Kolmogorov-Arnold Network – An Effective and Efficient Feature Transformation for Graph Collaborative Filtering. arXiv. https://doi.org/10.48550/arXiv.2406.01034
12. Vaca-Rubio, C. J., Blanco, L., Pereira, R., Caus, M. (2024). Kolmogorov-Arnold Networks (KANs) for Time Series Analysis. arXiv. https://doi.org/10.48550/arXiv.2405.08790
13. Inzirillo, H., Genet, R. (2024). SigKAN: Signature-Weighted Kolmogorov-Arnold Networks for Time Series. arXiv. https://doi.org/10.48550/arXiv.2406.17890
14. 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
15. Li, Z. (2024). Kolmogorov-Arnold Networks are Radial Basis Function Networks. arXiv. https://doi.org/10.48550/arXiv.2405.06721
16. Bao, X., Liu, G., Yang, G., Wang, S. (2020). Multi-instance Multi-label Text Categorization Algorithm Based on Multi-quadric Function Radial Basis Network Model. 2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD), 1998, 133–136. https://doi.org/10.1109/icaibd49809.2020.9137478
17. Basha Kattubadi, I., Murthy Garimella, R. (2020). Novel Deep Learning Architectures: Feature Extractor and Radial Basis Function Neural Network. 2020 International Conference on Computational Performance Evaluation (ComPE), 024–027. https://doi.org/10.1109/compe49325.2020.9200146
18. 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), 9, 105–109. https://doi.org/10.1109/icinfa.2018.8812435
19. 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
20. 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), 9, 215–220. https://doi.org/10.1109/imcec.2018.8469684
21. 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
22. Seydi, S. T. (2024). Unveiling the Power of Wavelets: A Wavelet-based Kolmogorov-Arnold Network for Hyperspectral Image Classification. arXiv. https://doi.org/10.48550/arXiv.2406.07869
23. Qiu, Q., Zhu, T., Gong, H., Chen, L., Ning, H. (2024). ReLU-KAN: New Kolmogorov-Arnold Networks that Only Need Matrix Addition, Dot Multiplication, and ReLU. arXiv. https://doi.org/10.48550/arXiv.2406.02075
24. Krizhevsky, A. The CIFAR-10 dataset. Available at: https://www.cs.toronto.edu/~kriz/cifar.html
25. Hassani, A., Walton, S., Shah, N., Abuduweili, A., Li, J., Shi, H. (2021). Escaping the Big Data Paradigm with Compact Transformers. arXiv. https://doi.org/10.48550/arXiv.2104.05704
26. Guo, M.-H., Liu, Z.-N., Mu, T.-J., Hu, S.-M. (2022). Beyond Self-Attention: External Attention Using Two Linear Layers for Visual Tasks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1–13. https://doi.org/10.1109/tpami.2022.3211006
27. Lee-Thorp, J., Ainslie, J., Eckstein, I., Ontanon, S. (2022). FNet: Mixing Tokens with Fourier Transforms. Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. https://doi.org/10.18653/v1/2022.naacl-main.319
28. Liu, H., Dai, Z., So, D. R., Le, Q. V. (2021). Pay Attention to MLPs. arXiv. https://doi.org/10.48550/arXiv.2105.08050
29. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z. et al. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.1109/iccv48922.2021.00986
30. Brownlee, J. (2021). Weight Initialization for Deep Learning Neural Networks. Available at: https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/
31. Code examples. Computer vision. Keras. Available at: https://keras.io/examples/vision/