Kolmogorov-Arnold Networks with Trainable Activation Functions for Data Regression and Classification

Kuan-Lin Chen
Graduate Institute of Communication Engineering,
National Taiwan University
Taipei 10617, Taiwan
r13942067@ntu.edu.tw

Jian-Jiun Ding
Graduate Institute of Communication Engineering,
National Taiwan University
Taipei 10617, Taiwan
jjding@ntu.edu.tw

***Abstract*—Deep learning has become a widely utilized approach for a variety of applications. Some networks, such as Multilayer Perceptrons (MLPs), apply input feature data to compute regression and classification results. In recent years, many architectures, such as transformers, incorporate MLPs for backend computations and achieve notable success across a range of applications. In 2024, a novel network architecture, named the Kolmogorov-Arnold Network (KAN), has been introduced. Like MLPs, KANs share similarities in structure, but are grounded in the Kolmogorov-Arnold representation theorem. Unlike MLPs, which apply fixed activation functions at each neuron, KANs use trainable activation functions directly, making the model require fewer network layers and neurons for training. In addition, we conducted a comparative analysis using other models such as the radial basis function (RBF) network. This analysis aims to the training performance and compare the efficiency of KANs and MLPs on commonly used machine learning datasets. The results show that KANs outperform MLPs in regression and classification tasks.**

Keywords—Neural network, multilayer perceptron, Kolmogorov-Arnold network (KAN), radial basis function network

I. Introduction

Deep learning techniques are now applied to a variety of tasks, and basic tasks such as classification and regression problems can typically be addressed by constructing neural network frameworks to model data and make appropriate prediction. In tasks involving different features as inputs, regression problems that output a value and classification problems that output a category vector, often utilize machine learning algorithms such as random forest for classification or regression, and support vector machines (SVM) [1] etc. for classification or regression. In neural networks constructed through deep learning, MLPs are commonly used to fit the input features to the output results [2], [3], and can be applied to both classification and regression tasks. Moreover, radial basis function (RBF) Networks [4] can achieve similar results. When dealing with image-based data, while networks like MLPs can be used to perform tasks, they do not outperform convolution neural networks (CNNs), which are designed for tasks involving image inputs.

In recent years, neural network frameworks have been developed based on different mathematical principle, modeling technique, and computational foundation. In deep learning, parameters can be set to a trainable state, allowing gradient calculation through error estimation and updating the model accordingly. The latest network architecture, Kolmogorov-Arnold Networks (KANs) [5], constructs its model theory based on mathematical theorems, and models the network structure using B-spline functions, among other methods.

In 2024, Liu *et al.* proposed KANs [5] and found that they performed exceptionally well in multi-input and continuous polynomial function tasks and outperform traditional models in solving Partial Differential Equations (PDEs). When training on new data, MLPs often suffer from catastrophic forgetting, where they lose previously learned knowledge. In contrast, KANs do not exhibit such forgetting after training. The results from various experiments indicate that KAN networks hold significant potential. Therefore, this study aims to apply KANs to basic deep learning datasets and compare their performance with MLP networks. Additionally, since KANs have more hyperparameters to configure, this research also seeks to conduct verify the impact of these parameter settings on KAN network training via ablation studies.

In regression tasks, commonly used algorithms e.g. the SVM, the MLP, the RBF, simple Linear Regression (LR) [6], and Random Forest Regression. According to [7], MLP has been employed to predict the efficiency of water desalination, using the Adam optimizer to fine-tune the neural network parameters. Additionally, some studies have used algorithm like the RBF and the MLP to recognize complex relationships between haloketones and water quality [8], predicting outcomes based on specific input features such as pH, temperature, and the concentration of certain chemical ions.

In classification tasks, common algorithms include Decision Tree Classification [9] and *k*-Nearest Neighbors (*k*NN) [10]. However, deep neural networks like MLP remain the preferred approach for classification and prediction. For example, MLP has been applied to detect and classify DDoS attacks [11], where an the Auto-Encoder [12] is used for feature extraction, and the extracted features are feed into the MLP to enable effective automatic feature learning. Another common application of the MLP is in disease diagnosis [13].

II. THE PROPOSED METHOD

A. The Training Data

In this study, we used several machine learning datasets to conduct preliminary evaluations and validate the initial performance of the models. Specifically, we used 4 classification datasets and 4 regression datasets to compare the models' performance. The 4 classification datasets include the IRIS dataset, the MNIST dataset, the Fashion-MNIST dataset, and the CIFAR-10 dataset. The regression datasets consist of the Boston Housing dataset, the California Housing dataset, the Concrete dataset, and the ENB2012 dataset.

B. Models' Detail

The RBF Network is a specialized artificial neural network designed for tasks such as regression, classification, and function approximation. It comprises three layers: an input layer, an RBF layer, and an output layer. The RBF layer typically uses a Gaussian function (1) as its activation function, with trainable parameters β and c. This layer calculates the Euclidean distance between the input features and c, multiplies the result by $-\beta$, and applies a logarithm to produce its output. The output layer usually consists of a fully connected layer.

$$rbf(\|input - c\|) = exp(-\beta \times \|input - c\|). \quad (1)$$

The concept of the KAN is similar to that of the MLP, but in KAN networks, the linear output layers are removed, and instead, the nonlinear outputs are treated as trainable parameters. This approach allows the activation functions to be learnable and updated during training. Because KAN is based on the Kolmogorov-Arnold theorem (KA theorem). This theorem states that for any continuous function f of multiple variables, it can be represented as a finite sum of nested functions of single variables. In other words, any continuous function of multiple variables can be expressed as a finite sum of single-variable functions through addition. Specifically, f can be represented as shown in equation (2). This indicates that f takes an n dimensional input and projects it onto the real domain $\mathbb{R}$, where the values of all dimensions are within the range [0,1]. Equation (3) represents the inner functions, which have a domain of [0,1] and a range of real numbers $\mathbb{R}$. Equation (4) represents the outer function, which has both its domain and range in the real numbers $\mathbb{R}$. Thus, the Kolmogorov-Arnold theorem can be expressed through two layers of nested functions as shown in (5). Here, $f(x_1, x_2, \dots, x_n)$ is a multivariable function with n inputs. Each outer function takes as input the sum of n inner functions, and the result of $2n+1$ outer functions is summed to express a multivariable continuous function.

$$f: [0,1]^{\mathrm{n}} \to \mathbb{R}, \quad (2)$$

$$\phi_{\mathrm{q,p}}: [0,1] \to \mathbb{R}, \quad (3)$$

$$\Phi_{\mathrm{q}}: \mathbb{R} \to \mathbb{R}, \quad (4)$$

$$f(x) = f(x_1, x_2, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \phi_{q,p}(x_p)\right). \quad (5)$$

The KAN can offer a more effective fitting for most real-world applications, as these applications are often nonlinear and involve multiple inputs. Specifically, the KAN layer involves several calculations, including residual function and B-spline calculations. In this study, the residual function used is the SiLU function (5), which is also the default function employed by the model. The input x, after being processed through the residual function, undergoes interpolation calculations using the B-spline function. The output of the B-spline function is then weighted by the output of the residual function. Finally, a *mask* is used to control the final output, and the results are aggregated to produce the final output vector.

$$SiLU(x) = \frac{x}{1+e^{-x}}. \quad (5)$$

In the model design, we have structured the KAN network into two configurations: a single KAN layer (KAN-1) and a two KAN layers (KAN-2). For the MLP, we have configurations with 1 layer, 2 layers, and 4 layers of fully connected layers, denoted as MLP-1, MLP-2, and MLP-4, respectively.

For regression datasets, we investigated the model's training performance under conditions with standardization. Standardization was performed using the Min-Max Scaler method for each feature. This method projects the original features into the [0,1] range, aligning with the assumptions of the KA theorem.

After preprocessing the data, it is trained using six different network models. In the study, the Adam optimizer is employed for parameter updates, the β_1 and β_2 are 0.9 and 0.999, respectively. The learning rate in classification task is 0.0003; in regression task is 0.001; training epochs in classification is 15; in regression is 500; batch size is 64.

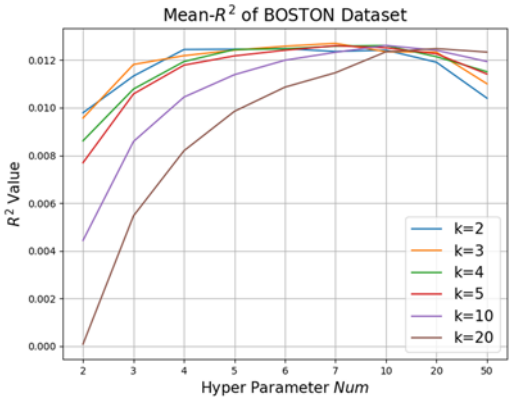
C. Evaluating model

In this paper, each of the six network models is trained 20 times on each dataset. After each training, we will evaluate model via testing dataset and calculate some metrics, like the average values of the Mean Squared Error (MSE) and R^2 score (for regression tasks) or accuracy (for classification tasks).

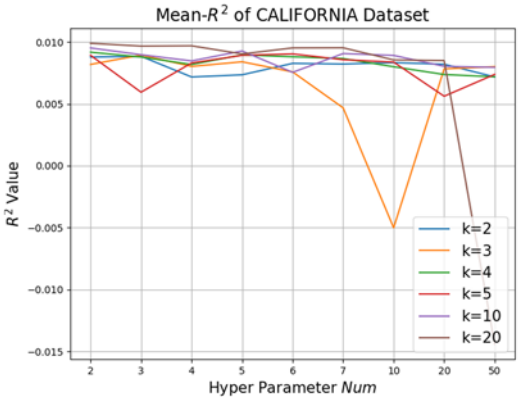
To verify whether there are significant performance differences between the models, we used the Kruskal-Wallis H test to calculate the p-value. Additionally, we performed the Tukey Honestly Significant Difference (HSD) test to compare pairs of models and determine if there are significant differences between them. The Kruskal-Wallis H test is used to compare the median differences between two or more groups of data, and it is particularly suitable when the data don't follow a normal distribution or when variances are not equal. The equation of Kruskal-Wallis H is shown in (6), where N represents the total number of evaluation values (accuracy or MSE) across all training results. R_i is the sum of ranks for the i^{th} model, and n_i is the number of evaluation values for the i^{th} model. Since each model is trained and evaluated 20 times, so n_i equals 20. The Kruskal-Wallis H test begins by ranking all evaluation values from all models in ascending order, starting from 1. The sum of the ranks for evaluation values within each model is denoted as R_i. This sum is then used to compute the test statistic. The Tukey HSD test is a post-hoc multiple comparison test often used following analysis of variance (ANOVA). It compares the performance of each pair of models and determines whether the differences between their performances are statistically significant. The equation of Tukey HSD as shown in (7), $q(\alpha, m, N-m)$ represents a distribution based on the Studentized range statistic, where α is the significance level, set to 0.05 in this study. The term $N-m$ is used in the ANOVA context to calculate the degrees of freedom for the variability between two models. The critical value can be obtained from the distribution tables. MS_w represents the within-group mean square, calculated as the sum of squares of all evaluation values divided by the degrees of freedom.

$$H = \frac{12}{N(N+1)} \sum_{i=1}^{6} \frac{R_i^2}{n_i} - 3(N+1), \quad (6)$$

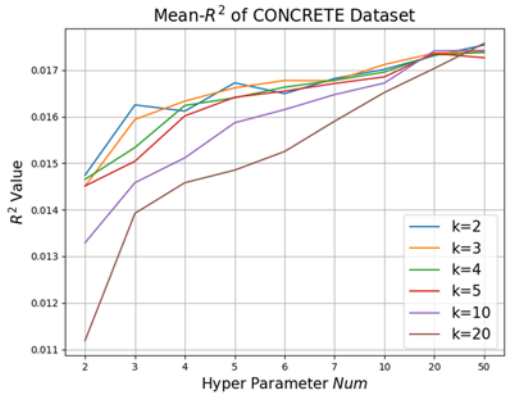
$$HSD = q(\alpha, \mathrm{m}, \mathrm{N}-\mathrm{m}) \sqrt{\frac{\mathrm{MS}_w}{n}}. \quad (7)$$



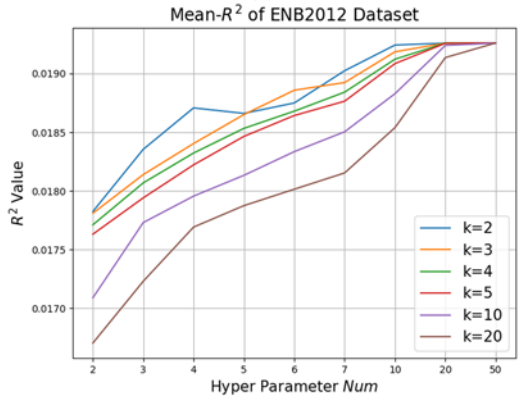
(a)



(b)

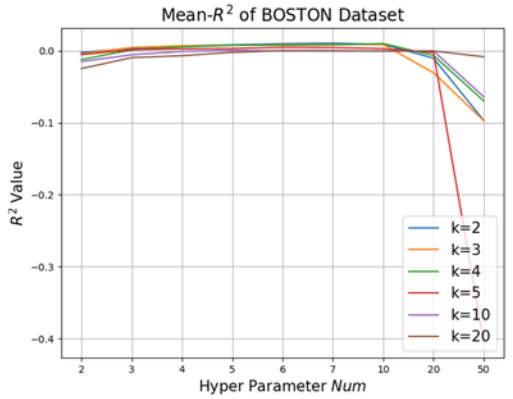


(c)

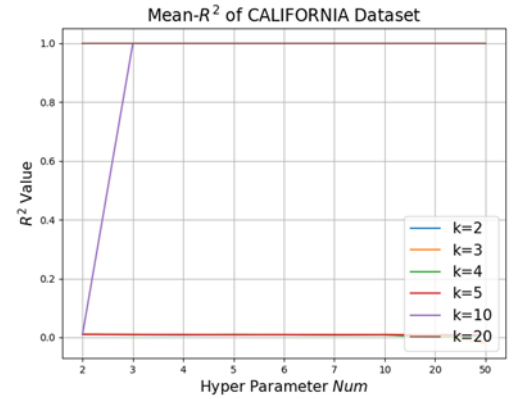


(d)

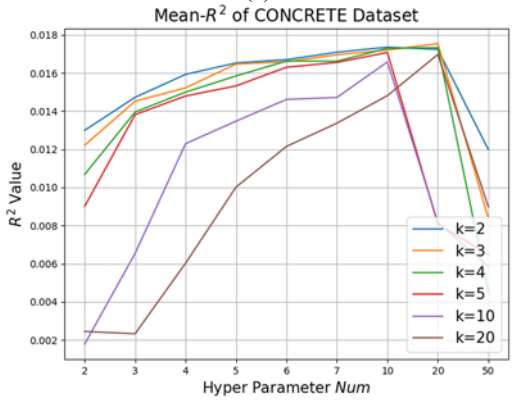
Figure 1. Grid searching result in classification dataset of KAN-1.



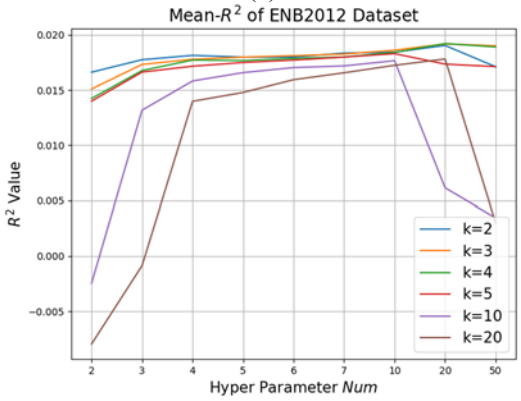
(a)



(b)



(c)



(d)

Figure 2. Grid searching result in classification dataset of KAN-2.

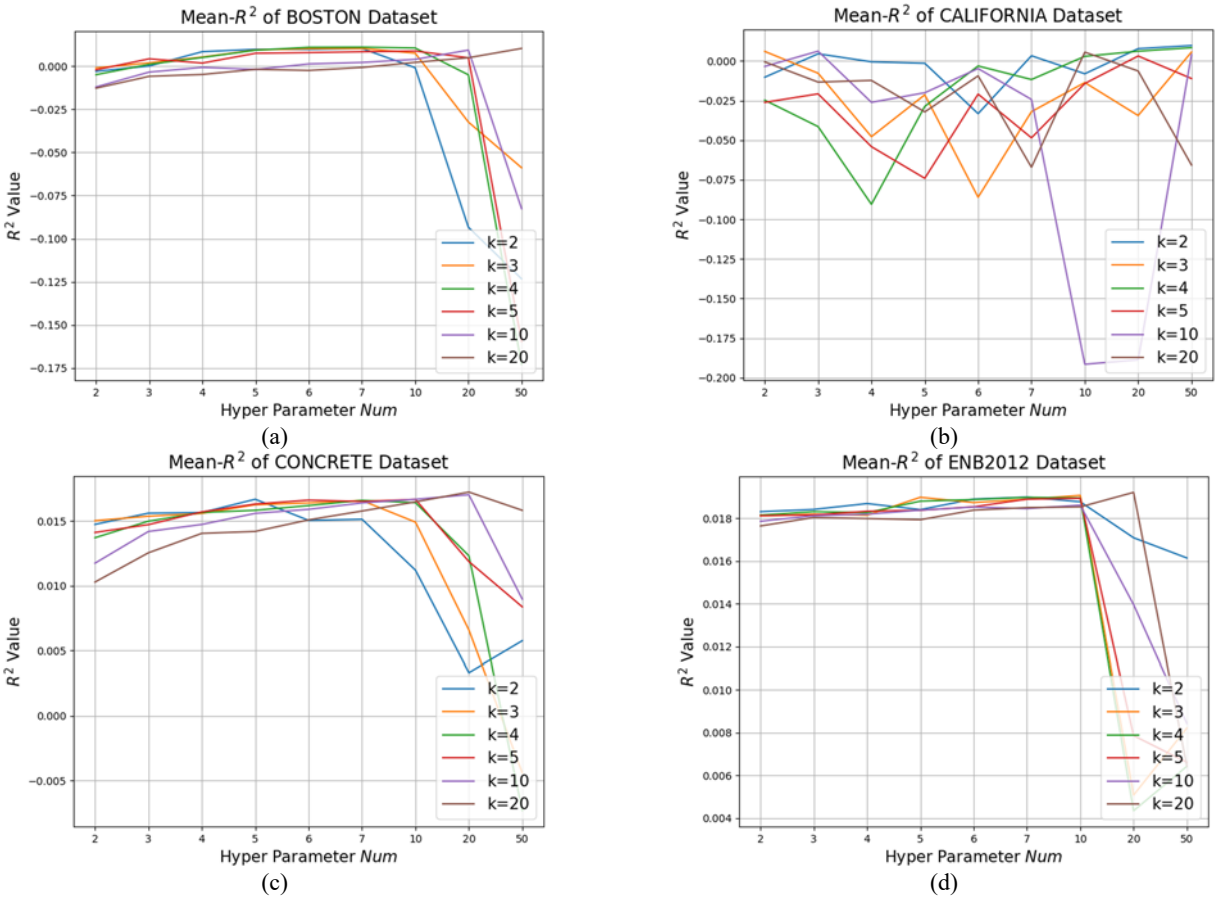


Figure 3. Grid searching result in regression dataset of KAN-2bn.

D. Ablation Experiment

In this study, we also investigated the training outcomes of KAN-1 and KAN-2 under different parameter combinations. We employed a simpler grid search to validate the results of training KANs with various parameter settings. Additionally, we tried the impact of adding a Batch Normalization (BN) layer between the two KAN layers in KAN-2, resulting in a new configuration called KAN-2bn. We aimed to determine whether KAN-2bn improves the stability of model training under different parameter combinations. The experimental parameters range of *num* is [2, 3, 4, 5, 6, 7, 10, 20, 50], and range of k is [2, 3, 4, 5, 10, 20].

We analysis the effects of these parameters on KAN network training using ANOVA. In this study, we treated the hyperparameters as a factor and employed ANOVA to assess the significance of different *num* and k hyperparameter settings on regression results. In ANOVA, we compute the F-value to determine the impact of these hyperparameters on the results, specifically r_i represents the actual R^2 values where the R^2 value is defined as:

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{9}$$

y_i, $\bar{y}$, f_i, are the ground truth, the mean of the ground truth, and the predicted result, respectively; $\bar{r}_j$ represents the predicted R^2 values from the ordinary least squares model [14]; $\bar{r}$ is the average of these R^2 values; N is the total number of data points; and k_h denotes the number of levels for these hyperparameter factors. Finally, the F-value (10) is obtained by dividing these two results. The F-value indicates the influence of the hyperparameter factor on the variation of the results. A larger F-value signifies a greater impact of the factor on the outcome.

$$ANOVA_F = \frac{\sum_{j=1}^{k_h} n_j (\bar{r}_j - \bar{r})^2}{k_h - 1} \frac{N - k_h}{SS_r}. \tag{10}$$

We calculate the p-value to assess the significance of hyperparameter factors on results. A p-value smaller than 0.01 indicates a significant impact. Using the F-distribution, the p-value is derived by computing the cumulative distribution function (CDF) of the F-statistic (11).

$$p = 1 - P(F \leq ANOVA_F). \tag{11}$$

III. EXPEROMENTAL RESULTS AND DISCUSSION

In this paper, we repeated training of KAN, MLP, MLR, and RBF networks on classification and regression tasks across various datasets. We compared the training results comprehensively. Additionally, we performed an ablation study on different KAN networks and examined the impact of hyperparameters in the KAN layer.

A. 3.1 Model Performance Comparison

The experiments show that KAN-1 excels in classification tasks when sufficient data is available. On the IRIS dataset, however, the limited data leads to less stable training for shallower MLPs like KAN-1. In such cases, deeper models like MLP-4 outperform KAN-1 by better capturing data relationships. RBF networks underperform due to their inability to handle spatially correlated data effectively.

TABLE I. THE RESULTS OF CLASSIFICATION TASKS

Dataset Name	Model Name	Acc Mean
IRIS	KAN-1	0.366667
	KAN-2	0.366667
	MLP-1	0.335833
	MLP-2	0.334583
	RBF	0.333333
MNIST	KAN-1	0.935085
	KAN-2	0.89604
	MLP-1	0.92027
	MLP-2	0.9213
	RBF	0.1135
FASHION MNIST	KAN-1	0.860905
	KAN-2	0.84289
	MLP-1	0.841635
	MLP-2	0.844065
	RBF	0.1
CIFAR 10	KAN-1	0.40345
	KAN-2	0.40867
	MLP-1	0.39942
	MLP-2	0.406685
	RBF	0.1

TABLE II. THE RESULTS OF REGRESSION TASKS

Dataset Name	Model Name	R^2 Mean
BOSTON	KAN-1	0.012455
	KAN-2	0.006805
	MLP-1	0.01137
	MLP-2	0.011695
	RBF	-0.98845
CALIFORNIA	KAN-1	0.00746
	KAN-2	0.009005
	MLP-1	0.006865
	MLP-2	0.00742
	RBF	-0.99644
CONCRETE	KAN-1	0.0166
	KAN-2	0.01649
	MLP-1	0.012465
	MLP-2	0.01247
	RBF	-1.04160
ENB2012	KAN-1	0.01867
	KAN-2	0.01789
	MLP-1	0.01765
	MLP-2	0.01779
	RBF	-0.98931

TABLE II presents the results of the regression tasks, demonstrating that KAN outperforms other network in this domain. However, a notable drawback is that KAN requires longer training time and a larger number of trainable parameters compared to other networks with the same number of units. It is also essential to standardize the input data to the range [0, 1] to ensure proper training of the KAN.

TABLE III presents the statistical results of the Tukey HSD test, which compares the performance of each pair of models. From the table, it is evident that the performance gap between KAN and RBF is considerable. In certain tasks, KAN also exhibits significant performance differences compared to MLP. However, when comparing similar architectures, the performance gap between KAN and MLP is not statistically significant. This gap can be further minimized through fine-tuning the parameters of the KAN network to improve training outcomes. Additionally, preliminary ablation experiments were conducted to investigate the impact of KAN layer hyperparameter settings on network performance.

According to the Kolmogorov-Arnold theorem, KAN networks can achieve impressive results even with fewer layers. Future work could focus on developing deeper KAN architectures or integrating them with CNNs to benchmark against traditional CNNs.

TABLE III. THE RESULT OF REGRESSION TASKS

Task type		Classification	Regression
Model Name		*p*-value	*p*-value
KAN-1	KAN-2	0.9983	0.9765
KAN-1	MLP-1	0.9678	0.9959
KAN-1	MLP-2	0.9932	0.9999
KAN-1	MLP-4	0.0003	0.9999
KAN-1	RBF	0.0	0.0027
KAN-2	MLP-1	0.9969	0.8716
KAN-2	MLP-2	0.9999	0.9503
KAN-2	MLP-4	0.0001	0.9915
KAN-2	RBF	0.0	0.0004
MLP-1	MLP-2	0.9995	0.9993
MLP-1	RBF	0.0	0.0083
MLP-1	MLP-4	0.0	0.9861
MLP-2	MLP-4	0.0001	0.9987
MLP-2	RBF	0.0	0.0041
MLP-4	RBF	0.0	0.0017
Kruskal-Wallis H test		196.42	96.524

TABLE IV. THE RESULT OF REGRESSION TASKS

Parameter name	F-value	P-value
KAN-1 *num*	0.441362	0.894705
KAN-1 *k*	0.377792	0.863429
KAN-1 *num* & *k*	0.113273	1.0
KAN-2 *num*	0.105907	0.998985
KAN-2 *k*	**6.679846**	**0.000011**
KAN-2 *num* & *k*	0.068042	1.0
KAN-2bn *num*	1.549844	0.143952
KAN-2bn *k*	0.323008	0.898593
KAN-2bn *num* & *k*	0.405667	0.999372

B. *Ablation Experiment Results of KANs*

This study includes ablation experiments on KANs, using grid search to evaluate training outcomes under various parameter combinations. The results indicate that a lower *k* value in KAN-1 leads to higher R^2, while increasing *num* further improves R^2, up to a point where excessive *num* reduces performance, the result as shown in Figure 1. However, the impact of these hyperparameters is not significant, especially for larger datasets like California, where Figure 2 shown that tuning relationships become unclear. Similar trends are observed for KAN-2. Figure 3 shows that KAN-2bn get more drastic variations. Overall, the influence of *num* and *k* on training is minimal. Future research could focus on tuning other hyperparameters, such as learning rate, architecture, and epochs, alongside optimization algorithms for improved results.

TABLE IV further analyzes *num* and *k*, showing their varying effects across architectures. In KAN-1, both parameters have high F-values but *p*-values above 0.001, indicating limited significance. For KAN-2, *k* exhibits a larger F-value and a p-value below 0.001, suggesting its adjustment may have a more pronounced effect.

IV. CONCLUSION

This study adopts KANs, which was proposed in 2024, in classification and regression. We compared the KAN to MLP and RBF networks in classification and regression tasks and aimed to identify significant differences. The results show that KANs can outperform MLPs. Ablation studies and ANOVA analyses reveal that the impact of hyperparameters varies across architectures, datasets, and tasks, highlighting the need for careful tuning in specific contexts. The underlying KA-theorem suggests potential applications of KAN principles to

advanced models like CNNs and RNNs, warranting further exploration.

REFERENCES

[1] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intell. Syst. their Appl.*, vol. 13, no. 4, pp. 18–28, Jul.1998, doi: 10.1109/5254.708428.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control. Signals, Syst.*, vol. 2, no. 4, pp. 303–314, Dec.1989, doi: 10.1007/BF02551274.

[3] M. C. Popescu, V. E. Balas, L. Perescu-Popescu, and N.Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Trans. Circuits Syst.*, vol. 8, no. 7, pp. 579–588, 2009.

[4] M. J. L. Orr, "Introduction to radial basis function networks." Technical Report, center for cognitive science, University of Edinburgh …, 1996.

[5] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljaciˇ c´, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold networks," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.19756.

[6] A. K. Kuchibhotla, L. D. Brown, A. Buja, and J. Cai, "All of linear regression," Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.06386

[7] H. Salem, A. E. Kabeel, E. M. S. El-Said, and O. M. Elzeki, "Predictive modelling for solar power-driven hybrid desalination system using artificial neural network regression with Adam optimization," *Desalination*, vol. 522, article 115411, Jan. 2022, doi: 10.1016/j.desal.2021.115411.

[8] S. Palabıyık and T. Akkan, "Evaluation of water quality based on artificial intelligence: performance of multilayer perceptron neural networks and multiple linear regression versus water quality indexes," *Environ. Dev. Sustain.*, Jun. 2024, doi: 10.1007/s10668-024-05075-6.

[9] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *J. Appl. Sci. Technol. Trends*, vol. 2, no. 01, pp. 20–28, Mar. 2021, doi: 10.38094/jastt20165.

[10] P. Cunningham and S. J.Delany, "*K*-nearest neighbour classifiers - A tutorial," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–25, Jul. 2022, doi: 10.1145/3459665.

[11] Y. Wei, J. Jang-Jaccard, F. Sabrina, A. Singh, W. Xu, and S. Camtepe, "AE-MLP: A Hybrid deep learning approach for ddos detection and classification," *IEEE Access*, vol. 9, pp. 146810–146821, 2021, doi: 10.1109/ACCESS.2021.3123791.

[12] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," Mar. 2020, [Online]. Available: http://arxiv.org/abs/2003.05991

[13] S. Mishra, H. K. Tripathy, P. K. Mallick, A. K. Bhoi, and P. Barsocchi, "EAGA-MLP—An enhanced and adaptive hybrid classification model for diabetes diagnosis," *Sensors*, vol. 20, no. 14, p. 4036, Jul. 2020, doi: 10.3390/s20144036.

[14] C. Dismuke and R. Lindrooth, "Ordinary least squares," *Methods Des. Outcomes Res.*, vol. 93, no. 1, pp. 93–104, 2006.