A Comprehensive Review on Kolmogorov-Arnold Networks through Implementation on Various Dataset

Seyed Mohammad Hoseyni Iman Gandomi Mahdi Aliyari-Shoorehdeli
Mohammadhosini60@gmail.com Iman2001.gandomi@gmail.com aliyari@eetd.kntu.ac.ir

***Abstract*—This paper presents a comprehensive review and implementation of the Kolmogorov-Arnold Network (KAN) on various datasets. The Kolmogorov-Arnold Representation Theorem provides the theoretical foundation for KAN, enabling the decomposition of complex multivariate continuous functions into simpler univariate functions. We explore the architecture of KAN, emphasizing its use of B-splines for stable and bounded polynomial approximations. We implement and test KAN on the MNIST, CIFAR-10, Intracranial Hemorrhage, and ARAS-CaSe datasets, comparing its performance against traditional Multi-Layer Perceptrons (MLPs) and convolutional neural networks (CNNs). Our experiments demonstrate significant improvements in accuracy and convergence speed for KAN models, particularly in image classification and medical image segmentation tasks. We also investigate the integration of KAN layers into CNN architectures, highlighting the potential of KAN to enhance the performance of deep learning models in various applications. Our results underscore the robustness and efficiency of KAN, paving the way for its broader adoption in the machine-learning community.**

I. Introduction

A. Kolmogorov-Arnold Network Architecture

The Kolmogorov-Arnold Representation Theorem, established by mathematicians Andrey Kolmogorov and Vladimir Arnold, states that any multivariate continuous function can be decomposed into a finite sum of continuous univariate functions and addition operations. This theorem forms the mathematical foundation of KAN, suggesting that complex functions can be broken down into simpler, more manageable components.

$$f(x) = f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right) \tag{1}$$

$$y = F(X_1, X_2, X_3, \ldots, X_n) \tag{2}$$

1. Pass input features through uni-variate functions

$$\varphi_1(X_1) \quad \varphi_2(X_2) \quad \varphi_3(X_3) \quad \ldots \quad \varphi_n(X_n) \tag{3}$$

2. Sum these up

$$\varphi_1(X_1)+\varphi_2(X_2)+\varphi_3(X_3)+\ldots+\varphi_n(X_n) = \sum_{q=1}^{n} \varphi_q(X_q) \tag{4}$$

3. Pass the sum through another function

$$f \left(\sum_{q=1}^{n} \varphi_q(X_q) \right) \tag{5}$$

We can represent the inner functions as a matrix of φ functions with dimensions $n \times m$, populated with different activation functions. Additionally, there is an input vector containing n features. The input vector will pass through these activation functions. Each of these functions is a simple polynomial curve dependent on the input x.

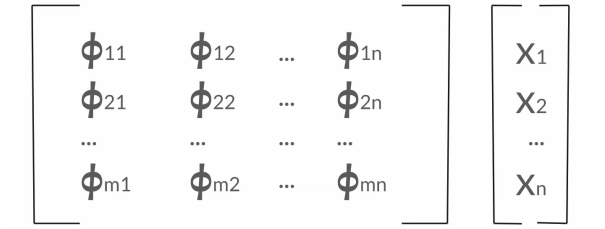


Fig. 1: A sample image.

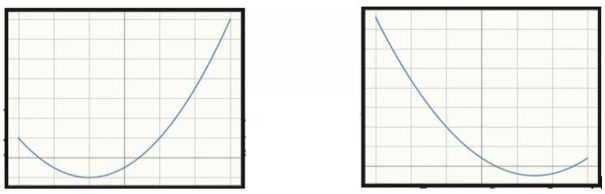


Fig. 2: A sample image.

When the values of the variables a, b, and c change, the shape of the polynomial diagram and the activation value will be affected. These are learnable parameters that determine the form of the activation function. We will update these values through backpropagation to optimize the network's predictions.

After performing a column-wise sum of the matrix, we apply a series of univariate operations. The process is as follows: given two input features and an output layer of size five, each output is passed through five different parameterized univariate activation functions. We then sum the corresponding activations to obtain the features

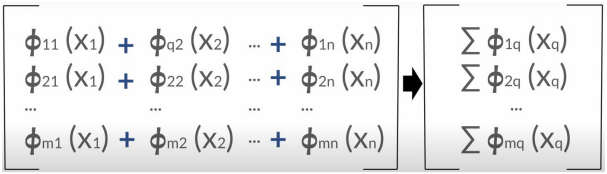


Fig. 3: A sample image.

of each output node. This entire operation constitutes one KAN layer with an input size of 2 and an output size of 5.

Similar to Multi-Layer Perceptrons (MLPs), we can stack multiple KAN layers to create a deeper neural network. The output of one layer serves as the input for the next layer.

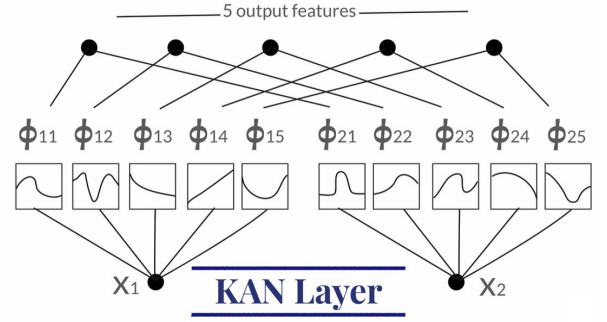


Fig. 4: A sample image.

The parameters of each activation spline can be trained using backpropagation and gradient descent. While fixed activation functions are employed at the nodes in MLPs, learnable activation functions are utilized along the edges in KANs and are summed up at the nodes.

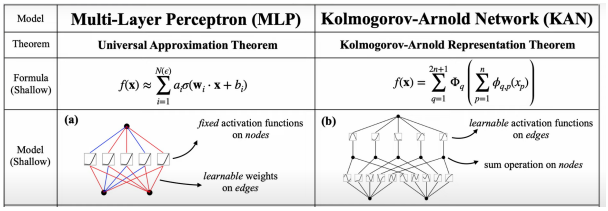


Fig. 5: A sample image.

In MLPs, the learnable parameters, which are weights and biases, are linear. In contrast, KANs do not use linear weight matrices. Instead, each weight parameter is replaced by a learnable nonlinear activation function called phi.

Higher-order polynomials tend to escalate rapidly with small changes in x, making them practically difficult and unstable to train in neural networks. Therefore, B-splines, which are more stable and bounded, are used instead.

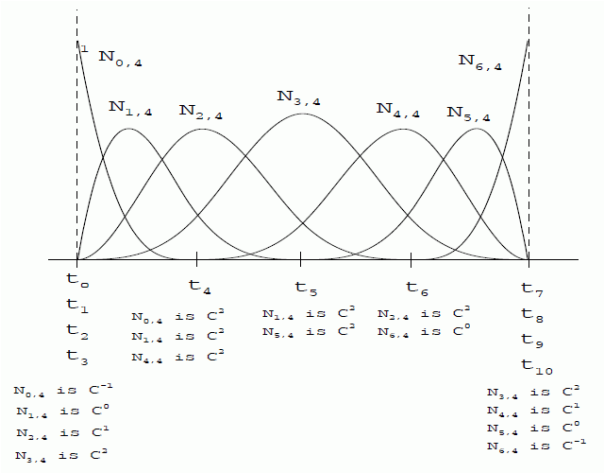


Fig. 6: Basis Function in B-Splines

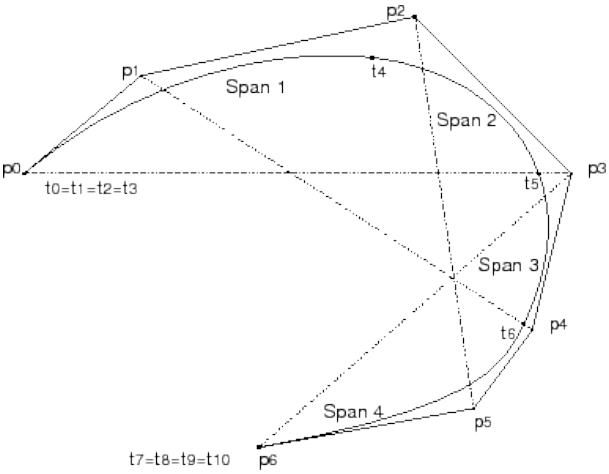


Fig. 7: B-Spline Example

B. B-Splines

B-splines, or Basis splines, are a family of piecewise-defined polynomials used for smooth curve representation and approximation. They are defined over a partition of the domain called a knot vector $T = \{t_0, t_1, \ldots, t_{n+p+1}\}$, where t_i are the knots and p is the degree of the spline. A B-spline of degree p is defined by a set of basis functions $N_{i,p}(x)$, which are non-negative and have local support. The B-spline curve is given by:

$$C(x) = \sum_{i=0}^{n} c_i N_{i,p}(x),$$

where c_i are the control points and $N_{i,p}(x)$ are the B-spline basis functions of degree p. Figure 6 illustrate the basis functions that are used in the KAN model to approximate the non-linear function in the edges.

The degree vector P often denotes the degree of the spline and helps in constructing the basis functions. The Figure 7 indicates a sample of B-splines in approximating a curve between some data points. [1], [2]

II. Datasets

A. MNIST Dataset

The MNIST [3] dataset is a collection of handwritten digits that serves as a benchmark for image classification algorithms. It was created by Yann LeCunet al. and has become a standard for testing new algorithms in the field of machine learning. The dataset consists of 70,000 grayscale images of handwritten digits from 0 to 9, split into a training set of 60,000 images and a test set of 10,000 images. Each image is 28x28 pixels in size, making it a relatively simple and small dataset that is easy to work with. The images are in grayscale, and each pixel value ranges from 0 to 255, where 255 represents a white pixel and 0 represents a black pixel. Figure 8 shows some sample images from the MNIST dataset.

Fig. 8: Sample images from the MNIST dataset.

B. CIFAR-10 Dataset

The CIFAR-10 [4] dataset, created by Alex Krizhevsky, is another widely used benchmark for image classification tasks. Unlike MNIST, which consists of grayscale images of handwritten digits, CIFAR-10 comprises colored images in 10 different classes. The dataset contains 60,000 32x32 color images, with each class having 6,000 images. The classes represent objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. Figure 9 are some sample images from the CIFAR-10 dataset.

C. Intracranial Hemorrhage Dataset

The dataset titled "Computed Tomography Images for ICH Detection and Segmentation" [5], which is publicly available on PhysioNet [6], provides CT scan radiography images related to Intracranial Hemorrhage (ICH) situation and the corresponding label for classification and segmentation task. The CT scans were collected between February and August 2018 at Al Hilla Teaching Hospital, Iraq. This dataset comprises 82 non-contrast CT scans, each containing 34 slices with a 5 mm slice thickness. Of the subjects, 56% (46 patients) were male, and 44% (36 patients) were female. The dataset includes individuals ranging in age from 1 day to 72 years, with

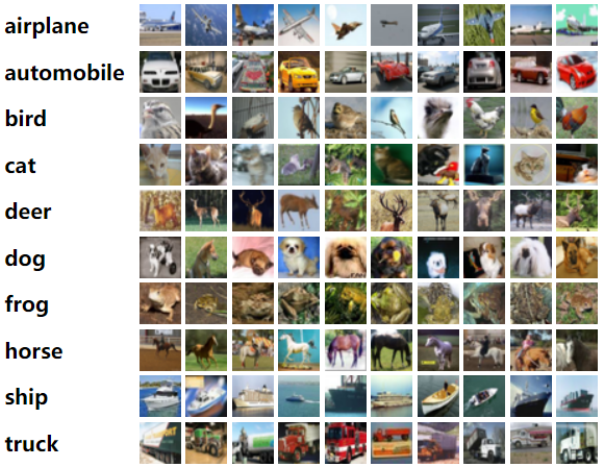
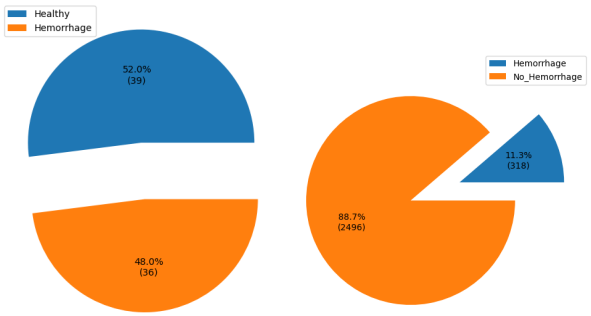


Fig. 9: Sample images from the CIFAR-10 dataset.

an average age of 27.8 ± 19.5 years. The broad age range impacts the skull's scale and shape observed in the CT scans and this variation

could affect the performance of the deep learning model. Out of the 82 patients, 36 were diagnosed with ICH. For data labeling, each slice was reviewed by two radiologists to determine the presence of hemorrhage or fractures [7].

As illustrated in Figure 10, the dataset consists of 2,814 slices, including 2,496 healthy slices and 318 slices with hemorrhage. This distribution exhibits a significant imbalance problem at the slice-level. However, at the patient-level, this imbalance problem is not observed. The imbalance at the slice-level is attributable to the fact that in patients with hemorrhage, many slices are healthy; in other words, hemorrhage occurs in only a few slices of a patient with ICH.



(a) (b)

Fig. 10: Distribution of slices (a) and patients (b) in the dataset.

During data collection, 7 CT scans from patients aged 59 to 65 were missed. Kyung et al. suggest that the CT scans of patients 58 and 79 exhibit poor image quality according to radiologist opinions and should be eliminated [8]. Consequently, 73 CT scans in NIFTI format are currently available. The CT scan images

conform to the Hounsfield Unit scale, thus a windowing operation should be applied to the slices. The brain window is the target window in this research, with the window level set to 40 and the window width set to 120 [5].

D. ARAS-CaSe Dataset

The ARAS-CaSe [9] dataset is a comprehensive resource for capsulorhexis segmentation in cataract surgery, addressing gaps in existing datasets. It consists of 533 annotated frames selected from over 800 surgical videos, ensuring diversity and high-quality representation of the surgical regions. The dataset includes annotations for cornea and needle using three methods: two-class occlusion-aware, three-class occlusion-aware, and two-class occlusion-unaware. ARAS-CaSe dataset in shown in Figure 11

III. Experiments

A. Multi-Layer KAN vs. Multi-Layer perceptron

The original KAN implementation is available in the pykan [10] repository. In the first step, a multi-layer KAN (MLKAN) has been established with this repository to train the network but this repository is inefficient and we can't establish the model properly on the GPU, thus other implementations of KAN are candidates to create an MLKAN model in python. Awsome-KAN [11] repository is a comprehensive collection of available implementations and extensions of KAN paper thus Efficient-KAN [12], Fast-KAN [13], and Faster-KAN [14] has been choose to implement a MLKAN model in pytorch.

1) Efficient KAN: Efficient-KAN The repository enhances the performance and speed of KAN by addressing memory and computational inefficiencies in the original implementation. Originally, expanding all intermediate variables to perform different activation functions required significant memory and processing power. By reformulating the computation to activate inputs with different basis functions (B-splines) and then combining them linearly, the memory cost is reduced and the process is streamlined into a straightforward matrix multiplication, efficient for both forward and backward passes.
It also adjusts regularization methods for compatibility and includes an option to toggle learnable activation function scales. [12]

2) Fast KAN: Fast-KAN improved the performance of KAN layers by replacing the 3rd-order B-spline basis with Gaussian Radial Basis Functions (RBFs) which the formula is indicated in Equation 6.

$$b_i(u) = \exp\left(-\left(\frac{u-u_i}{h}\right)^2\right) \tag{6}$$

This change makes the network significantly faster, reducing the forward time compared to the efficient KAN, while maintaining or slightly improving accuracy. The simplification comes from the ease of calculating RBFs while the grids are uniform. The approximate of the B-spline basis by RBF is illustrated in Figure 12. Additionally, LayerNorm is used to scale inputs, eliminating the need for grid adjustments. This approach also suggests that KANs can be seen as a type of RBF Network, bridging the concepts of RBF Networks and KANs. [13]

3) Faster KAN: Faster-KAN enhances the performance of KAN, Efficient-KAN, and FastKAN by experimenting with alternative basis functions, different exponent values, and varying the h parameter. One of the key improvements involves using the Reflectional Switch Activation Function (RSWAF) which uses the Equation 7 formula instead of RBF.

$$b_i(u) = 1 - \left(\tanh\left(\frac{u-u_i}{h}\right)\right)^2 \tag{7}$$

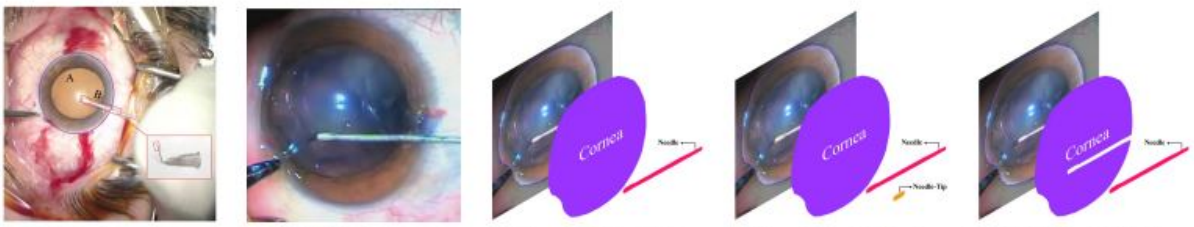
Figure 16 illustrates the RSWAF function, approximating the B-spline basis, while being computationally efficient with uniform grids. The latest version of the implementation allows users to choose whether the inverse of the denominator (1/h) should be a learnable parameter, although this feature is still experimental. Results show that RSWAF functions can closely approximate a 3rd-order B-spline basis for a network configuration of [28x28, 256, 10] in efficient-KAN, suggesting their potential as a viable alternative to traditional basis functions used in KANs. [14]

Table I indicate the comparison between the performance of different extensions and implementations of KAN [15]. Based on this table and by manually comparing the metrics of a sample structure, the Fast-KAN implementation has been chosen for further experiments since Fast-KAN is sufficiently fast and also approximates B-splines better than Faster-KAN.

B. Trial on MNIST

In this section, a grid search has been performed to obtain the best hyperparameters to train an MLKAN model on the MNIST dataset. This hyperparameter contains the shape of the input image and changes between 14×14 and 28×28, the number of grids to estimate the B-spline that a number between $3, 5, 50, 100$ is chosen for training the model, grid range that either is $(-1, 1)$ or $(-2, 2)$, Normalize the input vector, and the structure of the model that is chosen between [`inupt_size`, 10], [`inupt_size`, 64, 10], [`inupt_size`, 64, 16, 10].
To provide a proper comparison between the performance of MLKAN and MLP, 3 MLP models with the input shape of 14×14 and the structures that are mentioned for MLKAN models.

Table II shows the 20 configurations of training MLKAN which obtained the best performance on the MNIST dataset. Regarding that, the claim in the paper has been proved for input vectors with small lengths that increasing the parameters in each layer, will increase the



(a) Main Regions of Capsulorhexis Surgery (A: Cornea, B:Needle) (b) Original Frame (c) First Method (2-class, Occlusion-aware) (d) Second Method (3-class, Occlusion-aware) (e) Third Method (2-class, Occlusion-unaware)

Fig. 11: ARAS-CaSe Dataset

TABLE I: Comparison of performance different implementation of KAN

Implementation	forward	backward	forward	backward	num params	num trainable params
effkan-cpu	31.98 ms	44.49 ms	nan GB	nan GB	10010000	10010000
effkan-gpu	4.76 ms	4.54 ms	0.13 GB	0.19 GB	10010000	10010000
fast-kan-cpu	9.96 ms	17.06 ms	nan GB	nan GB	10015019	10015001
fast-kan-gpu	1.44 ms	2.13 ms	0.11 GB	0.14 GB	10015019	10015001
faster-kan-cpu	10.58 ms	15.42 ms	nan GB	nan GB	10014022	10014000
faster-kan-gpu	1.20 ms	2.01 ms	0.12 GB	0.14 GB	10014022	10014000
mlp-cpu	9.77 ms	7.27 ms	nan GB	nan GB	10020001	10020001
mlp-gpu	0.49 ms	1.07 ms	0.10 GB	0.13 GB	10020001	10020001
pykan-cpu	15.59 ms	17.53 ms	nan GB	nan GB	2431	1551
pykan-gpu	50.56 ms	93.93 ms	0.02 GB	0.02 GB	2431	1551

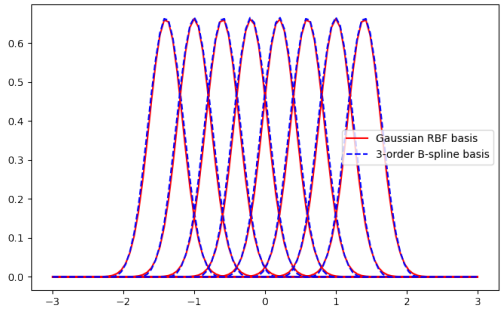


Fig. 12: Comparison of RBF and B-splines

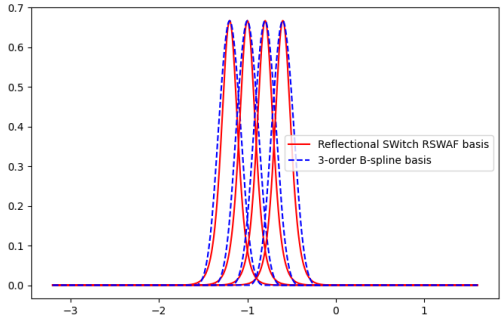


Fig. 13: Comparison of RSWAF and B-splines

performance more efficiently than increasing the number of layers. Moreover, the grid range does not change the performance significantly thus we can neglect its effect on the training data, but the number of grids and the image size will enhance the model performance significantly due to the normalization situation in the input vector, the best number values for this parameters are not predictable. As indicated in the last two columns in the table, the conversion speed of this model is remarkably high.

Table III shows the comparison of the performance of MLP and MLKAN on the MNIST dataset. The MLKAN model increased the accuracy metric on the MNIST up to 12 percent which is a considerable improvement for a model with few parameters. Also this model increased the speed of convergence.

The future work for this experiment contains using a validation set during training the model to examine the potential of overfitting in MLKAN, also assessing the performance of MLKAN and MLP models on external datasets could examine the potential of generalization of models.

C. Trial on Intracranial Hemorrhage Dataset

The classification of ICH lesions using MLKAN has been achieved through max-pooling the final layer of a ResNet50 model, which has been trained with an MLP head on the ICH dataset as a feature extractor. For this purpose, MLKAN was established instead of MLP layers, and the training process tuned the weights of the MLKAN model. A grid search was performed

TABLE II: The first 20 configuration on MNIST

layers hidden	img size	grid min	grid max	num grids	normalize	best train loss	best val loss	best train acc	best val acc	total epochs	epochs to //0.5 acc	epochs to 0.9 acc
(784, 64, 10)	28	-2	2	10	TRUE	0.053914	0.096377	0.987183	0.9702	65	2	10
(196, 64, 10)	14	-2	2	5	FALSE	0.080792	0.101838	0.978393	0.9701	75	2	9
(196, 64, 16, 10)	14	-1	1	5	TRUE	0.071126	0.100345	0.980108	0.9699	63	3	11
(196, 64, 10)	14	-2	2	10	TRUE	0.068108	0.097892	0.981953	0.9699	71	2	9
(196, 64, 16, 10)	14	-1	1	3	FALSE	0.073697	0.102042	0.980191	0.9697	76	4	11
(784, 64, 16, 10)	28	-1	1	5	TRUE	0.065518	0.101526	0.982945	0.9695	63	2	11
(784, 64, 10)	28	-2	2	5	FALSE	0.05638	0.09918	0.986428	0.9692	67	2	9
(196, 64, 10)	14	-1	1	5	TRUE	0.087578	0.107325	0.975562	0.9682	73	2	11
(784, 64, 10)	28	-2	2	3	FALSE	0.083164	0.10603	0.976666	0.9681	72	3	12
(196, 64, 10)	14	-2	2	5	TRUE	0.100513	0.110111	0.972459	0.9681	84	4	13
(196, 64, 16, 10)	14	-1	1	3	TRUE	0.102107	0.110764	0.970434	0.9677	78	4	16
(196, 64, 10)	14	-1	1	5	FALSE	0.077266	0.11894	0.980093	0.967	72	3	11
(196, 64, 10)	14	-1	1	3	FALSE	0.097135	0.113742	0.973002	0.9666	85	3	12
(196, 64, 16, 10)	14	-1	1	10	TRUE	0.060211	0.110875	0.98424	0.9665	56	3	10
(784, 64, 10)	28	-1	1	3	FALSE	0.073445	0.109347	0.980168	0.9665	71	2	11
(784, 64, 10)	28	-2	2	5	TRUE	0.095391	0.115146	0.97391	0.9658	67	3	11
(784, 64, 10)	28	-1	1	5	TRUE	0.079959	0.117288	0.978315	0.9658	77	2	12
(196, 64, 10)	14	-1	1	10	TRUE	0.062033	0.111746	0.984936	0.9656	69	2	9
(196, 64, 16, 10)	14	-1	1	5	FALSE	0.055092	0.115447	0.986746	0.9644	63	3	12
(784, 64, 16, 10)	28	-1	1	5	FALSE	0.03956	0.11598	0.992116	0.9644	57	2	11

TABLE III: The comparison of MLKAN and MLP on MNIST

Model	Hidden Layers	Accuracy on Test	Number of Epoch
MLP	[196, 10]	0.84	300
MLKAN	[196, 10]	0.94	86
MLP	[196, 64, 10]	0.85	300
MLKAN	[196, 64, 10]	0.97	74
MLP	[196, 64, 16, 10]	0.86	300
MLKAN	[196, 64, 16, 10]	0.97	62

to obtain the best hyperparameters for training the MLKAN model on the ICH dataset.

In the first step, a test subset was chosen at the patient level, and then the features of slices were extracted from ResNet50 to enable MLKAN to classify the ICH slices from healthy ones. The hyperparameters in the grid search included the number of grids to estimate the B-spline, which was chosen from the set 3, 5, 50, 100, and the model structure, which was chosen from the following configurations: [2048, 1], [2048, 10, 1], [2048, 512, 1], [2048, 1028, 256, 1], [2048, 256, 64, 1], [2048, 512, 128, 1], [2048, 512, 128, 32, 1], [2048, 1024, 256, 64, 1], [2048, 1024, 512, 256, 128, 1], [2048, 512, 128, 32, 8, 1], [2048, 512, 64, 256, 1288, 1].

For the second trial of training the MLKAN on the ICH dataset, a 5-fold cross-entropy was applied to the dataset, and the positive instances were augmented with a ratio of 3.3. In this phase, the hyperparameters were chosen more accurately based on the test loss from the previous section. This grid search included the number of grids to estimate the B-spline, chosen from the set 3, 5, 7, 10, a weighted loss of 2 and 4 for positive instances, and the model structure, chosen from the following configurations: [2048, 64, 1], [2048, 512, 128, 1], [2048, 1024, 256, 64, 1], [2048, 512, 128, 32, 1], [2048, 512, 128, 32, 8, 1], [2048, 512, 64, 256, 128, 1], [2048, 1024, 512, 256, 128, 1].

Table IV shows the results for training the model on the ICH dataset while the dataset is split into two subsections of train and test. These results do not show significant improvement in accuracy metric in comparison with the literature accuracy. based on these results, the more accurate hyperparameters have been chosen for training the 5-fold cross-validation.

In the next step, the training with the 5-fold cross-validation has been performed and since the weight of loss for each train is different, a simple BCE loss is considered as the global loss parameter to assess the performance of models. Table V shows the best performance of models on the validation sets, regarding it the model with the [2048, 1024, 256, 64, 1] architecture has been chosen for further examinations. The chosen model is trained on the folds and the best epoch of each mode selected by the loss value on the validation set and the voting mechanism applied to them. Figure 14 shows the performance of each fold and voting mechanism on the test set and there is no improvement in comparison to MLP layers. Figure 15 shows that this model misses more instances in comparison with MLP layers. In conclusion, it seems that MLKAN performs better tasks for data with a small number of inputs.

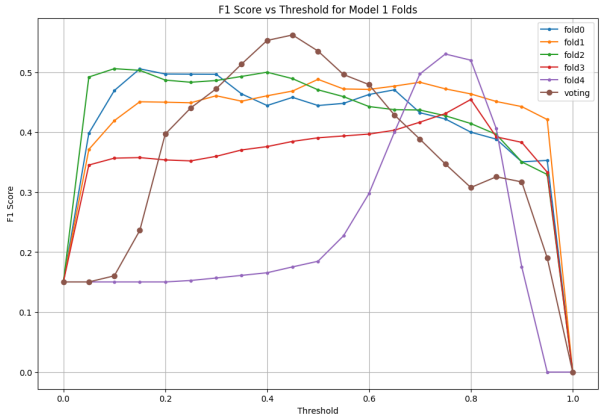


Fig. 14: F1 Score vs. Threshold

The future work for this experiment contains splitting the folds in patient-level scope to assess the overfitting issue on the validation set. Also training the MLKAN in an End-to-End approach is preferable to obtain the effect of KAN layers on the training of a CNN model.

IV. CNN KAN

We utilized the CNN-KAN GitHub [16] for implementing the Convolutional Kolmogorov-Arnold Networks (Convolutional KANs) described in the accompanying paper [17]. This repository provided the code

TABLE IV: The first 20 configurations on the ICH dataset for splitting the dataset to train and test

Hidden Layers	Num Grids	Best Train Loss	Best Test Loss	Best Train Acc	Best Test Acc	Epoch
(2048, 512, 128, 1)	5	0.060924	0.503722	0.989232	0.921875	30
(2048, 512, 128, 32, 1)	10	0.035903	0.507132	1	0.91875	52
(2048, 1)	3	1.063245	1.287778	0.778536	0.91875	29
(2048, 1024, 256, 64, 1)	3	0.09449	0.479004	0.983848	0.917188	28
(2048, 1028, 256, 1)	5	0.031139	0.658408	0.992462	0.915625	28
(2048, 1024, 256, 64, 1)	10	0.034609	0.537252	1	0.915625	39
(2048, 1028, 256, 1)	10	0.007805	0.606771	1	0.915625	31
(2048, 512, 1)	20	0.00289	0.844236	1	0.914063	28
(2048, 512, 128, 1)	3	0.101748	0.468515	0.982771	0.914063	28
(2048, 1024, 512, 256, 128, 1)	3	0.090147	0.456397	0.983848	0.914063	26
(2048, 512, 64, 256, 1288, 1)	5	0.015296	0.848396	0.997487	0.9125	29
(2048, 64, 1)	30	0.044446	0.579466	0.994257	0.9125	36
(2048, 256, 64, 1)	3	0.144993	0.486096	0.977746	0.910937	29
(2048, 1028, 256, 1)	3	0.141208	0.623723	0.981694	0.910937	27
(2048, 512, 128, 32, 1)	3	0.098838	0.456384	0.983489	0.910937	29
(2048, 512, 128, 32, 1)	5	0.066154	0.465782	0.98636	0.910937	34
(2048, 1024, 256, 64, 1)	5	0.030005	0.500429	0.995334	0.910937	28
(2048, 1024, 512, 256, 128, 1)	5	0.020189	0.48607	0.998923	0.910937	32
(2048, 512, 128, 32, 8, 1)	5	0.152308	0.437507	0.98636	0.909375	43
(2048, 512, 128, 1)	20	0.006973	0.577016	1	0.909375	56

TABLE V: The first 20 configurations on the ICH dataset for 5-fold-cross-validation

Model Structure	Grid Num	Weight	Best Average Val Loss	Best Global Average Val Loss	Best Average Val Accuracy	Best Global Test Loss	Best Test Accuracy
2048, 1024, 256, 64, 1	10	2	0.017461504	0.014333	0.996614	0.241029	0.9225
2048, 1024, 256, 64, 1	10	4	0.036286868	0.025592	0.996628	0.23747	0.922187
2048, 1024, 256, 64, 1	3	2	0.104714902	0.065235	0.979987	0.23206	0.923125
2048, 1024, 256, 64, 1	3	4	0.032171533	0.024149	0.997124	0.22693	0.923125
2048, 1024, 256, 64, 1	5	2	0.018391454	0.016081	0.997103	0.256471	0.922187
2048, 1024, 256, 64, 1	5	4	0.022752248	0.016763	0.996614	0.247552	0.922812
2048, 1024, 256, 64, 1	7	2	0.021816487	0.016254	0.995675	0.2157	0.923438
2048, 1024, 256, 64, 1	7	4	0.023228022	0.017508	0.997103	0.216536	0.9225
2048, 1024, 512, 256, 128, 1	10	2	0.019457832	0.015526	0.996626	0.209025	0.925625
2048, 1024, 512, 256, 128, 1	10	4	0.070145403	0.045674	0.995159	0.210852	0.925
2048, 1024, 512, 256, 128, 1	3	2	0.021470506	0.017471	0.997591	0.209437	0.925313
2048, 1024, 512, 256, 128, 1	3	4	0.034717674	0.029203	0.996626	0.212343	0.9225
2048, 1024, 512, 256, 128, 1	5	2	0.013685671	0.010417	0.997591	0.213766	0.925625
2048, 1024, 512, 256, 128, 1	5	4	0.065259795	0.035681	0.99425	0.240313	0.923438
2048, 1024, 512, 256, 128, 1	7	2	0.017807907	0.014353	0.997591	0.212815	0.924375
2048, 1024, 512, 256, 128, 1	7	4	0.025217444	0.018876	0.996616	0.210845	0.924688
2048, 512, 128, 1	10	2	0.016161421	0.01339	0.99663	0.29069	0.9225
2048, 512, 128, 1	10	4	0.023245117	0.016201	0.996152	0.269668	0.916875
2048, 512, 128, 1	3	2	0.018316118	0.016398	0.995648	0.304341	0.924688
2048, 512, 128, 1	3	4	0.025356819	0.017691	0.996714	0.339419	0.919687

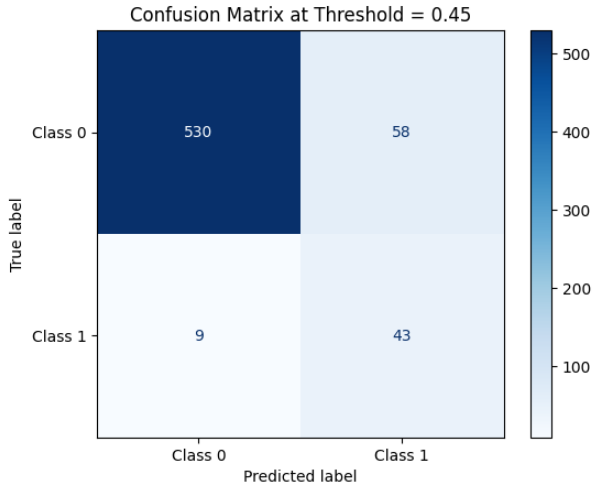


Fig. 15: Confusion Matrix of Voting Mechanism

necessary to build and test these innovative neural network architectures, which integrate the non-linear activation functions of Kolmogorov-Arnold Networks (KANs) with convolutional layers. The aim was to validate their performance on the MNIST dataset and compare it against traditional Convolutional Neural Networks (CNNs).

Different experiments were conducted to analyze the performance of various models employing KAN Convolutional Layers compared to a classical convolutional neural network, and these experiments are described. Two datasets were utilized during experimentation: The MNIST dataset, which was the focus of the main paper, and the CIFAR-10 dataset, which was mentioned as a potential future work in the original article, have been utilized in this project. Each architecture was trained on both datasets to generate the models used for performance comparison. The proposed architectures incorporated a mix of Linear, Kan Linear, Kan Con-

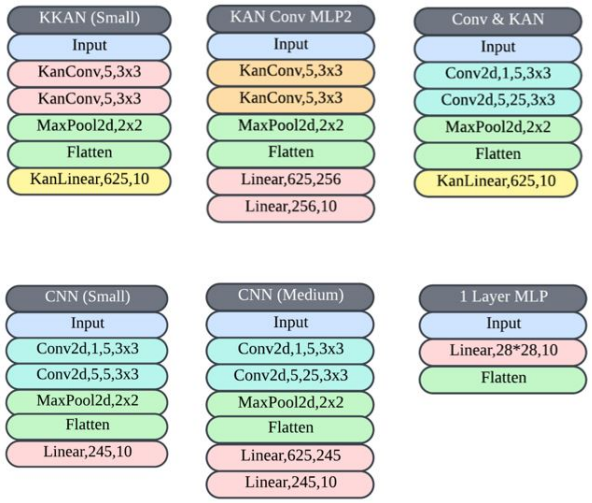


Fig. 16: Convolutional KAN and standard architectures used in experimentss

TABLE VI: Comparison of accuracy, precision, recall, F1 score and parameter count for the proposed models tested on the MNIST Dataset

Model	Accuracy	percision	Recall	F1 Scre	Params
MNIST Dataset					
KKAN(Small)	98.67	98.67	98.65	98.66	94.87K
Conv & KAN	98.25	98.24	98.24	98.25	95K
KAN Conv & 2 Layer MLP	98.48	98.48	98.47	98.47	164K
CNN (Medium)	99.04	99.03	99.02	99.03	157K
CNN (Small)	97.80	97.80	97.78	97.79	2.7K
1 Layer MLP	92.32	92.22	92.21	92.20	7.9K

volutional, and Convolutional layers.Figure 16 shows different architectures used:

A. CNN result

This section presents an analysis of the performance of the various proposed models in the previously described experiments.

1) MNIST Dataset: Table VI provides an analysis of performance metrics, including accuracy, precision, recall, F1 score, number of parameters, and training time per epoch for various models tested on the MNIST dataset. This analysis aids in understanding the efficiency and effectiveness of each model configuration.

2) CIFAR Dataset: Similar to Table VI with the MNIST dataset, Table VII presents a comparison of accuracy, precision, recall, F1 score, parameter count, and training time per epoch for the proposed models tested on the CIFAR-10 dataset.

TABLE VII: Comparison of accuracy, precision, recall, F1 score and parameter count for the proposed models tested on the CIFAR-10 Dataset

Model	Accuracy	percision	Recall	F1 Scre	Params
CIFAR-10 Dataset					
KKAN(Small)	48_92	48.26	48.92	48.37	94.87K
Conv & KAN	44.71	44.31	44.71	44.33	95K
KAN Conv & 2 Layer MLP	48.02	47.43	48.02	47.57	164K
CNN (Medium)	49.81	49.32	49.81	49.36	157K
CNN (Small)	41.26	40.82	41.26	40.81	2.7K
1 Layer MLP	30.76	30.00	30.76	30.05	7.9K

V. Segmentation KAN

In this work, we utilized the repository U-KAN [18] developed by the CUHK-AIM Group. U-KAN enhances the U-Net architecture by incorporating Kolmogorov-Arnold Network (KAN) layers, optimizing medical image segmentation and generation tasks.

"U-KAN Makes Strong Backbone for Medical Image Segmentation and Generation," explores the integration of Kolmogorov-Arnold Networks (KANs) into the U-Net architecture to enhance its performance in medical image segmentation and generation tasks. U-Net, a widely used model for image segmentation, is augmented with KAN layers to create a new model named U-KAN. This model leverages non-linear learnable activation functions inspired by the Kolmogorov-Arnold representation theorem, aiming to improve both accuracy and interpretability over traditional linear methods. The authors validate the U-KAN model through rigorous benchmarks in medical image segmentation, demonstrating that it achieves higher accuracy with reduced computational cost compared to existing methods. Additionally, U-KAN's potential as an alternative U-Net noise predictor in diffusion models is explored, showing promising results in generating task-oriented model architectures. U-KAN architecture is shown in Figure 18.

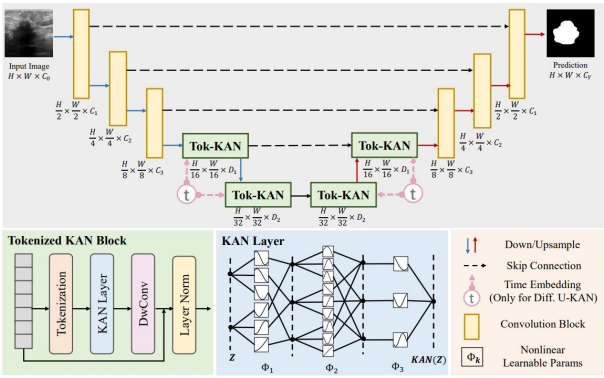


Fig. 17: Overview of U-KAN pipeline.

We tested U-KAN on the provided datasets (BUSI, GLAS, and CVC-ClinicDB) and our own dataset. The results demonstrated significant improvements in segmentation accuracy with reduced computational costs. The adaptability of U-KAN in different medical imaging contexts underscores its potential as a robust backbone for medical image analysis.

A. Segmentation Results

To evaluate the performance of our segmentation model, we conducted an experiment over 10 epochs, measuring the model's accuracy using Intersection over Union (IoU) and Dice Coefficient as the primary metrics. These metrics are widely used in image segmentation tasks to assess the overlap between the predicted segmentation mask and the ground truth.

Our model achieved a validation IoU of 0.94 and a validation Dice Coefficient of 0.97 after 10 epochs. These results indicate a high degree of accuracy and robustness in the model's ability to delineate the target regions within the images. The high IoU score reflects that the predicted and actual segmentation masks have significant overlap, while the Dice Coefficient further confirms the precision and recall of our model in segmenting the desired regions.

The Figure 18 illustrates the segmentation results obtained from our model. The images demonstrate the effectiveness of our approach in accurately segmenting the target areas, showcasing both the visual and quantitative performance of our model.

VI. Conclusion

References

[1] Wolfram MathWorld. B-spline. https://mathworld.wolfram.com/B-Spline.html. Accessed: 2024-07-20.

[2] Nicholas M. Patrikalakis, Takashi Maekawa, and G. Yu Cho. B-splines. https://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node17.html. Accessed: 2024-07-20.

[3] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[4] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[5] Murtadha Hssayeni, M Croock, A Salman, H Al-khafaji, Z Yahya, and B Ghoraani. Computed tomography images for intracranial hemorrhage detection and segmentation. *Intracranial hemorrhage segmentation using a deep convolutional model. Data*, 5(1):14, 2020.

[6] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[7] Murtadha D Hssayeni, Muayad S Croock, Aymen D Salman, Hassan Falah Al-Khafaji, Zakaria A Yahya, and Behnaz Ghoraani. Intracranial hemorrhage segmentation using a deep convolutional model. *Data*, 5(1):14, 2020.

[8] Sunggu Kyung, Keewon Shin, Hyunsu Jeong, Ki Duk Kim, Jooyoung Park, Kyungjin Cho, Jeong Hyun Lee, GilSun Hong, and Namkug Kim. Improved performance and robustness of multi-task representation learning with consistency loss between pretexts for intracranial hemorrhage identification in head ct. *Medical Image Analysis*, 81:102489, 2022.

[9] Iman Gandomi, Mohammad Vaziri, Mohammad Javad Ahmadi, M Reyhaneh Hadipour, Parisa Abdi, and Hamid D Taghirad. A deep dive into capsulorhexis segmentation: From dataset creation to sam fine-tuning. In *2023 11th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pages 675–681. IEEE, 2023.

[10] Xiaoming. GitHub - KindXiaoming/pykan: Kolmogorov Arnold Networks — github.com. https://github.com/KindXiaoming/pykan. [Accessed 20-07-2024].

[11] GitHub - mintisan/awesome-kan: A comprehensive collection of KAN(Kolmogorov-Arnold Network)-related resources, including libraries, projects, tutorials, papers, and more, for researchers and developers in the Kolmogorov-Arnold Network field. — github.com. https://github.com/mintisan/awesome-kan. [Accessed 20-07-2024].

[12] GitHub - Blealtan/efficient-kan: An efficient pure-PyTorch implementation of Kolmogorov-Arnold Network (KAN). — github.com. https://github.com/Blealtan/efficient-kan. [Accessed 20-07-2024].

[13] GitHub - ZiyaoLi/fast-kan: FastKAN: Very Fast Implementation of Kolmogorov-Arnold Networks (KAN) — github.com. https://github.com/ZiyaoLi/fast-kan. [Accessed 20-07-2024].

[14] GitHub - AthanasiosDelis/faster-kan: Benchmarking and Testing FastKAN — github.com. https://github.com/AthanasiosDelis/faster-kan. [Accessed 20-07-2024].

[15] GitHub - Jerry-Master/KAN-benchmarking: Benchmark for efficiency in memory and time of different KAN implementations. — github.com. https://github.com/Jerry-Master/KAN-benchmarking. [Accessed 20-07-2024].

[16] GitHub - AntonioTepsich/Convolutional-KANs: This project extends the idea of the innovative architecture of Kolmogorov-Arnold Networks (KAN) to the Convolutional Layers, changing the classic linear transformation of the convolution to learnable non linear activations in each pixel. — github.com. https://github.com/AntonioTepsich/Convolutional-KANs. [Accessed 20-07-2024].

[17] Alexander Dylan Bodner, Antonio Santiago Tepsich, Jack Natan Spolski, and Santiago Pourteau. Convolutional kolmogorov-arnold networks. *arXiv preprint arXiv:2406.13155*, 2024.

[18] Chenxin Li, Xinyu Liu, Wuyang Li, Cheng Wang, Hengyu Liu, and Yixuan Yuan. U-kan makes strong backbone for medical image segmentation and generation. *arXiv preprint arXiv:2406.02918*, 2024.

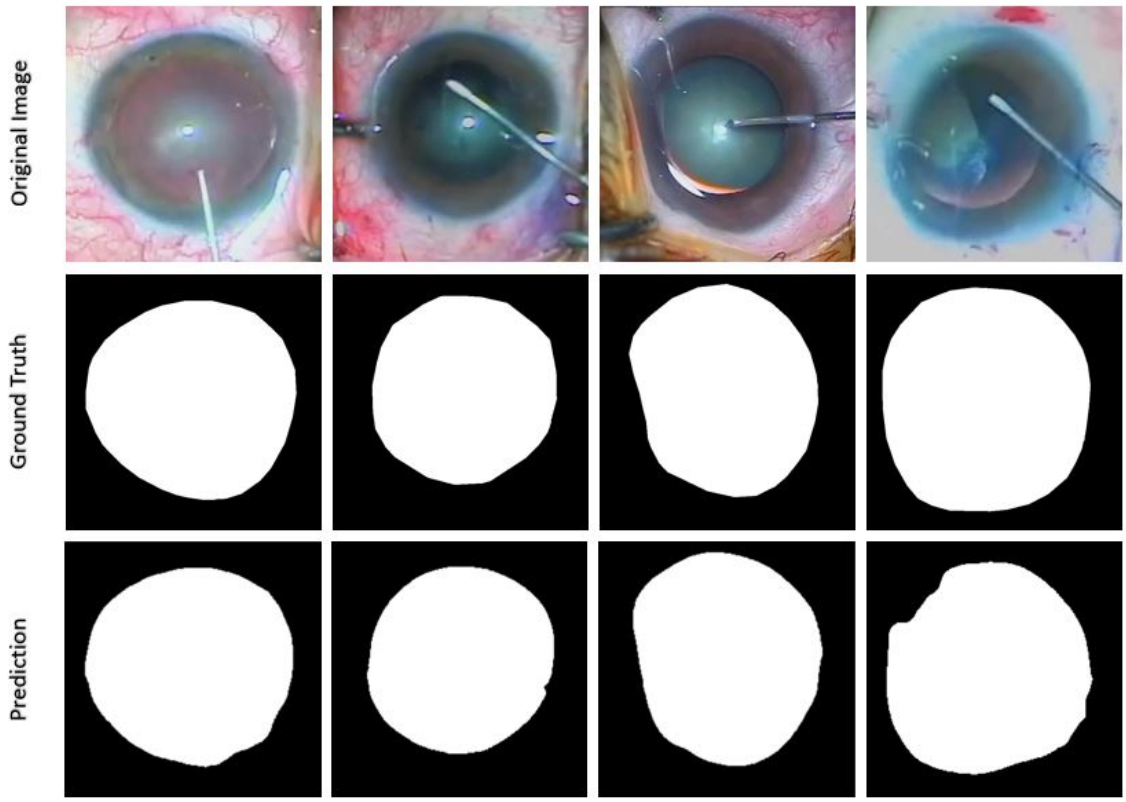


Fig. 18: Visual representation of segmentation results from our model. The images show the original images (first row) ground truth segmentation masks (second row), the predicted masks (third row)