Suitability of KANs for Computer Vision: A preliminary investigation

Basim Azam, Naveed Akhtar

School of Computing and Information Systems

The University of Melbourne, Melbourne, Australia

Email: {basim.azam, naveed.akhtar1}@unimelb.edu.au

Abstract—**Kolmogorov-Arnold Networks (KANs) introduce a paradigm of neural modeling that implements learnable functions on the edges of the networks, diverging from the traditional node-centric activations in neural networks. This work assesses the applicability and efficacy of KANs in visual modeling, focusing on the image recognition task. We mainly analyze the performance and efficiency of different network architectures built using KAN concepts along with conventional building blocks of convolutional and linear layers, enabling a comparative analysis with the conventional models. Our findings are aimed at contributing to understanding the potential of KANs in computer vision, highlighting both their strengths and areas for further research. Our evaluation[1] shows that whereas KAN-based architectures perform in-line with the original claims of [1] for performance and model-complexity in the case of simpler vision datasets like MNIST, the advantages seem to diminish even for slightly more complex datasets like CIFAR-10.**

I. INTRODUCTION

Neural networks have been foundational in advancing the field of machine learning, particularly in the tasks requiring complex pattern recognition such as image and speech processing [2]–[4]. Traditionally, Multi-Layer Perceptrons (MLPs) have been a cornerstone in neural network architectures [5], [6]. MLP is a fully connected network consisting of multiple layers of nodes, each applying a nonlinear activation function to a weighted sum of inputs from the previous layer [7], [8]. This configuration has proved potent in approximating nonlinear functions across various domains [9], [10].

MLPs are often challenged by the demands of high-dimensional data, such as images [11]. This stems from their structure, which lacks the capacity to inherently capture spatial patterns in the data, resulting in scalability and efficiency concerns. Consequently researchers investigate architectures that can handle such complexities more effectively. Convolutional Neural Networks (CNNs) [12] have emerged as a powerful class of neural networks that are particularly suited for analyzing visual imagery [13], [14]. Building upon the foundational principles of MLPs, CNNs adapt and extend basic concepts to more effectively model the intricate structures of input data. CNNs are designed to automatically and adaptively learn spatial hierarchies of features through backpropagation [15], [16]. This capability makes them effective for tasks such as image recognition [13], object detection [17], [18], and segmentation [19], [20].

The recent proposition of Kolmogorov-Arnold Networks (KANs) [1], [21], [22] introduces an innovative modification to the traditional neural networks by employing learnable functions on the edges of the graph rather than using fixed activation functions at the nodes. This architecture is inspired by the Kolmogorov-Arnold Representation Theorem [23], which posits that any multivariate continuous function can be decomposed into univariate functions and a summing operation. In KANs, each element within the layer applies a unique, learnable function to its inputs. In their seminal work, Liu et al. [1] claimed that KANs can address some of the intrinsic limitations of MLPs, particularly in handling complex functional mappings in high-dimensional spaces.

Building on theoretical foundations of KAN, the application of this type of neural architectures to visual data presents a novel area of study. Given the structured nature of image data and the critical role of efficient function approximation in image recognition and classification, KANs could offer significant advantages. Preliminary attempts of ConvKANs [24] (the convolutional adaptation of KANs) have shown promising results by replacing the traditional dot product in convolution operations with a learnable non-linear activation, adapting the flexible and powerful Kolmogorov-Arnold theorem directly into the processing of visual information.

Several implementations and research efforts are currently being made to explore the potential of KANs in various domains. The Convolutional-KANs project [24] extends the idea of KANs to convolutional layers, replacing the classic linear transformation of the convolution with learnable non-linear activations in each pixel. The Vision-KAN repository [25] explores the possibility of MLPs with KANs in Vision Transformers. The KAN-GPT project [26] implements Generative Pre-trained Transformers (GPTs) using KANs for language modeling tasks, demonstrating the potential of KANs in natural language processing applications. The KAN-GPT-2 repository [27] trains small GPT-2 style models using KANs instead of MLPs. The KANeRF project [28] integrates KANs into Neural Radiance Fields (NeRF) for view synthesis tasks. The KAN-UNet repository [29] implements a U-Net architecture with Kolmogorov-Arnold Convolutions (KA con-

---

[1]This manuscript is a work in progress, which includes results from an ongoing exploration in this direction. The results and discussion are expected to evolve over time. This independent effort is intended to contributes to our understanding of KANs for visual modeling, and not to provide a yes/no answer to the question 'if KANs should replace conventional neural modeling techniques for computer vision'.

volutions) for image segmentation tasks. Similarly, the kansformers project [30] explores the use of KANs in Transformer architectures, replacing the traditional linear layers with KAN layers.

Although there is currently a large interest in developing implementation repositories providing various KAN variants pertinent to visual modeling, a formal scientific effort focusing on the suitability of KANs in general and the emerging repositories in particular for visual modeling tasks is still missing. This work is aimed at partially addressing this gap as a preliminary effort focusing on providing independent empirical evidence of KANs' suitability for vision in the context of image recognition task. We replicate ideas from Convolutional-KANs [24] and torch-conv-kan [31] implementations to evaluate their performance in terms of accuracy, training efficiency, and model parameters while experimenting with datasets such as MNIST and CIFAR. Our experiments are expected to provide insights into the strengths and limitations of KANs in handling complex visual data. The study aims at helping understand answers to the following questions:

1) How do KANs perform in terms of accuracy, training efficiency, and model parameters in the context of visual modeling?
2) Can KANs be effectively integrated into existing CNN frameworks to enhance their performance or efficiency?
3) What are the potential limitations or challenges in adapting KANs to complex visual tasks?

This manuscript is structured to provide an assessment of KANs' abilities for visual modeling (considering image recognition task). Following this introduction, we present a theoretical formulation of KANs and their extension to ConvKANs. We then describe the experimental setup, implementation details, and results of applying KANs to standard datasets. The subsequent sections provide discussion on the validation of claims regarding KANs in [1], followed by a critical analysis of their suitability for the broader applications in computer vision. The manuscript concludes with a summary of the findings and directions for future research. This document will be periodically updated as new data becomes available and further experiments are conducted.

II. THEORETICAL FORMULATION

This section explores the mathematical foundations and theoretical constructs of KANs [1] and their adaptation into convolutional neural networks [24]. The foundational novelty of KANs is in learning parametric functions on graph edges. This concept can be used in traditional fully-connected layers of a network as well as convolutional layers. We use ConvKAN as a general term for the convolutional networks leveraging KAN concepts. When both convolutional and linear layers are implemented with KANs, we often refer to the network as KConvKAN to particularly emphasize on the usage of KAN-based convolutional layers. Figure 1 provides the nomenclature used in this manuscript for a quick reference.

A. Multi-Layer Perceptrons (MLPs)

A Multi-Layer Perceptron (MLP) is a fully connected feedforward neural network consisting of multiple layers of nodes (neurons). Each node in a layer is connected to every node in the subsequent layer, and the node applies a nonlinear activation function to the weighted sum of its inputs.

Universal Approximation Theorem [7] provides foundations to the popularity of MLPs. The theorem states that a feedforward network with a single hidden layer containing a finite number of neurons can approximate any continuous function on compact subsets of $\mathbb{R}^n$, given appropriate activation functions. Mathematically, an MLP with L layers can be expressed as:

$$\text{MLP}(\mathbf{x}) = \sigma_L(\mathbf{W}_L \sigma_{L-1}(\mathbf{W}_{L-1} \cdots \sigma_1(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \cdots \\ + \mathbf{b}_{L-1}) + \mathbf{b}_L), \quad (1)$$

where:
- $\mathbf{x}$ is the input vector.
- $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and bias vector for the l-th layer.
- σ_l is the activation function for the l-th layer.

B. Kolmogorov-Arnold Networks (KANs)

Here, we present the concepts related to KANs by building on the work of Liu et al. [1]. Interested readers are referred to [1] for more details.

1) Kolmogorov-Arnold Representation Theorem: The Kolmogorov-Arnold Representation Theorem [32] states that any multivariate continuous function can be represented as a finite composition of continuous functions of a single variable and the binary operation of addition. Specifically, for a smooth function $f : [0, 1]^n \to \mathbb{R}$,

$$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right), \quad (2)$$

where $\phi_{q,p}$ and Φ_q are continuous functions and x_p denotes the pth component of the input vector $\mathbf{x}$.

2) KAN Architecture: KANs generalize the Kolmogorov-Arnold representation by using learnable activation functions on graph edges. A KAN layer with n_{in}-dimensional inputs and n_{out}-dimensional outputs is defined as:

$$\mathbf{\Phi} = \{\phi_{q,p}\}, \quad p = 1, 2, \ldots, n_{\text{in}}, \quad q = 1, 2, \ldots, n_{\text{out}} \quad (3)$$

where $\phi_{q,p}$ are learnable functions, parameterized as splines. The output of a KAN layer is given by:

$$\mathbf{x}_{l+1} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \quad j = 1, \ldots, n_{l+1}. \quad (4)$$

In matrix form it can be expressed as:

$$\mathbf{x}_{l+1} = \mathbf{\Phi}_l \mathbf{x}_l, \quad (5)$$

where $\mathbf{\Phi}_l$ is the function matrix corresponding to the lth KAN layer. KANs are claimed to possess faster neural scaling laws
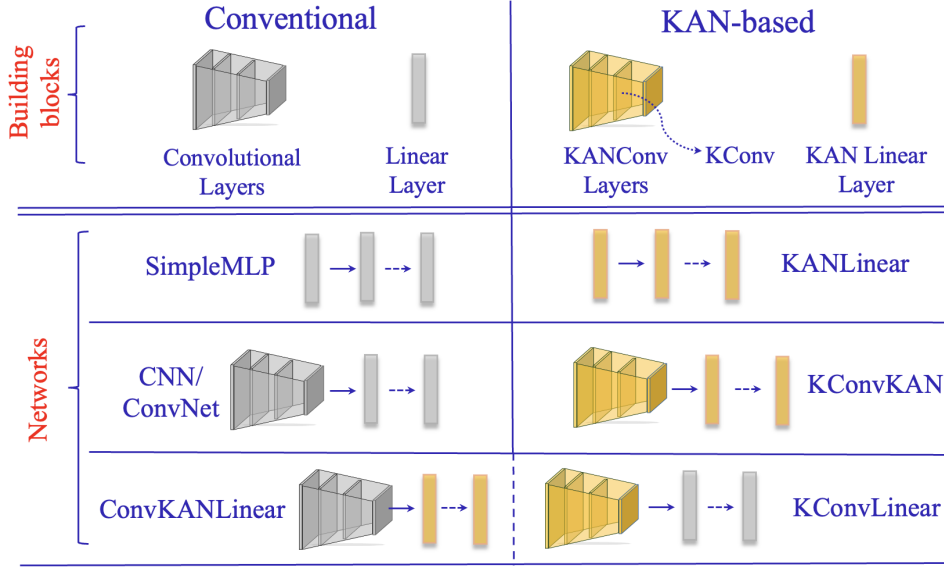
Fig. 1. Categorization of the types of network architectures used in this work. We employ KAN-based building blocks with conventional layers to construct different types of networks. The same naming conventions are used throughout this work.

than MLPs [1]. In particular, the approximation bound for KANs is given by:

$$\left\| f - \left(\mathbf{\Phi}_{L-1}^{G} \circ \cdots \circ \mathbf{\Phi}_{0}^{G} \right) \mathbf{x} \right\|_{C^m} \leq CG^{-k-1+m}, \quad (6)$$

where G is the grid size, k is the order of the B-spline, and C is a constant. We followed the notational convention from [1], where $\mathbf{\Phi}_{L-1}^{G} \circ \cdots \circ \mathbf{\Phi}_{0}^{G}$ denotes the composition of the functions matrices from layer 0 to layer $L-1$. The composition operation $\circ$ indicates the output of one funciton matrix is used as the input to the next function.

C. Extension to ConvKANs

1) Convolutional Neural Networks (CNNs): Convolutional neural networks [12] are a class of deep neural networks that use convolutional layers to process data with a grid-like topology, such as images. This makes them ideally suitable for computer vision tasks. A convolutional layer applies a set of learnable filters (kernels) to the input, producing feature maps. Mathematically, the output feature $y_{i,j}$ computed at location (i,j) on the featured grid by a convolutional kernel k is expressed as:

$$y_{i,j} = \sum_{m,n} x_{i+m,j+n} k_{m,n}. \quad (7)$$

2) KAN-based Convolutions: The natural idea of extending KAN concept to convolutions is to replace the scalar product in the convolution operation used in CNNs with a learnable non-linear activation function applied to each element. Given the pre-existence of CNNs, this extension naturally emerges from the foundational concept of KANs. Hence, we avoid associating credit of ConvKANs to any specific contribution. We can define the convolutional kernel implemented for ConvKANs as:

$$\text{ConvKAN Kernel} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix}, \quad (8)$$

and the computation of a KAN-based convolutional kernel is given by:

$$y_{i,j} = \sum_{m,n} \phi_{m,n}(x_{i+m,j+n}), \quad (9)$$

where $\phi_{m,n}$ are learnable functions parameterized as splines. Henceforth, for brevity, we refer to a KAN-based Convolutional layer as KANConv layer. Note the difference between ConvKAN, which denotes a network and KANConv, which is used to identify a convolutional layer. We abbreviate KAN-Conv as KConv for KConvKANs.

D. Further Details

1) KANConv Layer: Consider a KAN layer with input $\mathbf{x} \in \mathbb{R}^{n_{\text{in}}}$ and output $\mathbf{y} \in \mathbb{R}^{n_{\text{out}}}$. The layer applies a matrix of learnable functions $\mathbf{\Phi}$:

$$\mathbf{y} = \mathbf{\Phi}(\mathbf{x}), \quad (10)$$

where $\mathbf{\Phi} = \{\phi_{q,p}\}$ and $\phi_{q,p}$ are parameterized as B-splines. On similar lines, consider a KANConv layer with input $\mathbf{x} \in \mathbb{R}^{H \times W \times C_{\text{in}}}$ and output $\mathbf{y} \in \mathbb{R}^{H' \times W' \times C_{\text{out}}}$. The layer applies a set of KAN based kernels $\mathbf{\Phi}$:

$$y_{i,j,c} = \sum_{m,n} \phi_{m,n,c}(x_{i+m,j+n}) \quad (11)$$

where $\phi_{m,n,c}$ are parameterized as B-splines.

For a $K \times K$ kernel, each element of the kernel has a learnable function ϕ with parameter count $G + 2$, where G is the grid size. The total parameter count for a KANConvs layer is $K^2(G + 2)$.

Based on the original proposal [1], each learnable function ϕ in KANs is parameterized as a B-spline:

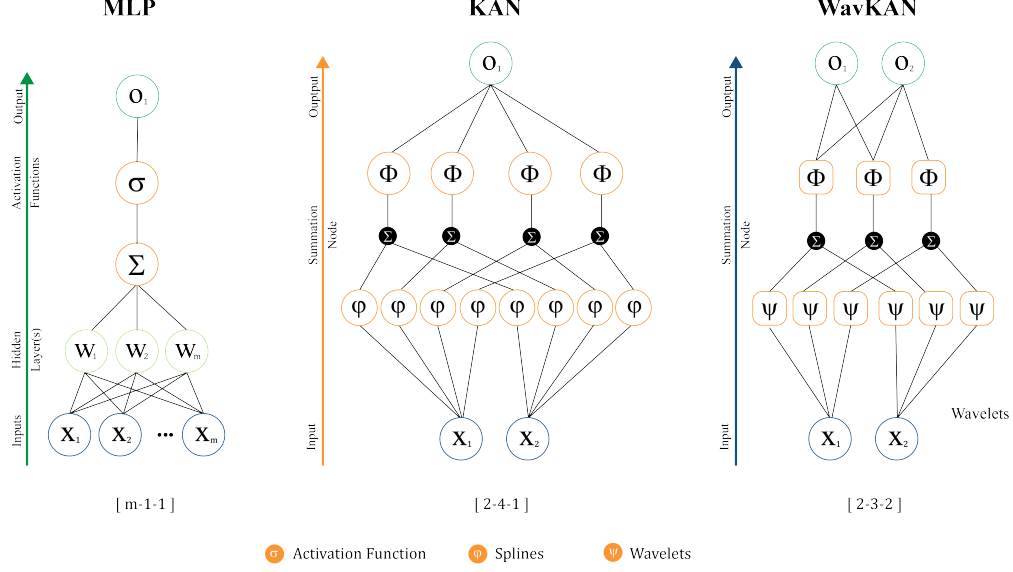$$\phi(x) = w_1 \cdot \text{spline}(x) + w_2 \cdot b(x), \quad (12)$$

Fig. 2. A high-level comparison of basic network configurations using Multi-Layer Perceptrons (MLP), Kolmogorov-Arnold Networks (KAN), and Wavelet KAN. KAN-based models use learnable functions on edges instead of applying fixed activation functions on nodes/neurons. Traditional KAN and WavKAN mainly differ in the types of functions used. Number of nodes in network layers are mentioned at the bottom.

where $b(x)$ is a basis function (e.g., SiLU), and spline(x) is a linear combination of B-splines.

KANConv layers provide a flexible and adaptive filtering process, enabling nuanced and controllable transformations of input data, which may lead to learning more complex patterns efficiently. Figure 2 provides a high-level comparison to understand the structural composition of MLP, KAN, and WavKAN. An architecture based entirely on KAN Linear layers and KANConv layers is termed KConvKAN herein.

E. Wavelet KANs (WavKANs)

The Wavelet Transform [33], [34] is utilized in signal processing to analyze the frequency content of a signal as it varies over time. The wavelet based extension of KANs, WavKAN [35] integrates wavelets into the convolutional framework to enhance feature extraction. The continuous wavelet transform employs a fundamental function called the "mother wavelet." This function acts as a prototype that can be adjusted in scale and position to align with different segments of the signal. The form of the mother wavelet is crucial because it determines which aspects of the signal are more critical.

WavKAN employs a mother wavelet $\psi \in L^2(\mathbb{R})$ to analyze the signal $g(t) \in L^2(\mathbb{R})$. A mother wavelet must satisfy the following criteria:

1) **Zero Mean**:

$$\int_{-\infty}^{\infty} \psi(t)\, dt = 0. \tag{13}$$

2) **Admissibility Condition**:

$$C_\psi = \int_0^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega}\, d\omega < +\infty. \tag{14}$$

where $\hat{\psi}$ is the Fourier transform of the wavelet $\psi(t)$.

The continuous wavelet transform of a function is represented by wavelet coefficients:

$$C(s,\tau) = \int_{-\infty}^{+\infty} g(t) \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) dt, \tag{15}$$

where:

- $g(t)$ is the signal/function to be approximated by the Wavelet basis.
- $\psi(t)$ is the mother wavelet.
- $s \in \mathbb{R}^+$ is the scale factor.
- $\tau \in \mathbb{R}$ is the shift factor.
- $C(s,\tau)$ measures the match between the wavelet and the signal at scale s and shift τ.

A signal can be reconstructed from its wavelet coefficients using the inverse CWT:

$$g(t) = \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_0^{+\infty} C(s,\tau) \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) \frac{ds\, d\tau}{s^2}. \tag{16}$$

where C_ψ is a constant that ensures the reconstruction's accuracy.

In WavKANConv layers, the wavelet transform is integrated into the convolutional operation. Each wavelet-transformed feature is then processed through the KAN framework, allowing for localized wavelets combined with the adaptive learning capabilities of KANs.

A WavKANConv layer operation with input $\mathbf{x} \in \mathbb{R}^{H \times W \times C_{\text{in}}}$ and output $\mathbf{y} \in \mathbb{R}^{H' \times W' \times C_{\text{out}}}$ can be expressed as:

$$y_{i,j_c} = \sum_{m,n} \phi_{m,n,c} \left(\int x_{i+m,j+n} \cdot \frac{1}{\sqrt{s_c}} \psi_c \left(\frac{t - \tau_{m,n}}{s_c} \right) dt \right),$$

(17)

where $x_{i+m,j+n}$ represents the input data at position $(i + m, j + n)$, $\psi_c(t, s_c, \tau_{m,n})$ denotes the mother wavelet function associated with channel c, $\frac{1}{\sqrt{s_c}}$ is the normalization factor for the wavelet function, and $\phi_{m,n,c}$ are learnable functions. WavKANConvs provide a powerful extension to KANs, incorporating wavelet-based frequency analysis into the adaptive filtering process of convolutional networks. This combination enables the extraction of complex patterns with enhanced localization in both time and frequency domains, potentially leading to superior performance in various visual tasks. The deeper and complex architectures are presented based on these adaptations, the detailed workings and configuration of KConvKANs is described in the next section.

III. EXPERIMENTS

This section details the experimental setup and procedures used to evaluate the performance of networks on MNIST and CIFAR-10 datasets (in this version). The goal of these experiments is to assess the suitability of KANs for the fundamental computer vision task of recognition, focusing on aspects such as accuracy, model size, training time, and parameter efficiency. The architectural overview and the internal visualizations of a KConvKAN used in our experiments are illustrated in Fig. 3 for a clear understanding. The shown KConvKAN in Fig. 3(a) depicts the flow of data through the network that consists of two KAN Convolutional (KANConv) layers, each followed by a MaxPool2D layer, then a Flatten layer, and finally a KAN Linear layer that maps the flattened output of the convolutional layers to the final classification output via a log-softmax function. Fig. 3(b) provides insights into the feature maps and learned spline weights at different stages of the KConvKAN when processing a sample image. The feature map progression is shown after each KANConv and MaxPool2D layer. The 3D spline weights and heat map representation of spline weights in the KAN Linear layer are also visualized. The visualization is meant to demonstrate the transformation of the input through the network layers for understanding.

A. Experimental Setup

The experiments are carried out on a single node of the Spartan High Performance Computing (HPC) system at the University of Melbourne, equipped with 4 NVIDIA A100 GPUs (80GB each), 495GB of RAM, and 32 CPU cores. One GPU was allocated from the gpu-a100 partition for each experiment, ensuring robust computational support for our deep learning models. In our experiments, we utilized the AdamW optimizer with a learning rate of 1×10^{-3} and varying levels of weight decay to enhance regularization. Training sessions spanned several epochs, managed under an exponential learning rate scheduler with a gamma of 0.8, effectively reducing the learning rate each epoch to refine convergence.

The models were evaluated using the cross-entropy loss across classes. We have replicated the 'Convolutional-KANs'[2] [24] and 'torch-conv-kan'[3] [31] implementations to evaluate their performance on the MNIST and CIFAR-10 datasets. Our replication efforts note the following. The 'Convolutional-KANs' implementation demonstrated adaptation to Convolutional KANs. The 'torch-conv-kan' implementation provided framework for applying KANs in convolutional layers.

B. Datasets

In this version, the models are evaluated on two datasets widely recognized in the machine learning community for evaluating image recognition models.

- **MNIST Dataset:** This dataset includes 70,000 handwritten digits, divided into a training set of 60,000 images and a test set of 10,000 images. Each image is grayscale and has a resolution of 28x28 pixels.
- **CIFAR-10 Dataset:** Comprising 60,000 color images across 10 classes, with each class representing a different type of object (e.g., cars, birds), this dataset is split into a training set of 50,000 images and a test set of 10,000 images. Each image is a three-channel RGB color with 32x32 pixels, offering a more challenging task due to its color content and the greater complexity of the images compared to MNIST.
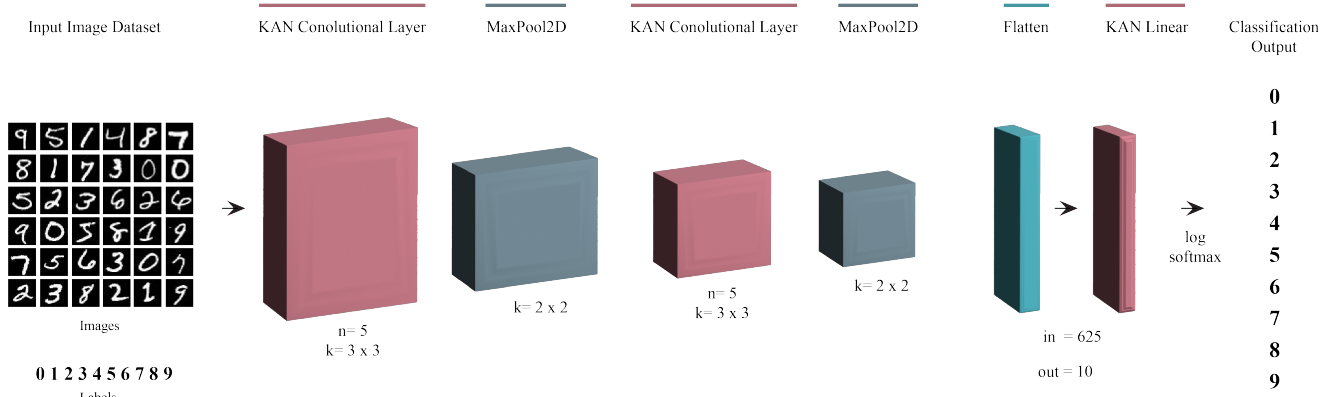
C. Model Configurations

In our experiments, we implemented several configurations of KANs, ConvsKANs, WavKAN, MLPs and traditional ConvNets to assess performance.

- **Baseline Models:** We used traditional MLPs and CNNs as baselines to establish a reference for evaluating the KANs and ConvKANs.
- **KANs with Traditional Layers:** We explored combinations of KAN layers with traditional convolutional and MLP layers to assess performance across different network architectures. Table I provides the configurations for our KANLinear layers, which are designed to test the flexibility and efficiency of KANs in processing linear layers with standard parameters. Table III provides compositional details of how we have integrated KAN layers with traditional neural network layers to test the hybrid models.
- **KANConv Implementations:** Baseline configurations of KANConv layers were tested to examine their effects on convolutional processing, particularly for feature extraction and classification accuracy. Table III outlines high-level composition of our KConvKAN models.
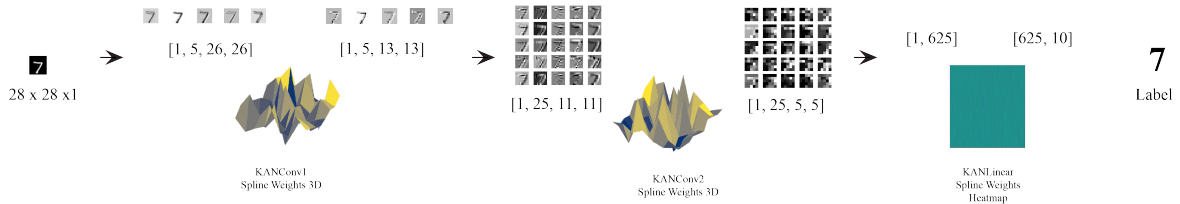
In this study, we evaluate numerous neural network architectures incorporating KAN principles. These configurations are designed to assess their utility without any bias toward specific model enhancements.

(a) Architectural Overview of Kolmogrov-Arnold Convolutional Network (KConvKAN)

(b) Visualization of Feature Maps and Spline Weights:

Fig. 3. **(a)** Architectural overview of a KConvKAN used in our experiments to classify on MNIST dataset. **(b)** Visualization of feature maps and spline weights for the corresponding layer in **(a)**.

TABLE I
KANLINEAR LAYER DETAILS

Model Type	In/Out Features	Grid Size	Spline Order	Scale Noise	Activation Function	Grid Range
KAN Linear	625/10	5	3	0.1	SiLU	[-1, 1]

TABLE II
CONVOLUTIONAL KAN (CONVKAN) LAYER DETAILS

Model Type	Grid Size	Spline Order	Scale Noise	Activation Function	Kernel Size/Stride
KANConv Layer	5	3	0.1	SiLU	(3,3)/(1,1)

D. Results

This section presents the performance evaluation of previously introduced network models trained on the MNIST and CIFAR-10 datasets. The evaluation focuses on key metrics such as accuracy, precision, recall, and F1 score. In addition, comparisons of parameters and evaluation time are presented. The comparative analysis of model performance on the MNIST and CIFAR-10 datasets reveals significant variations across different architectures, as detailed in Table IV. Models demonstrate high precision, recall, and F1 scores on MNIST. In contrast, performances on CIFAR-10 are considerably lower, underscoring the complexity of the dataset. Table V present the evaluation in terms of accuracy values and the parameters of each individual mode. The results indicate that models with increased parameter counts generally show higher accuracy on both datasets. This pattern suggests that the complexity of the models may play a significant role in handling the challenges presented by the respective datasets.

The KAN architectures, such as KConvKAN and ConvKANLinear, evaluated on MNIST and CIFAR-10, as highlighted in Table V, showcase varied effectiveness. For MNIST, KConvKAN (2 Layers) achieves an accuracy of 98.8%, which is competitive but not the highest. However, KConvKAN with 8 layers achieves 99.6% results along WavKAN (8 layers). On CIFAR-10, the performance increases with more specialized KAN models, indicating a nuanced benefit in handling more complex image tasks. While these KAN-enhanced models do not always lead in performance metrics, they maintain

Model Type	Layer Types	Activation Functions	KAN Layers
ConvKANLinear	Convolutional Layers & KAN Linear Layer	Multiple	KANLinear
KConvLinear	KConv Layers & Fully Connected Layers	Multiple	KConvs
KConvKAN	KConv Layers & KAN Linear Layers	SiLU	KConvs, KANLinear

Model	MNIST (%)				CIFAR-10 (%)			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
SimpleMLP	92.4	92.4	92.3	92.3	39.1	39.5	39.4	39.1
ConvNet (Small)	98.4	98.4	92.3	92.3	56.2	55.9	56.2	55.7
ConvNet (Medium)	99.1	99.1	99.1	99.1	64.2	64.4	64.2	64.2
ConvNet (Large)	99.4	99.4	99.4	99.4	71.0	71.3	71.0	70.8
ConvKANLinear	98.5	98.5	98.5	98.5	61.6	61.5	61.6	61.4
KConvLinear	98.3	98.3	98.3	98.3	59.3	59.3	59.3	59.2
KConvKAN (2 Layers)	98.8	98.8	98.8	98.8	62.6	62.4	62.6	62.3
KConvKAN (8 Layers)	99.6	99.6	99.5	99.6	78.8	78.6	78.5	78.6
WavKan (2 Layers)	98.8	98.8	98.7	98.8	64.4	64.4	66.3	66.1
WavKan (8 Layers)	99.6	99.6	99.5	99.6	79.7	79.4	79.5	79.4

consistently high scores across the board.

In Fig. 4, the relationship between the number of parameters in each model and the achieved accuracy on both datasets is explored. The graphical representation uses circle sizes to denote parameter count and color intensity to reflect accuracy, offering a visual correlation between model complexity and performance. The KAN architectures, positioned between the simplest and most complex models, illustrate a balanced approach to leveraging model design for effective learning. This balance might hint at potential efficiencies in parameter usage while achieving competitive accuracies, especially noticeable in the CIFAR-10 dataset where higher model complexities generally correlate with better performance.

E. Claims Validation

Kolmogorov-Arnold networks [1] are proposed as alternates to MLPs, and by extension, to other architectures like CNNs. Hence, their claims regarding accuracy, interpretability, scalability, and computational efficiency require validation for computer vision applications. In this section, we briefly discuss these aspect while comparing KANs and MLPs.

1) Claim-1: Accuracy: The original work on KANs [1] identifies that smaller KAN models can match or exceed the accuracy of larger MLPs in function fitting tasks. We can analyze this claim through our experiments on MNIST and CIFAR10 datasets. Our preliminary results show that KANs show competitive results with ConvNets of similar size.

However, their advantage seems to diminish as task complexity increases.

2) Claim-2: Computational Efficiency: Despite initial concerns about the potential computational overhead of implementing spline functions at scale, our implementations of KANs on standard hardware demonstrated a manageable increase in computational demands compared to MLPs. This is consistent with the original claim [1] that KANs can achieve comparable performance with potentially lower parameter counts.

3) Claim-3: Scalability: According to the KAN paper, these networks exhibit faster neural scaling laws than MLPs. However, when scaling up the networks to accommodate the high-dimensional data from CIFAR10, KANs required significant tuning of spline parameters to maintain performance, which adds to the training complexity. This observation leads to a nuanced view of the scalability claim, suggesting that practical scalability is contingent on the ability to manage increasing model complexity efficiently. The scalability of KANs to larger datasets, e.g., ImageNet, will be considered in the later versions.

IV. CONCLUSION

This work critically evaluates the efficacy of Kolmogorov-Arnold networks in the basic computer vision task of image recognition, contrasting their performance with conventional networks across MNIST and CIFAR-10 datasets. This work is a continuous effort that will be updated as more data and
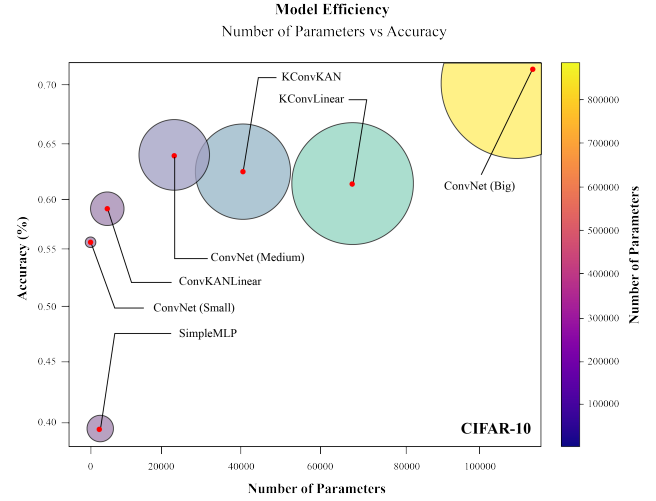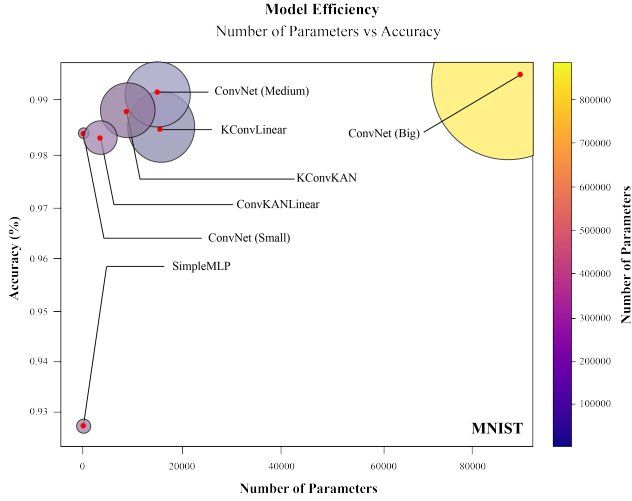
Fig. 4. Comparative performance visualization of different models on MNIST and CIFAR-10 datasets. The sizes and colors of the circles represent the scale of model parameters and performance metrics respectively, showcasing a range of outcomes from simple to complex models.

TABLE V
MODEL PERFORMANCE AND PARAMETERS

Dataset	Model	Accuracy (%)	Parameters	Epochs
MNIST	SimpleMLP	92.4	7,850	10
	ConvNet (Small)	98.4	2,740	
	ConvNet (Medium)	99.1	157,030	
	ConvNet (Large)	99.4	887,530	
	ConvKANLinear	98.5	163,726	
	KConvLinear	98.3	37,030	
	KConvKAN	98.8	94,650	
MNIST	KConvKAN-8	99.6	40,694,018	150
	WavKAN-2	98.8	951,370	
	WavKAN-8	99.6	10731562	
CIFAR-10	SimpleMLP	39.5	30.730	10
	ConvNet (Small)	56.2	3,580	
	ConvNet (Medium)	64.2	224,495	
	ConNet (Large)	71.0	1,134,890	
	ConvKANLinear	61.6	694,926	
	KConvLinear	59.3	48,370	
	KConvKAN	62.6	405,900	
CIFAR-10	KConvKAN-8	78.8	40,695,584	150
	WavKAN-2	64.4	952,650	
	WavKAN-8	79.7	10,732,202	

The variation in epochs across models is intended to investigate the effect of more extensive training durations on performance of KAN based models.

results are available. Our current findings reveal that while KANs do offer variation in the traditional neural network architectures, their superior accuracy and scalability are yet to be clearly evident for even slightly more complex vision datasets, e.g., CIFAR-10. The distinctive architecture of KANs, which integrates learnable spline functions directly onto the edges, provides an innovative approach to neural network design and opens new avenues for enhancing model interpretability and efficiency. The practical application of KANs in complex visual tasks require further optimization

TABLE VI
EVALUATION TIME IN COMPARISON WITH ACCURACIES

Model	Dataset	Accuracy	Train Time	Eval. Time
KConvKAN-8	MNIST	99.6%	1hr-55min	4.75 sec
WavKAN-2		98.8%	3hr-33min	5.72 sec
WavKAN-8		99.6%	8hr-40min	13.76 sec
KConvKAN-8	CIFAR-10	78.8%	2hr-05min	5.20 sec
WavKAN-2		64.4%	3hr-45min	7.81 sec
WavKAN-8		79.7%	8hr-55min	12.92 sec

The batch size used for WavKAN and WavKAN8 is 64, and for KConvKAN-8 it is 128, consistent across both datasets.

and exploration to fully harness their theoretical advantages. The research community is already engaged in leveraging KANs for vision or related tasks. Future work should focus on improving KAN-based architecture for scalability and generalization in diverse application scenarios within computer vision. Overall, whereas our investigation is yet to establish a clear performance advantage of KANs in vision domain, it does support the argument that investing in KAN-based vision models is worthwhile. With the large interest of vision and non-vision community in KANs, we can expect to see interesting developments in KANs for computer vision.

REFERENCES

[1] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov-arnold networks," *arXiv preprint arXiv:2404.19756*, 2024.
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[3] S. Haykin, "Neural networks and learning machines," 2009.
[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.
[5] P. S. Geidarov, "Clearly defined architectures of neural networks and multilayer perceptron," in *Optical Memory and Neural Networks*, vol. 26, no. 1. Allerton Press, Inc., 2017, pp. 62–76.
[6] Q. B. Diep, H. Y. Phan, and T.-C. Truong, "Crossmixed convolutional neural network for digital speech recognition," *PeerJ Computer Science*, vol. 9, p. e1051, 2023.

[7] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[8] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[9] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a non-polynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[10] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[18] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, 2015.

[21] C. J. Vaca Rubio, "Kolmogorov-arnold networks (kans) for time series analysis," *arXiv preprint arXiv:2405.08790*, 2024.

[22] I. de Gregorio, "Kolmogorov-arnold networks: the latest advance in neural networks," *Towards Data Science*, 2024.

[23] J. Schmidt-Hieber, "The kolmogorov–arnold representation theorem revisited," *Neural Networks*, vol. 139, pp. 1–15, 2021.

[24] A. Tepsich, "Convolutional-kans," 2024. [Online]. Available: https://github.com/AntonioTepsich/Convolutional-KANs

[25] C. Ziwen, "Vision-kan," 2024. [Online]. Available: https://github.com/chenziwenhaoshuai/Vision-KAN

[26] A. N. Ganesh, "Kan-gpt," 2024. [Online]. Available: https://github.com/AdityaNG/kan-gpt

[27] CG80499, "Kan-gpt-2," 2024. [Online]. Available: https://github.com/CG80499/KAN-GPT-2

[28] D. Qu and Q. Chen, "Kanerf," 2024. [Online]. Available: https://github.com/Tavish9/KANeRF

[29] J. Talibi, "Kanu_net," 2024. [Online]. Available: https://github.com/JaouadT/KANU_Net

[30] A. Dash, "kansformers," 2024. [Online]. Available: https://github.com/akaashdash/kansformers

[31] I. Drokin, "torch-conv-kan," 2024. [Online]. Available: https://github.com/IvanDrokin/torch-conv-kan

[32] A. N. Kolmogorov, "On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables," *Proceedings of the USSR Academy of Sciences*, vol. 108, pp. 179–182, 1956, english translation: Amer. Math. Soc. Transl., 17 (1961), pp. 369–373.

[33] R. X. Gao and R. Yan, *Discrete Wavelet Transform*. Boston, MA: Springer US, 2011, pp. 49–68.

[34] ——, *Continuous Wavelet Transform*. Boston, MA: Springer US, 2011, pp. 33–48.

[35] Z. Bozorgasl and H. Chen, "Wav-kan: Wavelet kolmogorov-arnold networks," 2024.