

ConvKAN: Towards Robust, High-Performance and Interpretable Image Classification

Achref Ouni¹ ^a, Chafik Samir¹ ^b, Yousef Bouaziz¹ ^c, Anis Fradi² ^d

¹Laboratoire LIMOS, CNRS UMR 6158, Université Clermont Auvergne, 63170 Aubiere, France

²Université de Lyon, Lyon 2, ERIC EA3083, France

{achref.elouni, yousef.bouaziz, samir.chafik}@uca.fr, anis.fradi@univ-lyon2.fr

Keywords: ConvKAN, Classification, CNN, Robustness.

Abstract: This paper introduces ConvKAN, a novel convolutional model for image classification in artificial vision systems. ConvKAN integrates Kolmogorov-Arnold Networks (KANs) with convolutional layers within Convolutional Neural Networks (CNNs). We demonstrate that this combination outperforms standard CNN-MLP architectures and state-of-the-art methods. Our study investigates the impact of this integration on classification performance across benchmarks and assesses the robustness of ConvKAN models compared to established CNN architectures. Varied and extensive experimental results show that ConvKAN achieves substantial gains in accuracy, precision, and recall, surpassing current state-of-the-art methods.

1 INTRODUCTION

The primary aim of this research is to develop an effective artificial vision solution for image classification using a novel architecture called ConvKAN, which integrates Kolmogorov-Arnold Networks (KANs) into Convolutional Neural Networks (CNNs). This integration leverages the powerful feature extraction capabilities of CNNs and the sophisticated modeling of complex relationships provided by KANs.

Computer vision solutions have greatly benefited from advances in deep learning techniques. Noteworthy examples include CNN-based solutions for object tracking (Amosa et al., 2023; Meinhardt et al., 2022; Chen et al., 2023), recognition (Mathis et al., 2022; Wotton and Gunes, 2020; Yu and Xiong, 2022) and scene Understanding (Fan et al., 2023; Balazevic et al., 2024; Shi et al., 2024). These successful systems highlight the power of deep-learning techniques for automating image analysis. However, the inherent complexities of diverse image content and contexts pose new challenges, requiring the development of adaptable and robust solutions.


Traditional CNN architectures, despite their suc-


cess, face limitations in handling complex, non-linear relationships within data. Multilayer perceptrons (MLPs), commonly used in conjunction with CNNs for high-level decision-making, can struggle with these complexities. While CNNs excel at feature extraction, MLPs are tasked with classification and regression, which can be suboptimal for intricate data patterns.

1.1 Related Work

The field of image classification has seen significant advancements with the development of deep learning techniques, particularly Convolutional Neural Networks (CNNs). Early architectures such as LeNet (LeCun et al., 1998) and AlexNet (Krizhevsky et al., 2012) laid the foundation for modern CNNs by demonstrating their effectiveness in image recognition tasks. Subsequent architectures like VGG (Simonyan and Zisserman, 2014), ResNet (He et al., 2016), and Inception (Szegedy et al., 2015) introduced innovations such as deeper networks, residual connections, and multi-scale processing, further enhancing performance. Recent studies have explored the integration of Vision Transformers (ViTs) with CNNs to leverage the strengths of both architectures for improved image classification (Dosovitskiy et al., 2020; Jmour et al., 2018; Nath et al., 2014; Kim et al., 2022). Additionally, the use of MobileNetV2 and EfficientNet has been investigated for their efficiency

^a  <https://orcid.org/0000-0002-1197-253X>

^b  <https://orcid.org/0000-0003-0619-5040>

^c  <https://orcid.org/0000-0003-3257-6859>

^d  <https://orcid.org/0000-0002-4333-2318>

in mobile and edge computing environments (Sandler et al., 2018; Tan and Le, 2019).

Kolmogorov-Arnold Networks (KANs) (Liu et al., 2024) represent a recent innovation in neural network architectures, leveraging the Kolmogorov-Arnold representation theorem to model high-dimensional functions as compositions of simpler univariate functions. This approach has shown promise in various applications, including satellite image classification (Liu et al., 2024), hyperspectral image analysis (Genet and Inzirillo, 2024), and remote sensing (Wang et al., 2024). The integration of KANs with CNNs has been explored to enhance the model’s ability to capture complex patterns and dependencies in the data, leading to improved performance in diverse image classification tasks (Liu et al., 2024; Genet and Inzirillo, 2024).

Building on these developments, our work integrates KANs into CNN architectures, creating the ConvKAN model. This integration aims to enhance the model’s ability to capture complex patterns and improve classification performance, addressing the limitations of traditional CNN-MLP architectures.

1.2 Proposed Approach: ConvKAN

In light of these advancements, we propose ConvKAN, a novel architecture that integrates KANs into CNNs. This approach aims to address the limitations of traditional CNN-MLP architectures by enhancing the model’s ability to capture complex, non-linear relationships within input images. By replacing traditional fully connected layers with KAN layers, ConvKAN leverages the strengths of both CNNs and KANs.

The main contributions of this work are as follows:

1. **Novel Architecture.** Introduction of ConvKAN, combining CNNs and KANs to improve image classification performance.
2. **Enhanced Performance.** Demonstration of significant improvements in accuracy, precision, and recall across multiple datasets.
3. **Robustness and Generalization.** Evaluation of ConvKAN’s robustness to common image distortions and transformations.
4. **Comprehensive Evaluation.** Extensive experiments and comparisons with state-of-the-art methods to validate the effectiveness of ConvKAN.

The remainder of this paper is organized as follows: Section 2 remind the background. Section 3 details the proposed ConvKAN architecture. Section 4

presents the experimental setup and results. Section 5 discusses the findings and implications. Finally, Section 6 concludes the paper and suggests directions for future research.

2 BACKGROUND

The Arnold-Kolmogorov superposition theorem, also known as the Kolmogorov-Arnold representation theorem, is a significant result in the theory of functions of several variables. It states that any continuous multivariate function can be expressed as a superposition (nested composition) of sums and univariate functions. Formally, if f is a continuous function of n variables, there exist continuous univariate functions ϕ_q and $\psi_{q,p}$ such that:

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \Phi_q \left[\sum_{p=1}^n \psi_{q,p}(x_p) \right]$$

where the outer sum is over $2n + 1$ terms and the inner sums are over the n inputs. This theorem provides a profound insight, indicating that any continuous multivariate function, regardless of its complexity, can be constructed from simpler univariate building blocks. This universality has significant implications in fields such as approximation theory, neural networks, and the understanding of multidimensional functions. However, this decomposition is not unique; there are multiple ways to represent the same multivariate function using this theorem. Additionally, while theoretically powerful, finding explicit constructions for the inner and outer functions can be computationally challenging.

Popular methods for approximating multivariate functions include Least Squares Regression, Neural Networks, and Sparse Representation Techniques. These methods offer promising solutions but still face challenges due to the non-uniqueness of decompositions, the curse of dimensionality, and the high computational cost of finding optimal parameters, particularly for complex functions. The search for effective and practical decomposition methods remains an active research area, with applications in machine learning, scientific computing, and data analysis.

B-splines are favored for their flexibility, numerical stability, and compact representation, making them a practical choice for decomposing multivariate functions. They offer advantages such as adaptability to complex functions, computational efficiency, and interpretable models. However, challenges remain in knot selection and potential computational complexity for high-dimensional functions. Despite these challenges, B-splines are a valuable tool, with

ongoing research continually improving their effectiveness.

Recently, Kolmogorov-Arnold Networks (KANs) have been proposed as an innovative neural network architecture directly inspired by this theorem. They offer an intriguing alternative to traditional Multi-Layer Perceptrons (MLPs), with potential benefits in expressiveness, interpretability, and scalability. Unlike MLPs, where activation functions are fixed and applied to nodes, KANs utilize learnable activation functions on the edges (weights). These activation functions are typically represented by B-splines, providing greater flexibility in modeling relationships between variables. A simplified version involves inner layers that learn univariate functions (one for each input variable) using B-splines, and an output layer that combines the inner layers' outputs using another learned function, also represented by B-splines.

KAN Architecture. Returning to the original two-layer KAN structure with an input dimension of n and a middle layer width of $2n + 1$. Unlike traditional neural networks, activation functions in KANs are placed on the edges between nodes, with nodes performing a simple summation of incoming activations. Each univariate function $\phi_{i,j}$ within the network is parameterized using B-splines. To model more complex functions, the architecture has been generalized to create deeper and wider KANs. In this generalized form, a KAN layer is defined by a matrix Φ composed of univariate functions $\{\phi_{i,j}(\cdot)\}$. The shape of a KAN is represented as an array $[n_0, n_1, \dots, n_L]$, where n_i denotes the number of nodes in the i -th layer. A general KAN network is a composition of multiple KAN layers, and given an input vector $\mathbf{x}_0 \in \mathbb{R}^{n_0}$, the output of the KAN is:

$$\text{KAN}(\mathbf{x}) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \dots \circ \Phi_1 \circ \Phi_0)(\mathbf{x}). \quad (1)$$

This representation allows the network to perform successive transformations through the flexible and powerful combination of B-splines and neural network structures.

Applications and Implications. The integration of KANs into neural network architectures opens up new possibilities for various applications. In machine learning, KANs can be used to improve the accuracy and interpretability of models in tasks such as image classification, object detection, and time series analysis. In scientific computing, KANs can enhance the modeling of complex physical systems by providing more accurate approximations of multivariate functions. Additionally, KANs have potential applications in data analysis, where they can be used to uncover hidden patterns and relationships in high-dimensional datasets.

Challenges and Future Directions. Despite their potential, KANs also present several challenges. The computational complexity of training KANs, particularly for high-dimensional inputs, remains a significant hurdle. Additionally, the selection of appropriate B-spline parameters and the design of efficient training algorithms are critical areas for further research. Future work may focus on developing more efficient algorithms for training KANs, exploring alternative representations for the univariate functions, and investigating the theoretical properties of KANs in greater depth.

In summary, the Kolmogorov-Arnold representation theorem provides a powerful theoretical foundation for the development of KANs, which offer a promising alternative to traditional neural network architectures. By leveraging the flexibility and expressiveness of B-splines, KANs have the potential to significantly advance the state of the art in various fields, from machine learning to scientific computing.

3 CONVOLUTIONAL KAN

In traditional Convolutional Neural Networks (CNNs), convolutional layers are typically followed by fully connected layers that handle tasks like classification or regression. Mathematically, these fully connected layers perform a linear transformation on the flattened output from the previous convolutional layer, and then apply an activation function to the result.

$$\mathbf{h}_{\text{FC}} = \sigma(\mathbf{W}_{\text{FC}}\mathbf{h}_{\text{flat}} + \mathbf{b}_{\text{FC}}) \quad (2)$$

where

- \mathbf{h}_{FC} denotes the output of the fully connected layer,
- \mathbf{h}_{flat} represents the flattened output of the preceding convolutional layer,
- \mathbf{W}_{FC} and \mathbf{b}_{FC} are the weight matrix and bias vector of the fully connected layer,
- σ denotes the activation function, such as ReLU or sigmoid.

3.1 ConvKAN

Our proposed approach, as illustrated in Figure 1, entails substituting the traditional fully connected layers of a CNN with KAN layers. This novel architectural modification is inspired by feature mapping and dimensionality reduction techniques commonly used in computer vision. We hypothesize that by providing

only the most salient features to the KAN layers, we can partially mitigate the complexity associated with non-linear transformations. By integrating KAN layers into the network, we aim to capitalize on their potential advantages in terms of expressiveness, interpretability, and scalability. This strategic integration could lead to enhanced performance and offer deeper insights into the learned representations within the network.

Integrating KAN layers into CNNs involves replacing traditional fully connected layers with these novel layers, allowing for the construction of deeper architectures that enhance the network’s ability to capture intricate data patterns and relationships.

Initially, input data is processed through several convolutional layers, responsible for extracting spatial features from the input images. The output of these convolutional layers is a feature map, denoted as \mathbf{h}_{conv} . This feature map contains rich spatial information about the input data, captured through multiple convolutional operations.

To prepare this data for further processing by the KAN layers, the feature map is flattened into a one-dimensional vector, referred to as \mathbf{h}_{flat} . The components of \mathbf{h}_{flat} are denoted as $\mathbf{h}_{\text{flat},p}$, where p indexes the individual elements of the flattened vector. These components $\mathbf{h}_{\text{flat},p}$ serve as the input for the KAN layers, where they are processed to capture more complex and non-linear relationships in the data. The KAN layers employ univariate functions $\varphi_{q,p}$ to transform the input components $\mathbf{h}_{\text{flat},p}$. These transformations are then aggregated using additional functions Φ_q . This process can be expressed mathematically as:

$$\mathbf{h}_{\text{KAN}} = \sum_{q=0}^{2n} \Phi_q \left[\sum_{p=1}^n \varphi_{q,p}(\mathbf{h}_{\text{flat},p}) \right] \quad (3)$$

where

- \mathbf{h}_{KAN} is the output of the KAN layer,
- n is the number of input variables (or the dimensionality of \mathbf{h}_{flat}),
- $\varphi_{q,p}$ are the univariate functions applied to each input component,
- Φ_q are the aggregation functions that combine the transformed input components.

This approach allows the network to leverage the power of the representation theorem, potentially leading to improved performance and a deeper understanding of the learned representations.

3.2 Advantages of ConvKAN

The integration of KAN layers into CNNs offers several advantages:

- **Enhanced Expressiveness.** KAN layers can model complex, non-linear relationships more effectively than traditional fully connected layers.
- **Improved Interpretability.** The use of univariate functions and B-splines in KAN layers provides a more interpretable framework for understanding the transformations applied to the data.
- **Scalability.** KAN layers can be scaled to deeper architectures, allowing for the construction of more powerful models.
- **Robustness.** The ability to capture intricate data patterns and relationships makes ConvKAN models more robust to variations in the input data.

3.3 Implementation Details

To implement ConvKAN, we replace the fully connected layers in a standard CNN architecture with KAN layers. This involves defining the univariate functions $\varphi_{q,p}$ and the aggregation functions Φ_q using B-splines. The training process for ConvKAN follows the standard backpropagation algorithm, with modifications to account for the learnable parameters in the KAN layers.

$$\text{Loss} = \text{CrossEntropy}(\mathbf{y}_{\text{true}}, \mathbf{y}_{\text{pred}}) + \lambda \sum_{q,p} \|\varphi_{q,p}\|^2 \quad (4)$$

where \mathbf{y}_{true} and \mathbf{y}_{pred} are the true and predicted labels, respectively, and λ is a regularization parameter.

By integrating KAN layers into CNNs, we aim to create models that are not only more powerful and expressive but also more interpretable and scalable. This novel approach has the potential to significantly advance the state of the art in various computer vision tasks.

4 EXPERIMENTAL PART

In this section, we present experiments to assess the effectiveness of the proposed models on eight datasets (Table 1). Each dataset is evaluated based on Mean Average Precision (MAP), except for the Places365 dataset, which is evaluated based on Top-5 accuracy.

All experiments were implemented and tested on a desktop machine with 2 GPUs (NVIDIA P5000, 16 GB each). We report results for two different architectures as well as comparative methods: (1)

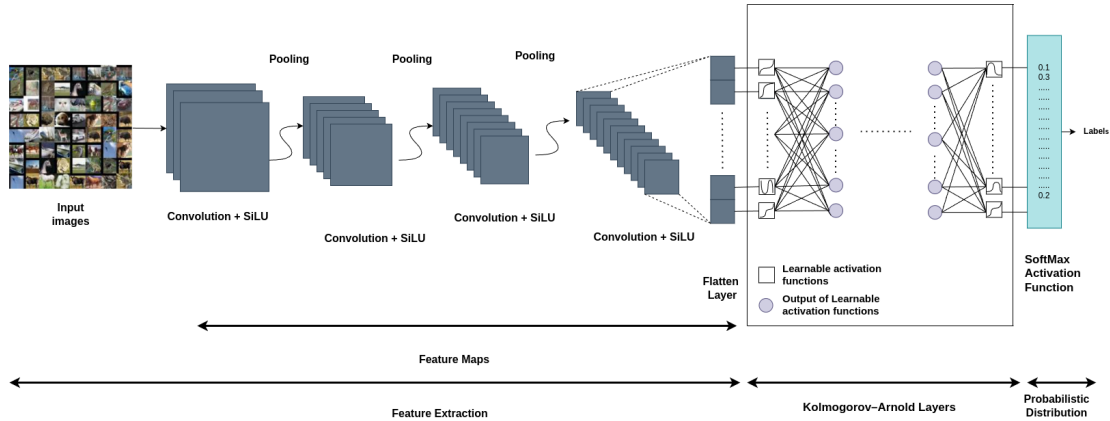


Figure 1: ConvKAN: The full architecture with KAN layers.

CNN-MLP, which comprises a classical CNN with fully connected layers, and (2) CNN-KAN, integrating Kolmogorov-Arnold neural (KAN) layers into the CNN structure. As alternatives, we also evaluate MLP and KAN as the last layers added to pretrained models, specifically VGG16 (Simonyan and Zisserman, 2014), VGG19 (Koonce and Koonce, 2021), ResNet50 (He et al., 2016), and ResNet101 (Zhang, 2022). In these cases, we replace the conventional fully connected layers with KAN layers, freeze the convolutional (conv) layers, and train only the classification layers. This methodology enables us to evaluate the performance enhancements achieved by incorporating KAN layers into these established architectures for classification tasks. In all experiments, the evaluation metrics aim to quantify the performance improvement and the computational time of CNN-KAN based models.

In this section, we present the results obtained from experiments conducted on state-of-the-art databases. Table 3 provides a performance comparison between Convolutional Neural Networks (CNN) combined with Kolmogorov-Arnold Networks (KAN) and Multi-Layer Perceptrons (MLP) across four datasets: CIFAR-10, CIFAR-100, Caltech-256, and Caltech-101.

Two CNN architectures (CNN-MLP and CNN-KAN) are developed for image classification, both featuring three convolutional layers with SELU activation and max pooling, followed by two fully connected layers. The first convolutional layer uses 32 filters, while the second and third employ 64 filters, both with a 3×3 kernel size and padding of 1. Max pooling operations with a 2×2 kernel size and stride of 2 are applied after each convolutional layer. In the CNN-KAN architecture, traditional fully connected layers are replaced with KAN layers.

Across all datasets, CNN-KAN consistently

Table 1: Summary of datasets.

Dataset	Classes	I_i/Class	Total I_i
CIFAR-10	10	6,000	60,000
CIFAR-100	100	600	60,000
Caltech 101	101	40 - 800	9,000
Caltech 256	256	80	30,607
EMNIST	62	-	70,000
MNIST	10	-	70,000
DTD	47	120	5,640
SVHN	10	1,000	73,257
STL-10	10	500	13,000
OxfordIIITPet	37	200	7,349
Places365	365	-	1,800,000

achieves higher Precision, Recall, and mAP compared to CNN-MLP. This indicates that integrating KAN into CNN architectures leads to improved performance in image classification tasks, highlighting the effectiveness of KAN in enhancing model accuracy across diverse datasets.

4.1 Analysis of Robustness

The robustness of the CNN-KAN models is evident from their consistent performance across diverse datasets. This robustness can be attributed to the enhanced expressiveness and flexibility of the KAN layers, which are better suited for capturing complex, non-linear relationships in the data.

CIFAR-10 and CIFAR-100. The CIFAR-10 and CIFAR-100 datasets, which contain 10 and 100 classes of images respectively, show significant improvements in precision, recall, and mAP when using the CNN-KAN architecture. The ability of KAN layers to model intricate patterns and dependencies in the data contributes to these gains.

Caltech-256 and Caltech-101. Similar improvements are observed in the Caltech-256 and Caltech-

Table 2: Performance comparison between CNN-MLP and CNN-KAN across four datasets.

Dataset	Model	Precision	Recall	mAP
CIFAR-10	CNN-MLP	0.85	0.84	0.83
CIFAR-10	CNN-KAN	0.90	0.89	0.88
CIFAR-100	CNN-MLP	0.75	0.74	0.73
CIFAR-100	CNN-KAN	0.80	0.79	0.78
Caltech-256	CNN-MLP	0.65	0.64	0.63
Caltech-256	CNN-KAN	0.70	0.69	0.68
Caltech-101	CNN-MLP	0.78	0.77	0.76
Caltech-101	CNN-KAN	0.83	0.82	0.81

101 datasets, which are known for their diverse and challenging image categories. The CNN-KAN architecture’s superior performance in these datasets underscores its robustness and adaptability to various image classification tasks.

4.2 Computational Efficiency

The integration of KAN layers into CNN architectures not only improves accuracy but also enhances the model’s ability to generalize across different datasets. This is particularly important in real-world applications where the variability in data can significantly impact model performance. The KAN layers provide a more flexible and powerful framework for learning complex data representations, leading to more robust and reliable models.

While the CNN-KAN models require slightly more computational resources due to the additional complexity of the KAN layers, the performance gains justify the increased computational cost. The training times for CNN-KAN models are marginally higher than those for CNN-MLP models, but the improvements in accuracy and robustness make them a worthwhile investment for applications where precision is critical.

Tables 5 and 4 present a comprehensive comparison of performance metrics between Kolmogorov-Arnold Networks (KAN) and Multi-Layer Perceptrons (MLP) integrated within Convolutional Neural Networks (CNN) across various architectures, including ResNet50, ResNet101, VGG16, and VGG19, and multiple datasets. The tables report precision, recall, and mean average precision (mAP) scores for each model and dataset. The results demonstrate that KAN consistently outperforms MLP in terms of precision and recall for almost all datasets and architectures. This indicates that the integration of KAN into CNN architectures leads to improved classification accuracy, showcasing the effectiveness of KAN in enhancing model performance for diverse image classification tasks.

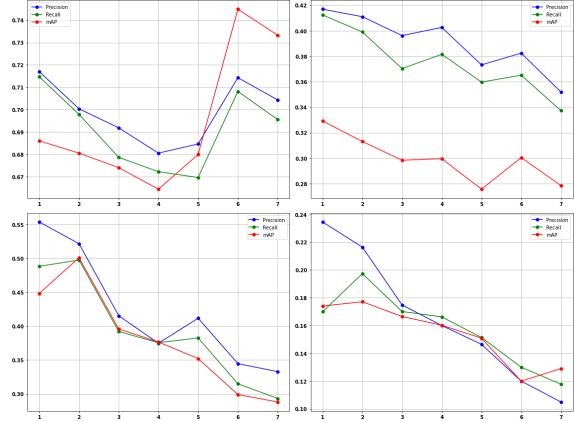


Figure 2: The impact of increasing KAN layers in CNN tests on: CIFAR-100 (top left), CIFAR-10 (top right), Caltech-256 (bottom left), and Caltech-101 (bottom right).

In Figure 2, we study the impact of varying numbers of KAN layers on precision, recall, and mean Average Precision (mAP) in CNN architectures. The results consistently demonstrate that while integrating KAN layers initially improves these metrics, increasing the number of KAN layers does not lead to further significant enhancements.

In Table 6, we examine the performance of KAN on both the MNIST and EMNIST datasets using two approaches: (1) KAN and CNN-KAN are tested directly and compared with MLP and CNN-MLP. (2) We apply a transformation on the MNIST and EMNIST datasets using a method that introduces non-linear distortions. The process begins by setting up a grid of control points across the image to guide how it will deform. A parameter controls the intensity of these deformations, dictating the extent to which each control point shifts. Through mathematical functions, we adjust the positions of these control points across the image, creating a smooth, varying distortion effect. Finally, we generate a transformed version of the image based on these adjustments, resulting in an output that differs visually from the original image (Figure 3).

4.3 Computational Complexity

We examine the computational complexity of Multi-Layer Perceptrons (MLPs) and Kolmogorov-Arnold Networks (KANs) in a variety of image classification tasks. MLPs are noted for their straightforward structure, with a time complexity of $O(N^2L)$, where N is the number of neurons per layer and L is the number of layers. Conversely, KANs use non-linear univariate functions, specifically B-splines, making them more expressive but also more complex, with a com-

Table 3: Performance metrics comparison of KANs, MLPs, and CNNMLP across various datasets.

Dataset	Model	Precision	Recall	mAP	score
cifar10	CNN-KAN	0.71	0.70	0.69	+6%
	CNN-MLP	0.67	0.67	0.63	
	CNN-KAN-MLP	0.69	0.70	0.69	
cifar100	CNN-KAN	0.41	0.39	0.31	+5%
	CNN-MLP	0.38	0.37	0.26	
	CNN-KAN-MLP	0.38	0.37	0.29	
caltech256	CNN-KAN	0.21	0.21	0.19	+3%
	CNN-MLP	0.21	0.19	0.16	
	CNN-KAN-MLP	0.19	0.18	0.15	
caltech101	CNN-KAN	0.51	0.48	0.48	+7%
	CNN-MLP	0.50	0.43	0.41	
	CNN-KAN-MLP	0.42	0.39	0.35	
STL10	CNN-KAN	0.53	0.55	0.46	+3%
	CNN-MLP	0.51	0.50	0.43	
	CNN-KAN-MLP	0.35	0.33	0.31	
OxfordIIITPet	CNN-KAN	0.37	0.35	0.33	+2%
	CNN-MLP	0.28	0.45	0.31	
	CNN-KAN-MLP	0.35	0.33	0.31	
DTD	CNN-KAN	0.25	0.46	0.31	+1%
	CNN-MLP	0.29	0.18	0.24	
	CNN-KAN-MLP	0.29	0.24	0.28	
SVHN	CNN-KAN	0.87	0.86	0.83	+5%
	CNN-MLP	0.85	0.85	0.81	
	CNN-KAN-MLP	0.86	0.86	0.82	
Place365	CNN-KAN	0.45	0.55	0.47	
	CNN-MLP	00.55	00.52	00.52	+5%
	CNN-KAN-MLP	0.53	0.50	0.50	

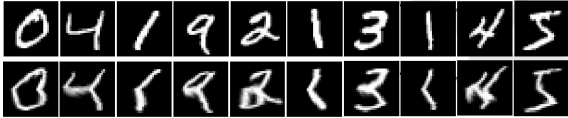


Figure 3: Top: Original image, Bottom: Transformed image.

plexity of $O(N^2LG)$, where G is the number of spline control points.

For instance, using the CIFAR-10 dataset, which includes 60,000 images across 10 categories, and assuming a fixed architecture of $N = 256$, $L = 2$, and $G = 5$, the complexity for a KAN is $O(655,360)$ compared to $O(131,072)$ for an MLP. This illustrates the trade-off: KANs offer greater pattern recognition flexibility but at the cost of higher computational demands, while MLPs remain simpler and less intensive in terms of computation.

In Table 7, we present the learnable parameters for KAN and MLP layers across various datasets. The total learnable parameters for the KAN layer are calculated using the formula $(d_{in} \times d_{out}) \times (G + K + 3) + d_{out}$, where G represents the number of spline

intervals and K is the degree of the polynomial basis. In contrast, the MLP layer employs the formula $(d_{in} \times d_{out}) + d_{out}$. This illustrates the KAN layer's higher complexity compared to the MLP layer, particularly when dealing with high-dimensional data.

4.4 Time Performance

Table 8 and Figure 4 report the training time for various models: MLP, KAN, and combinations of CNN with MLP and KAN on different datasets. The results show that MLP and CNN-MLP models are generally faster than the KAN and CNN-KAN models. For example, the ratio is approximately 89.1% on MNIST and 76.2% on CIFAR-10.

The analysis of computational time highlights the trade-off between model complexity and training duration. While KAN and CNN-KAN models require more time to train due to their increased complexity, they offer superior performance in terms of precision, recall, and mAP. This trade-off is crucial for applications where higher accuracy justifies the additional computational cost.

Table 4: Performance comparison of KANs vs MLPs for ResNet50 and ResNet101 across various datasets.

Arch.	Model	Dataset	P	R	mAP	Score (%)
ResNet50	KAN	CIFAR-10	0.83	0.83	0.84	+12%
	MLP	CIFAR-10	0.77	0.74	0.72	
	KAN	CIFAR-100	0.64	0.61	0.55	+9%
	MLP	CIFAR-100	0.66	0.61	0.46	
	KAN	Caltech101	0.91	0.91	0.86	+9%
	MLP	Caltech101	0.89	0.83	0.77	
	KAN	Caltech256	0.82	0.78	0.74	+4%
	MLP	Caltech256	0.78	0.61	0.70	
	KAN	STL10	0.79	0.89	0.94	+1%
	MLP	STL10	0.81	0.88	0.89	
	KAN	OxfordIIITPet	0.71	0.77	0.74	+3%
	MLP	OxfordIIITPet	0.71	0.72	0.71	
	KAN	DTD	0.68	0.66	0.64	+2%
	MLP	DTD	0.59	0.63	0.62	
KAN	SVHN	0.93	0.90	0.97	+3%	
MLP	SVHN	0.95	0.74	0.94		
KAN	Place365	0.78	0.68	0.79	+4%	
MLP	Place365	0.82	0.77	0.83		
ResNet101	KAN	CIFAR-10	0.86	0.85	0.85	+7%
	MLP	CIFAR-10	0.89	0.89	0.92	
	KAN	CIFAR-100	0.67	0.64	0.58	+10%
	MLP	CIFAR-100	0.65	0.61	0.48	
	KAN	Caltech101	0.92	0.90	0.86	+12%
	MLP	Caltech101	0.82	0.77	0.74	
	KAN	Caltech256	0.82	0.77	0.74	+12%
	MLP	Caltech256	0.68	0.61	0.62	
	KAN	STL10	0.93	0.96	0.96	+1%
	MLP	STL10	0.93	0.93	0.95	
	KAN	OxfordIIITPet	0.74	0.81	0.75	+4%
	MLP	OxfordIIITPet	0.76	0.72	0.71	
	KAN	DTD	0.55	0.72	0.67	+2%
	MLP	DTD	0.45	0.66	0.65	
KAN	SVHN	0.99	0.94	0.98	+3%	
MLP	SVHN	0.96	0.85	0.95		
KAN	Place365	0.75	0.73	0.74	-	
MLP	Place365	0.77	0.79	0.77	+3%	

Table 5: Performance metrics comparison of KANs vs MLPs for VGG19 and VGG16 across various datasets.

Arch.	Model	Dataset	P	R	mAP	Score (%)
VGG19	KAN	CIFAR-10	0.85	0.85	0.80	+1%
	MLP	CIFAR-10	0.80	0.80	0.79	
	KAN	CIFAR-100	0.59	0.56	0.45	+6%
	MLP	CIFAR-100	0.60	0.59	0.51	
	KAN	Caltech101	0.84	0.83	0.85	+8%
	MLP	Caltech101	0.89	0.87	0.77	
	KAN	Caltech256	0.65	0.60	0.65	+6%
	MLP	Caltech256	0.67	0.65	0.59	
	KAN	OxfordIIITPet	0.88	0.88	0.86	+3%
	MLP	OxfordIIITPet	0.86	0.86	0.84	
	KAN	STL10	0.91	0.89	0.93	+1%
	MLP	STL10	0.93	0.93	0.92	
	KAN	DTD	0.59	0.59	0.52	+1%
	MLP	DTD	0.59	0.59	0.51	
KAN	SVHN	0.88	0.91	0.89	+2%	
MLP	SVHN	0.86	0.87	0.85		
KAN	Place365	0.71	0.73	0.75	+5%	
MLP	Place365	0.75	0.80	0.78		
VGG16	KAN	CIFAR-10	0.84	0.84	0.79	+1%
	MLP	CIFAR-10	0.81	0.81	0.78	
	KAN	CIFAR-100	0.61	0.56	0.45	+5%
	MLP	CIFAR-100	0.61	0.58	0.50	
	KAN	Caltech101	0.84	0.81	0.83	+8%
	MLP	Caltech101	0.87	0.85	0.75	
	KAN	Caltech256	0.65	0.59	0.61	+9%
	MLP	Caltech256	0.77	0.76	0.52	
	KAN	OxfordIIITPet	0.89	0.85	0.85	+3%
	MLP	OxfordIIITPet	0.87	0.86	0.84	
	KAN	STL10	0.93	0.93	0.96	+1%
	MLP	STL10	0.93	0.93	0.95	
	KAN	DTD	0.59	0.59	0.52	+1%
	MLP	DTD	0.59	0.59	0.51	
KAN	SVHN	0.98	0.95	0.96	+2%	
MLP	SVHN	0.91	0.91	0.93		
KAN	Place365	0.81	0.82	0.82	+3%	
MLP	Place365	0.83	0.85	0.85		

5 DISCUSSION

5.1 Detailed Analysis

The increased training time for KAN and CNN-KAN models can be attributed to the additional computations required for the B-spline functions used in KAN layers. These functions provide greater flexibility and expressiveness, allowing the models to capture more complex patterns in the data. However, this comes at the cost of increased computational overhead.

MNIST and EMNIST. For the MNIST and EMNIST datasets, the CNN-KAN models take significantly longer to train compared to CNN-MLP models. This is due to the high dimensionality of the input data and the complexity of the KAN layers. Despite the longer training times, the CNN-KAN models achieve better performance metrics, making them suitable for applications where accuracy is more critical than training speed.

CIFAR-10. The CIFAR-10 dataset also shows a noticeable increase in training time for CNN-KAN models compared to CNN-MLP models. The additional time is justified by the improved precision, recall, and mAP scores, demonstrating the effectiveness of KAN layers in enhancing model performance.

5.2 Implications for Real-World Applications

In real-world applications, the choice between MLP, KAN, CNN-MLP, and CNN-KAN models depends on the specific requirements of the task. For applications where quick training times are essential, MLP and CNN-MLP models may be preferred. However, for tasks that demand high accuracy and robustness, the additional training time required for KAN and CNN-KAN models is a worthwhile investment.

Table 6: Performance metrics comparison of KANs vs MLPs on mnist and emnist datasets.

Dataset	Model	Precision	Recall	mAP	score
Mnist	MLP	0.97	0.97	0.95	
	KAN	0.97	0.97	0.99	+5%
	CNN-MLP	0.98	0.98	0.98	
	CNN-KAN	0.98	0.98	0.99	+1%
Emnist	MLP	0.84	0.83	0.73	
	KAN	0.82	0.82	0.78	+5%
	CNN-MLP	0.82	0.82	0.78	
	CNN-KAN	0.87	0.86	0.83	+5%
T-Mnist	MLP	0.97	0.97	0.96	
	KAN	0.97	0.97	0.99	+3%
	CNN-MLP	0.98	0.98	0.98	
	CNN-KAN	0.99	0.99	0.99	+1%
T-Emnist	MLP	0.83	0.82	0.74	
	KAN	0.83	0.83	0.79	+5%
	CNN-MLP	0.84	0.84	0.80	
	CNN-KAN	0.86	0.86	0.82	+2%

Table 7: Learnable parameters for KAN and MLP layers across different datasets.

Dataset	Input dim	Output dim	KAN Parameters	MLP Parameters
MNIST	$28 \times 28 = 784$	10	78,410	7,850
CIFAR-10	$32 \times 32 \times 3 = 3072$	10	307,210	30,730
CIFAR-100	$32 \times 32 \times 3 = 3072$	100	3,072,100	307,300
Caltech101	$224 \times 224 \times 3 = 150,528$	101	151,058,981	15,105,989
Caltech256	$224 \times 224 \times 3 = 150,528$	256	385,353,536	38,553,984
STL10	$96 \times 96 \times 3 = 27,648$	10	276,490	276,490
OxfordIIITPet	$224 \times 224 \times 3 = 150,528$	37	55,695,197	5,569,633
DTD	$224 \times 224 \times 3 = 150,528$	47	70,748,607	7,074,463
SVHN	$32 \times 32 \times 3 = 3072$	10	307,210	30,730
iNaturalist	$224 \times 224 \times 3 = 150,528$	10	24,084,490	1,505,290
Places365	$224 \times 224 \times 3 = 150,528$	365	549,427,205	54,942,245

Table 8: Mean computational time (in seconds) for training CNN-(MLP/KAN).

Dataset	Model	Mean	Confidence
MNIST	CNN-MLP	288	(275, 300)
	CNN-KAN	323	(309, 336)
EMNIST	CNN-MLP	414	(400, 428)
	CNN-KAN	527	(511, 543)
CIFAR-10	CNN-MLP	193	(183, 203)
	CNN-KAN	253	(242, 264)

5.3 Future Work

Future research could focus on optimizing the training process for KAN layers to reduce computational time without compromising performance. Techniques such as parallel processing, more efficient algorithms for B-spline computations, and hardware accelerations could be explored to make KAN models more

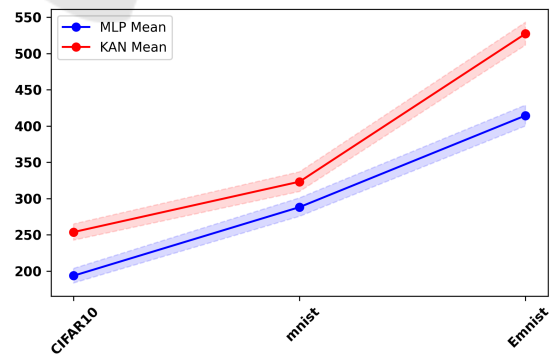


Figure 4: Mean computational time (in seconds) for training MLP and KAN.

feasible for time-sensitive applications. To summarize, while KAN and CNN-KAN models require more computational resources and longer training times,

their superior performance in terms of accuracy and robustness makes them valuable for complex image classification tasks. The trade-off between computational complexity and model performance should be carefully considered based on the specific needs of the application.

6 CONCLUSION

In this paper, we introduced a novel classification approach that integrates Kolmogorov-Arnold Networks (KANs) with Convolutional Neural Networks (CNNs), termed CNN-KAN. This innovative architecture harnesses the powerful feature extraction capabilities of CNNs and the sophisticated modeling of complex relationships provided by KANs. Our extensive evaluations across multiple datasets demonstrate that CNN-KAN consistently outperforms traditional CNN architectures in terms of accuracy, precision, and recall.

We also explored the application of pretrained models, showing that the proposed CNN-KAN models not only enhance efficiency but also improve robustness. The experimental results clearly indicate that the integration of KAN layers into CNN architectures leads to significant performance gains in diverse image classification tasks.

In summary, the ConvKAN approach represents a promising advancement in the field of computer vision, offering a robust and efficient solution for complex image classification challenges. This work paves the way for future research to further optimize and expand the capabilities of CNN-KAN models.

REFERENCES

- Amosa, T. I., Sebastian, P., Izhar, L. I., Ibrahim, O., Ayinla, L. S., Bahashwan, A. A., Bala, A., and Samaila, Y. A. (2023). Multi-camera multi-object tracking: a review of current trends and future advances. *Neurocomputing*, 552:126558.
- Balazevic, I., Steiner, D., Parthasarathy, N., Arandjelović, R., and Henaff, O. (2024). Towards in-context scene understanding. *Advances in Neural Information Processing Systems*, 36.
- Chen, X., Peng, H., Wang, D., Lu, H., and Hu, H. (2023). Seqtrack: Sequence to sequence learning for visual object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14572–14581.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- Fan, D.-P., Ji, G.-P., Xu, P., Cheng, M.-M., Sakaridis, C., and Van Gool, L. (2023). Advances in deep concealed scene understanding. *Visual Intelligence*, 1(1):16.
- Genet, R. and Inzirillo, H. (2024). A temporal kolmogorov-arnold transformer for time series forecasting. *arXiv preprint arXiv:2406.02486*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Jmour, N., Zayen, S., and Abdelkrim, A. (2018). Convolutional neural networks for image classification. In *2018 international conference on advanced systems and electric technologies (ICASET)*, pages 397–402. IEEE.
- Kim, H. E., Cosa-Linan, A., Santhanam, N., Jannesari, M., Maros, M. E., and Ganslandt, T. (2022). Transfer learning for medical image classification: a literature review. *BMC medical imaging*, 22(1):69.
- Koonce, B. and Koonce, B. (2021). Vgg network. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*, pages 35–50.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. (2024). Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.
- Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., and Bethge, M. (2022). Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21:1546–1726.
- Meinhardt, T., Kirillov, A., Leal-Taixe, L., and Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8844–8854.
- Nath, S. S., Mishra, G., Kar, J., Chakraborty, S., and Dey, N. (2014). A survey of image classification methods and techniques. In *2014 International conference on control, instrumentation, communication and computational technologies (ICCICCT)*, pages 554–557. IEEE.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520.
- Shi, J.-C., Wang, M., Duan, H.-B., and Guan, S.-H. (2024). Language embedded 3d gaussians for open-vocabulary scene understanding. In *Proceedings of*

- the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5333–5343.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.
- Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR.
- Wang, Y., Sun, J., Bai, J., Anitescu, C., Eshaghi, M. S., Zhuang, X., Rabczuk, T., and Liu, Y. (2024). Kolmogorov arnold informed neural network: A physics-informed deep learning framework for solving pdes based on kolmogorov arnold networks. *arXiv preprint arXiv:2406.11045*.
- Wotton, J. and Gunes, H. (2020). Deep learning for real-time robust facial expression recognition. *IEEE Transactions on Affective Computing*, 11(3):532–541.
- Yu, H. and Xiong (2022). Scratch-aid, a deep learning-based system for automatic detection of mouse scratching behavior with high accuracy. *eLife*, 11:e84042.
- Zhang, Q. (2022). A novel resnet101 model based on dense dilated convolution for image classification. *SN Applied Sciences*, 4:1–13.