MLP-KAN: UNIFYING DEEP REPRESENTATION AND FUNCTION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements in both representation learning and function learning have demonstrated substantial promise across diverse domains of artificial intelligence. However, the effective integration of these paradigms poses a significant challenge, particularly in cases where users must manually decide whether to apply a representation learning or function learning model based on dataset characteristics. To address this issue, we introduce MLP-KAN, a unified method designed to eliminate the need for manual model selection. By integrating Multi-Layer Perceptrons (MLPs) for representation learning and Kolmogorov-Arnold Networks (KANs) for function learning within a Mixture-of-Experts (MoE) architecture, MLP-KAN dynamically adapts to the specific characteristics of the task at hand, ensuring optimal performance. Embedded within a transformer-based framework, our work achieves remarkable results on four widely-used datasets across diverse domains. Extensive experimental evaluation demonstrates its superior versatility, delivering competitive performance across both deep representation and function learning tasks. These findings highlight the potential of MLP-KAN to simplify the model selection process, offering a comprehensive, adaptable solution across various domains.

1 INTRODUCTION

In recent years, deep learning has evolved from early neural network concepts to sophisticated architectures, such as transformer networks (Vaswani, 2017), driven by advancements in computational resources and the availability of large datasets, thereby achieving remarkable performance across diverse applications. Along with important technological breakthroughs, representation learning (OpenAI, 2023a; Anthropic, 2024; OpenAI, 2023b; Touvron et al., 2023) and function learning (Narayan et al., 1996; Zhang et al., 2022; Wu et al., 2005) moments of prominence and have been extensively explored and utilized in various research and application tasks related to data and learning nowadays. At the same time, the focus of function learning research has shifted from simple function fitting to deep learning (Cuomo et al., 2022; Cai et al., 2021), which excels in tasks requiring precise function approximation and has seen new advancements, particularly in its applicability to univariate function tasks. The key difference between representation learning and function learning lies in their objectives: representation learning aims to extract features from data to understand its underlying structure (Bengio et al., 2013), while function learning focuses on creating direct mappings between inputs and outputs, making it more suited for tasks requiring precise functional relationships (Zupan et al., 1997).

In this paper, we introduce MLP-KAN, a novel framework that unifies two distinct learning approaches into a cohesive system, utilizing the Mixture of Experts (MoE) methodology Jiang et al. (2023). Within the architecture of MLP-KAN, Multi-Layer Perceptrons (MLP) (Rumelhart et al., 1986) function as representation experts, while Kolmogorov-arnold networks(KAN) (Liu et al., 2024) are designated as function experts. The MoE mechanism efficiently routes inputs to the appropriate expert, significantly enhancing both efficiency and performance across a diverse range of tasks. MLP-KAN was developed to address the problem users encounter when determining whether to apply representation learning or function learning models across diverse datasets. By integrating MLPs and KANs within a mixture-of-experts framework, this architecture dynamically adapts to the in Figure 1, ensuring optimal performance without requiring manual model selection. The main challenge in our method is effectively integrating MLPs and KANs, ensuring the right model

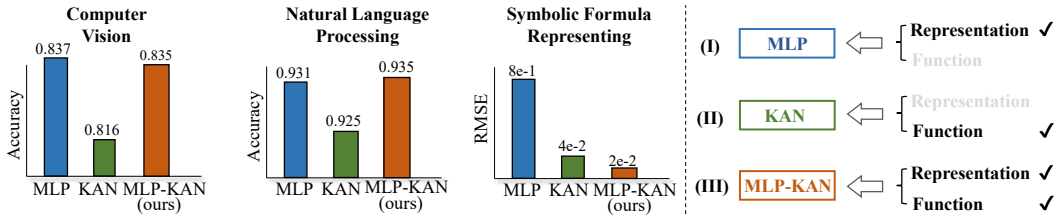

Figure 1: The comparison between the MLP, KAN, and our proposed MLP-KAN. In the domains of Computer Vision and Natural Language Processing, the goal is to achieve the highest accuracy possible. In contrast, for the Symbolic Formula Representation task, the objective is to minimize the root mean square error (RMSE). The numbers are the average values of the experimental results. MLP-KAN effectively combines the strengths of both, ensuring strong performance in representation and function learning, and eliminating the need for task-specific model selection.

is selected for each task without compromising performance. In addition, aligning the distinct training requirements of representation and functional learning while ensuring efficiency across various datasets poses a significant challenge.

To address the challenge of effectively integrating MLPs and KANs within the MoE framework, we utilized a soft MoE approach. This method enables dynamic and flexible routing between MLPs for representation learning and KANs for function learning. By incorporating this MoE system within a transformer framework, the model can seamlessly perform deep representation learning or deep function learning, adapting to the specific nature of the task at hand while maintaining efficiency across diverse datasets.

The main contributions of this work are as follows:

- We present MLP-KAN, a unified framework that synergizes MLP for representation learning with KAN for function learning. This novel architecture leverages a MoE mechanism to dynamically route tasks between representation and function experts, addressing the challenge of selecting the appropriate learning paradigm for diverse datasets.
- We propose a flexible and versatile model by integrating MLP-KAN within the transformer architecture, enabling efficient performance across both representation and function learning tasks. This integration enhances model capability and improves performance across a broad range of tasks, including computer vision, natural language processing, and symbolic formula representation.
- We perform extensive experimental evaluations, demonstrating that MLP-KAN consistently outperforms or matches state-of-the-art models such as MLP and KAN on widely recognized benchmarks, including computer vision,nature language processing, and functional dataset. Our approach achieves superior accuracy in representation learning tasks and lower RMSE in function learning tasks, underscoring its universal applicability across diverse domains.

2 Related Work

Deep Representation Learning. Deep representation learning has gained significant attention due to its ability to automatically discover hierarchical feature representations from raw data (Butepage et al., 2017; Zhong et al., 2016; Long et al., 2018), outperforming traditional hand-crafted feature extraction techniques. The introduction of deep learning methods, such as MLP based convolutional neural networks (Li et al., 2021) and recurrent neural networks, enabled breakthroughs in areas like image recognition (Zoph et al., 2018; He et al., 2016), object detection (Zhao et al., 2019; Yu et al., 2016; Liu et al., 2020), and natural language processing (Chowdhary & Chowdhary, 2020; Khurana et al., 2023) by capturing more abstract and high-level features. Recent advancements in deep architectures, including transformer-based models (Gillioz et al., 2020), have further pushed the boundaries of representation learning, proving highly effective across diverse domains. For example, generative AI, such as large language models (LLMs) (Yao et al., 2024; Zhao et al., 2023),

has garnered significant attention for its ability to generate coherent, contextually relevant text and learn deep representations from vast amounts of unstructured data. LLMs like GPT-4o (OpenAI, 2024) and LLaMA (Touvron et al., 2023) utilize MLP based transformer architectures, which excel at capturing long-range dependencies in sequential data, allowing them to perform tasks such as text generation, summarization, and translation with remarkable accuracy. Beyond natural language processing, LLMs have also influenced other fields, including code generation (Chung et al., 2024; Li et al., 2022), medical diagnosis (Kononenko, 2001; Amato et al., 2013), and drug discovery (Drews, 2000; Sliwoski et al., 2014), by leveraging their deep learning capabilities to model complex relationships in data. These advancements highlight the growing importance of deep representation learning in not only understanding and generating human-like text but also in solving a wide range of interdisciplinary challenges (Newell et al., 2001). In these models, MLP play a crucial role as fundamental building blocks, serving as dense layers that transform and learn high-dimensional representations by mapping inputs to deeper abstract features (Donoho et al., 2000).

Deep Function Learning. Deep function learning focuses on capturing complex mathematical relationships and patterns within data, particularly in scientific and engineering domains (Sarker, 2021; Shen, 2018; Karpatne et al., 2017). Techniques such as Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) have emerged as powerful tools for solving partial differential equations (PDEs) (Evans, 2022) by embedding physical laws into neural network architectures, allowing for accurate modeling of phenomena governed by underlying physical principles (Raissi et al., 2019; Cuomo et al., 2022). Beyond traditional neural networks, deep function learning leverages overparameterized models, which enable the precise interpolation of data, even in the presence of noise, enhancing both generalization and optimization performance (Karniadakis et al., 2021; Advani et al., 2020; Chen et al., 2022). Recent advancements have demonstrated the potential of these methods for tasks such as surrogate modeling (Razavi et al., 2012), sensitivity analysis (Christopher Frey & Patil, 2002; Lenhart et al., 2002), and discovery of new scientific relationships (Wren et al., 2004; Klahr & Simon, 1999). KAN are highly effective for function learning due to their ability to capture complex non-linear relationships through learnable spline-based univariate functions, offering superior approximation capabilities and scaling compared to traditional MLP (Yu et al., 2024; Liu et al., 2024; Zhang, 2024; Vaca-Rubio et al., 2024).

3 PRELIMINARY

Table 1: Comparison between MLP and KAN.

Feature	MLPs	KANs
Activation Functions	Fixed functions (e.g., ReLU, SiLU)	$\varphi(x) = \sum_{i=1}^{k} c_i B_i(x)$
Weight Structure	Scalar weights	Spline-based weights $\varphi(x)$
Layer Architecture	Standard fixed depth	$\Phi_q \left(\sum_{p=1}^{n} \varphi_{q,p}(x_p) \right)$
Error Scaling	Limited by dimensionality	$\|f - (KAN)\|_{C^m} \leq CG^{-k-1+m}$
Scaling Law	$\ell \propto N^{-\alpha}$ with lower α	$\ell \propto N^{-\alpha}$ with higher $\alpha = 4$
Expressiveness	Suited for general representation learning	Suited for functional learning

KAN are inspired by the Kolmogorov-Arnold Representation Theorem (Liu et al., 2024), which asserts that any multivariate continuous function $f(x)$ can be decomposed into a sum of univariate functions. This is formally stated as:

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

where $\varphi_{q,p}(x_p)$ and Φ_q are univariate functions, summing over q and p. Unlike traditional Multi-Layer Perceptrons (MLPs), which use fixed activation functions at each neuron, KANs introduce learnable univariate activation functions on the edges between layers (Vaca-Rubio et al., 2024;

Aghaei, 2024). Each weight in KANs is replaced by a learnable spline function:

$$\varphi(x) = \sum_{i=1}^{k} c_i B_i(x) \tag{2}$$

where $B_i(x)$ are basis functions (such as B-splines) and c_i are trainable coefficients (Eilers & Marx, 1996). This spline-based approach allows KANs to better capture non-linear relationships, particularly in high-dimensional tasks where MLPs tend to struggle.

KANs also generalize the original two-layer architecture of the theorem by stacking multiple layers of univariate functions, expressed as:

$$KAN(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_0)(x) \tag{3}$$

The approximation capabilities of KANs scale better compared to MLPs, as shown in Table 1. The error bound for KANs with splines of order k and grid size G is $\|f - (KAN)\|_{C^m} \leq CG^{-k-1+m}$ where C is a constant, and m represents the order of derivatives considered. Furthermore, KANs exhibit superior neural scaling laws, with the test loss decreasing as $\ell \propto N^{-\alpha}$ where N is the number of parameters and α depends on the spline order k. For cubic splines ($k = 3$), KANs achieve $\alpha = 4$, outperforming MLPs, which often cannot reach these scaling efficiencies. This makes KANs particularly effective for high-dimensional function approximation (Sprecher & Draghici, 2002; Köppen, 2002).

4 METHODOLOGY

4.1 MLP-KAN

As shown in Figure 2, our proposed MLP-KAN is composed of NE experts, which can be classified into two types: representation experts and function experts. Representation experts, based on MLP architectures, focus on learning rich feature representations, while function experts, utilizing FasterKAN architectures, specialize in tasks requiring smooth and precise interpolation over continuous data points. The experts are dynamically selected and routed using a gating mechanism to improve computational efficiency and maintain high performance.

Representation Expert. In the context of MLP-KAN models, half of the experts are designed as representation experts, utilizing multi-layer perceptrons (MLPs). These experts excel in tasks requiring the learning of rich feature representations, such as image classification. Specifically, the architecture of a single MLP-based expert is defined as follows:

$$\text{Expert}_i = \text{MLP}(\mathbf{X}) \quad \text{for } i = 1, \ldots, \frac{NE}{2} \tag{4}$$

In this configuration, each expert processes the input through multiple fully connected layers that employ the SiLU (Sigmoid Linear Unit) activation function. Unlike ReLU (Rectified Linear Unit) (Hahnloser et al., 2000), SiLU provides smooth gradients and mitigates the issue of dying neurons, enhancing the robustness and efficiency of learning.

The process of forward propagation within each expert is executed as follows: Given an input $\mathbf{X} \in \mathbb{R}^{B \times N \times D}$, where B is the batch size, N is the sequence length, and D is the feature dimension, the transformation through the MLP involves applying a linear transformation followed by the SiLU activation function:

$$\mathbf{h}^{(1)} = \text{SiLU}(\mathbf{W}^{(1)}\mathbf{X} + \mathbf{b}^{(1)}), \ \mathbf{h}^{(2)} = \mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)} \tag{5}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{H \times D}$ and $\mathbf{W}^{(2)} \in \mathbb{R}^{D' \times H}$ are the weight matrices, and $\mathbf{b}^{(1)} \in \mathbb{R}^{H}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{D'}$ are the bias vectors of the corresponding layers. The output $\mathbf{h}^{(2)}$ is passed on for further processing.

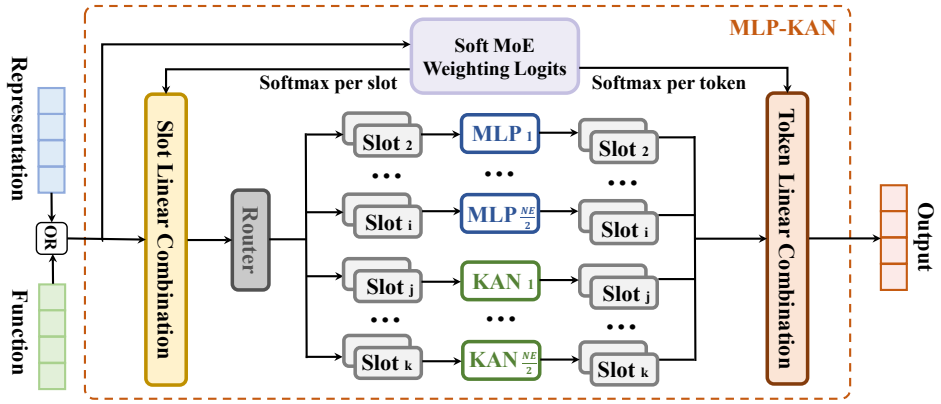

Figure 2: The framework combines a soft mixture of experts (MoE) with a unification of MLPs and KANs, denoted as the MLP-KAN module, to dynamically select experts for each token. The input tokens are passed through a multi-headed self-attention mechanism followed by layer normalization. The routing process involves soft weighting of experts for each slot and token via linear combinations and a softmax layer per slot and token. MLP and KAN experts are arranged in parallel, and based on the input's characteristics, either MLP or KAN is selected for computation, enhancing the model's ability to handle diverse representations efficiently. The gating mechanism determines the most relevant expert for each token, improving overall computational efficiency. This architecture retains the residual connections of the traditional Transformer while expanding its capacity to model complex functional and representational data.

Function Expert. The other half of the experts in MLP-KAN are defined as function experts to handle specialized data, particularly in functional datasets. These experts are based on the FasterKAN (Delis, 2024) architecture, which is known for its strong performance in tasks requiring smooth interpolation over continuous data points.

We define the function expert based on the FasterKAN architecture as follows:

$$\text{Expert}_i = \text{FasterKAN}(\mathbf{X}) \quad \text{for } i = \frac{NE}{2} + 1, \ldots, NE \tag{6}$$

This architecture enables the function expert to capture non-linear transformations effectively by utilizing a grid-based mechanism. Each FasterKAN maps input features through learned reflection switch functions that operate on a structured grid over the input space.

The transformation of an input $\mathbf{X} \in \mathbb{R}^{B \times N \times D}$ through the expert's layers follows these steps:

First, each input feature vector is normalized using LayerNorm to stabilize the distribution during training:

$$\mathbf{X}_{\text{norm}} = \text{LayerNorm}(\mathbf{X}) \tag{7}$$

Subsequently, the reflectional switch function $\phi(\mathbf{x})$ computes the differences between the normalized input, predefined grid points and hyper-parameter denominator, followed by a non-linear transformation to approximate smooth basis functions:

$$\phi(\mathbf{X}) = 1 - \tanh\left(\frac{\mathbf{X} - \text{grid}}{\text{denominator}}\right)^2 \tag{8}$$

Lastly, the computed basis values are passed through a spline transformation $\mathbf{W}_{\text{spline}}$ to map the input to the output dimension:

$$\mathbf{y} = \mathbf{W}_{\text{spline}} \cdot \phi(\mathbf{X}) \tag{9}$$

By integrating FasterKAN for half of the experts, MLP-KAN is well-equipped to process functional data, leveraging FasterKAN's interpolation across a smooth grid representation. The remaining experts can follow alternative architectures, allowing MLP-KAN to dynamically select the optimal model based on the input's characteristics.

Gating Mechanism. In MLP-KAN, the gating mechanism serves a pivotal function in dynamically routing input tokens to the most relevant experts. This mechanism efficiently selects a subset of experts for each input sequence, reducing computational overhead while maintaining robust model performance.

Given an input sequence $\mathbf{X} \in \mathbb{R}^{B \times N \times D}$, the gating mechanism computes the similarity between the input tokens and a set of learnable slot embeddings $\mathbf{E} \in \mathbb{R}^{NE \times S \times D}$, where NE is the number of experts and S is the number of slots per expert. This similarity is calculated as follows:

$$\text{logits}_{b,n,e,s} = \langle \mathbf{X}_{b,n,:}, \mathbf{E}_{e,s,:} \rangle, \quad \text{for } b \in [1, B], n \in [1, N], e \in [1, NE], s \in [1, S] \tag{10}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, and the resulting logits logits $\in \mathbb{R}^{B \times N \times E \times S}$ represent the unnormalized attention scores between each token and the expert slots.

Next, a softmax function is applied along the expert and slot dimensions to compute the dispatch weights $\alpha \in \mathbb{R}^{B \times N \times E \times S}$, determining the contribution of each token to each expert:

$$\alpha_{b,n,e,s} = \frac{\exp(\text{logits}_{b,n,e,s})}{\sum_{e',s'} \exp(\text{logits}_{b,n,e',s'})} \tag{11}$$

These dispatch weights α are then used to aggregate the input tokens across the sequence for each expert, resulting in routed inputs $\mathbf{z} \in \mathbb{R}^{B \times E \times S \times D}$:

$$\mathbf{z}_{b,e,s,:} = \sum_{n=1}^{N} \alpha_{b,n,e,s} \mathbf{X}_{b,n,:} \tag{12}$$

Finally, each expert processes its routed inputs, and the outputs from all experts are aggregated using softmax-normalized combination weights. This ensures that the final output $\mathrm{F}(\mathbf{X})$ is a unified combination of contributions from all experts, based on the initial input $\mathbf{X}$.

4.2 Integration into Transformer Architecture.

To enhance the capacity of standard Transformers, we replace the MLP layers in each block with MLP-KAN modules. As shown in Figure 3, the output of the Transformer block is computed as:

$$\mathbf{Y}_l = \mathbf{X}_l + \mathrm{MHA}(\mathrm{LN}(\mathbf{X}_l)) + \frac{1}{NE} \sum_{e=1}^{NE} \mathrm{F}_e(\mathrm{LN}(\mathbf{X}_l + \mathrm{MHA}(\mathrm{LN}(\mathbf{X}_l)))) \tag{13}$$

In this formulation, l represents the layer index, ranging from 1 to L, where L is the total number of layers in the model. NE denotes the total number of experts in the MLP-KAN module for each layer, ensuring sufficient diversity of expertise. The function F_e corresponds to the computation performed by the e-th expert, which is dynamically selected by the gating mechanism to handle specific token characteristics efficiently.

This formula underscores that the output of each layer l is computed by adaptively combining the contributions from all NE experts. This dynamic selection mechanism ensures that the overall computation remains scalable across L layers, while effectively tailoring the model's capacity to the input tokens at each step.

5 Experiment

5.1 Experimental Setup

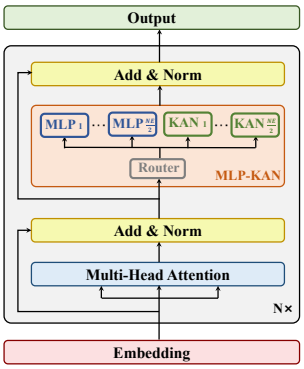

Figure 3: Architecture of the transformer encoder with MLP-KAN Integration.

Datasets. We have validated the effectiveness of our method on several public datasets. In representation learning, we have validated the CIFAR-10, CIFAR-100, and mini-ImageNet datasets (Krizhevsky et al., 2010; Vinyals et al., 2016) in the field of computer vision, and the SST2 dataset (Socher et al., 2013) in the field of natural language processing. In function learning, we have validated thirty functions on the Feynman dataset (Udrescu & Tegmark, 2020). The CIFAR-10 and CIFAR-100 datasets are the tasks of image classification, both consisting of 50,000 images for the training set and 10,000 images for the test set. However, the former has only 10 categories, while the latter has 100 categories. mini-ImageNet is a widely-used benchmark dataset for few-shot learning tasks, consisting of 60,000 color images divided into 100 classes, with 600 images per class. Both CV datasets use top-1 accuracy (top1-acc.) and top-5 accuracy (top5-acc.) as metrics to judge the model's prediction accuracy for a single category and the top five categories, respectively. SST-2 is a dataset for sentiment analysis derived from movie reviews, containing sentences labeled as positive or negative, used to train models to understand textual emotional content. Specifically, we use the F1 score (F1) and the accuracy score (Acc) to measure performance. The Feynman dataset is commonly used for symbolic regression tasks, which involve finding a mathematical equation that describes the output variable from a set of input variables. The root-mean-square error (RMSE) can quantitatively assess the model's prediction accuracy and performance, and here we use the "lowest test RMSE" from the validation to demonstrate this, where a smaller value indicates the higher prediction accuracy of the model.

Training and Evaluation Details. To comprehensively demonstrate the superiority of MLP-KAN, our experimental setup involved comparisons with MLP and KAN. These extensive experiments demonstrate that our method can be universally applied across various domains and consistently achieves excellent results. All experiments were conducted using four A100 GPUs. During the training phase, we meticulously tuned parameters to optimize the learning process. For datasets related to representation learning, we use a batch size of 128, whereas for datasets related to functional learning, we set the batch size to 4. The learning rate was initially set at 5e-5, and the training continues until convergence. We applied dropout to the output of each MLP-KAN using a dropout rate of 0.1. Regarding the hyperparameters of MLP-KAN, we configured $n = 8$ (i.e., 8 experts) and $k = 2$ (i.e., top2 experts).

5.2 Function Learning

The results from Table 2 demonstrate that MLP-KAN significantly outperforms both MLP and KAN across a variety of equations. or simpler equations like I.6.20a, MLP-KAN achieves an RMSE of 3.87×10^{-4}, which is much lower than KAN's 8.82×10^{-4} and MLP's 1.37×10^{-1}. This illustrates our method's ability to accurately capture basic functional relationships with far fewer errors than MLP, which often over-parameterizes for simple tasks. For more complex equations involving multiple variables, such as I.9.18, MLP-KAN maintains a strong advantage, achieving an RMSE of 3.13×10^{-3} compared to KAN's 4.87×10^{-3} and MLP's much higher 1.40×10^{-2}. This shows that our MLP-KAN scales effectively and can manage the intricacies of complex interactions that MLP struggles to capture without excessive parameters. Our proposed MLP-KAN demonstrates versatility across different types of equations, such as in I.12.5, where it achieves a lower RMSE (3.61×10^{-3}) than both KAN and MLP. The results reflect its ability to adapt dynamically to different functional forms, from basic algebraic equations to those involving physical constants and nonlinearities. n physics-based equations like I.15.3t, which involves relativistic transformations, MLP-KAN outperforms both KAN and MLP with an RMSE of 7.18×10^{-2} compared to KAN's 3.69×10^{-2} and MLP's 3.44×10^{-1}. This indicates the superior ability of our method to generalize across equations that require deep understanding of physical laws. Our proposed achieves superior performance without the excessive parameter overhead required by MLPs, making it computationally efficient. For example, in I.14.4, MLP-KAN achieves an RMSE of 6.79×10^{-3}, far outperforming MLP's 1.11×10^{-1}, demonstrating that MLP-KAN can achieve better accuracy with fewer resources. Across almost all equations, MLP-KAN consistently outperforms both KAN and MLP, often achieving RMSEs that are orders of magnitude smaller. This consistent superiority highlights

Table 2: Comparison of losses for Feynman Equations. Results highlighted in **bold** represent the best performance in the comparison, while those underlined represent the second-best results. Each Experiment using learning rate is 0.001 and epochs are 1000.

Feynman Eq.	Original Formula	Variables	KAN loss	MLP loss	MLP-KAN loss
$I.6.20a$	$\frac{e^{-\theta^2/2}}{\sqrt{2\pi}}$	θ	8.82×10^{-4}	1.37×10^{-1}	$\mathbf{3.87 \times 10^{-4}}$
$I.6.20$	$\frac{e^{-\theta^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$	θ, σ	1.42×10^{-2}	1.20×10^{-1}	$\mathbf{8.44 \times 10^{-3}}$
$I.6.20b$	$\frac{e^{-(\theta-\theta_1)^2/2\sigma^2}}{\sqrt{2\pi\sigma^2}}$	θ, θ_1, σ	1.59×10^{-2}	1.16×10^{-1}	$\mathbf{4.99 \times 10^{-3}}$
$I.8.4$	$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$	x_1, x_2, y_1, y_2	$\mathbf{4.58 \times 10^{-3}}$	1.91×10^{-1}	1.23×10^{-2}
$I.9.18$	$\frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	$G, m_1, m_2, x_1, x_2, y_1, y_2, z_1, z_2$	4.87×10^{-3}	1.40×10^{-2}	$\mathbf{3.13 \times 10^{-3}}$
$I.10.7$	$\frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$	m_0, v, c	$\mathbf{2.04 \times 10^{-2}}$	3.22×10^{-1}	1.46×10^{-1}
$I.11.19$	$x_1y_1 + x_2y_2 + x_3y_3$	$x_1, y_1, x_2, y_2, x_3, y_3$	3.37×10^{-2}	9.89×10^{-2}	$\mathbf{2.65 \times 10^{-2}}$
$I.12.1$	μN_n	μ, N_n	9.22×10^{-3}	3.34×10^{-1}	$\mathbf{7.17 \times 10^{-3}}$
$I.12.2$	$\frac{q_1q_2}{4\pi\epsilon r^2}$	q_1, q_2, ϵ, r	6.75×10^{-3}	4.75×10^{-2}	$\mathbf{3.06 \times 10^{-3}}$
$I.12.4$	$\frac{q_1}{4\pi\epsilon r^2}$	q_1, ϵ, r	5.62×10^{-3}	4.87×10^{-2}	$\mathbf{3.86 \times 10^{-3}}$
$I.12.5$	q_2E_f	q_2, E_f	$\mathbf{2.93 \times 10^{-3}}$	3.25×10^{-1}	3.61×10^{-3}
$I.12.11$	$q(E_f + Bv\sin(\theta))$	q, E_f, B, v, θ	6.38×10^{-2}	1.85×10^{-1}	$\mathbf{3.56 \times 10^{-2}}$
$I.13.4$	$\frac{1}{2}m(v^2+u^2+w^2)$	m, v, u, w	2.10×10^{-2}	1.26×10^{-1}	$\mathbf{9.68 \times 10^{-3}}$
$I.13.12$	$Gm_1m_2\left(\frac{1}{r_2}-\frac{1}{r_1}\right)$	G, m_1, m_2, r_1, r_2	$\mathbf{8.69 \times 10^{-3}}$	3.87×10^{-2}	9.78×10^{-3}
$I.14.3$	mgz	m, g, z	8.98×10^{-3}	1.64×10^{-1}	$\mathbf{2.80 \times 10^{-3}}$
$I.14.4$	$\frac{1}{2}k_sx^2$	k_s, x	$\mathbf{5.13 \times 10^{-3}}$	1.11×10^{-1}	6.79×10^{-3}
$I.15.3x$	$\frac{x-ut}{\sqrt{1-\frac{u^2}{c^2}}}$	x, u, t, c	$\mathbf{3.50 \times 10^{-2}}$	3.48×10^{-1}	8.52×10^{-2}
$I.15.3t$	$\frac{t-ux/c^2}{\sqrt{1-\frac{u^2}{c^2}}}$	t, u, x, c	$\mathbf{3.69 \times 10^{-2}}$	3.44×10^{-1}	7.18×10^{-2}
$I.15.10$	$\frac{m_0v}{\sqrt{1-\frac{v^2}{c^2}}}$	m_0, v, c	2.36×10^{-2}	2.27×10^{-1}	$\mathbf{1.47 \times 10^{-2}}$
$I.16.6$	$\frac{u+v}{1+\frac{uv}{c^2}}$	u, v, c	$\mathbf{8.73 \times 10^{-3}}$	1.45×10^{-1}	1.06×10^{-2}
$I.18.4$	$\frac{m_1r_1+m_2r_2}{m_1+m_2}$	m_1, r_1, m_2, r_2	$\mathbf{6.18 \times 10^{-3}}$	2.33×10^{-1}	2.26×10^{-2}
$I.18.5$	$rF\sin(\theta)$	r, F, θ	5.67×10^{-2}	2.03×10^{-1}	$\mathbf{4.93 \times 10^{-2}}$
$I.18.16$	$mrv\sin(\theta)$	m, r, v, θ	6.88×10^{-2}	1.02×10^{-1}	$\mathbf{3.40 \times 10^{-2}}$
$I.24.6$	$\frac{1}{4}m(\omega^2+\omega_0^2)x^2$	m, ω, ω_0, x	7.99×10^{-3}	6.20×10^{-2}	$\mathbf{5.87 \times 10^{-3}}$
$I.25.13$	$\frac{q}{C}$	q, C	1.07×10^{-2}	5.17×10^{-1}	$\mathbf{8.33 \times 10^{-3}}$
$I.26.2$	$\arcsin(n\sin(\theta_2))$	n, θ_2	2.74×10^{-2}	4.45×10^{-1}	$\mathbf{1.15 \times 10^{-2}}$
$I.27.6$	$\frac{1}{1/d_1+n/d_2}$	d_1, d_2, n	$\mathbf{5.97 \times 10^{-3}}$	1.42×10^{-1}	6.18×10^{-3}
$I.29.4$	$\frac{\omega}{c}$	ω, c	5.27×10^{-3}	2.26×10^{-1}	$\mathbf{3.45 \times 10^{-3}}$
$I.29.16$	$\sqrt{x_1^2+x_2^2-2x_1x_2\cos(\theta_1-\theta_2)}$	$x_1, x_2, \theta_1, \theta_2$	8.48×10^{-2}	2.91×10^{-1}	$\mathbf{5.31 \times 10^{-2}}$
$I.30.3$	$I_0\frac{\sin^2(n\theta/2)}{\sin^2(\theta/2)}$	I_0, n, θ	2.24×10^{-1}	4.07×10^{-1}	$\mathbf{1.99 \times 10^{-1}}$
Avg.			$(2.69 \pm 0.53) \times 10^{-2}$	$(2.04 \pm 0.41) \times 10^{-1}$	$(\mathbf{2.58 \pm 0.48}) \times \mathbf{10^{-2}}$

Table 3: Comparison of results in representation learning. Results highlighted in **bold** represent the best performance in the comparison, while those underlined represent the second-best results. Each Experiment using learning rate is $5 \times \mathbf{10^{-4}}$ and epochs are 300.

Method	Dataset: CIFAR-10		Dataset: CIFAR-100		Dataset: mini-ImageNet		Dataset: SST2	
	Acc1	Acc5	Acc1	Acc5	Acc1	Acc5	Acc	F1
KAN	0.904±0.019	0.989±0.008	0.731±0.022	0.933±0.015	0.623±0.025	0.803±0.018	0.925±0.009	0.925±0.011
MLP	**0.922±0.011**	**0.997±0.006**	**0.752±0.016**	**0.958±0.011**	**0.680±0.021**	**0.845±0.013**	0.931±0.007	0.930±0.008
MLP-KAN	0.920±0.008	0.996±0.004	0.750±0.019	0.952±0.011	0.679±0.021	0.843±0.010	**0.935±0.006**	**0.933±0.010**

MLP-KAN 's versatility and adaptability to both simple and complex mathematical forms, making it the most robust and efficient solution for function learning across diverse domains.

5.3 Representation Learning

As shown in Table 3, our proposed MLP-KAN shows consistent high performance, demonstrating particular strengths across diverse datasets. Notably, MLP-KAN achieves the second-best results for both top-1 and top-5 accuracy metrics on CIFAR-10, with scores of 0.920 and 0.996, respectively, closely trailing the MLP method. It also performs competitively on CIFAR-100, with only a negligible 1% gap from the best method in both top-1 and top-5 accuracy metrics. Furthermore, MLP-KAN consistently outperforms KAN, which achieves an Acc1 of 0.904 for CIFAR-10 and 0.731 for CIFAR-100. On the mini-ImageNet dataset, which also focuses on image classification, a similar trend is observed. In addition, MLP-KAN excels in the NLP task on the SST2 dataset, achieving the best results with an accuracy of 0.935 and an F1 score of 0.933. This superior performance highlights MLP-KAN's versatility and robustness in handling not only image data but also text data, making it an excellent choice for representation learning.

5.4 Ablation and Analysis

Number of Experts. In this ablation study, we investigate the impact of the number of experts in the MoE component of MLP-KAN on the performance of CIFAR-10 and CIFAR-100. As observed in Table 4, increasing the number of experts from 4 to 10 yields steady improvements in both top-1 and top-5 accuracy across both datasets. Notably, the top-1 accuracy for CIFAR-10 increases from 0.908 to 0.928, while CIFAR-100 improves from 0.742 to 0.755 when the number of experts increases from 4 to 10. However, performance gains begin to diminish after using 8 experts. The difference between using 8 and 10 experts is marginal: The accuracy of the top-1 of CIFAR-10 only increases by 0.8%, and CIFAR-100 sees a mere 0.5% improvement. While the model with 10 experts delivers slightly better results, the computational cost associated with using more experts becomes significant. Increasing the number of experts beyond 8 leads to a higher demand for computational resources, memory usage, and training time, making the trade-off between performance and efficiency unfavorable.

Table 4: Results of CIFAR-10 and CIFAR-100 accuracy with different numbers of experts.

Expert	CIFAR-10 (Acc1)	CIFAR-10 (Acc5)	CIFAR-100 (Acc1)	CIFAR-100 (Acc5)
8	**0.920**	**0.996**	**0.750**	**0.953**
4	0.908	0.990	0.742	0.950
6	0.914	0.996	0.740	0.952
10	**0.928**	**0.997**	**0.755**	**0.958**

Number of Top-K. In this ablation study, we examine the impact of varying the Top-K value on the accuracy of CIFAR-10 and CIFAR-100. As shown in Table 5, we experiment with Top-K values of 1, 2, and 3, measuring their impact on both top-1 and top-5 accuracy across both datasets. Interestingly, we observe that setting Top-K to 2 yields the best performance. For CIFAR-10, both top-1 and top-5 accuracies improve slightly compared to K=1. Specifically, the top-5 accuracy increases from 0.990 to 0.996, while top-1 remains constant at 0.920. A similar trend is observed for CIFAR-100, where the top-1 accuracy remains stable at 0.750, but top-5 accuracy improves slightly from 0.952 to 0.953. On the other hand, when Top-K is set to 3, we notice a decline in performance. Both CIFAR-10 and CIFAR-100 exhibit reduced accuracy, with CIFAR-10 top-1 accuracy dropping to 0.908 and CIFAR-100 top-1 accuracy falling to 0.742. This indicates that increasing Top-K beyond 2 leads to diminished returns, as the additional experts likely introduce more noise or less relevant expertise.

Table 5: Results of CIFAR-10 and CIFAR-100 accuracy with different Top-k values.

Top-k	CIFAR-10 (Acc1)	CIFAR-10 (Acc5)	CIFAR-100 (Acc1)	CIFAR-100 (Acc5)
2	**0.920**	**0.996**	**0.750**	**0.953**
1	0.920	0.990	0.750	0.952
3	0.908	0.991	0.742	0.949

6 Conclusion

In this paper, we propose a novel approach that effectively enhances both representation learning and function learning. This approach demonstrates excellent performance when integrated with MLP and KAN experts. Additionally, our proposed MLP-KAN can seamlessly replace the existing MLP layers in the transformer architecture. Furthermore, our extensive evaluations confirm that MLP-KAN significantly improves performance in each area.

REFERENCES

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

Alireza Afzal Aghaei. fkan: Fractional kolmogorov-arnold networks with trainable jacobi basis functions. *arXiv preprint arXiv:2406.07456*, 2024.

Filippo Amato, Alberto López, Eladia María Peña-Méndez, Petr Vaňhara, Aleš Hampl, and Josef Havel. Artificial neural networks in medical diagnosis, 2013.

Anthropic. The claude 3 model family: Opus, sonnet, haiku, 2024. URL `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf`.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Judith Butepage, Michael J Black, Danica Kragic, and Hedvig Kjellstrom. Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6158–6166, 2017.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12): 1727–1738, 2021.

Tianlong Chen, Xiaohan Chen, Wuyang Chen, Howard Heaton, Jialin Liu, Zhangyang Wang, and Wotao Yin. Learning to optimize: A primer and a benchmark. *Journal of Machine Learning Research*, 23(189):1–59, 2022.

KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

H Christopher Frey and Sumeet R Patil. Identification and review of sensitivity analysis methods. *Risk analysis*, 22(3):553–578, 2002.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Athanasios Delis. Fasterkan. `https://github.com/AthanasiosDelis/faster-kan/`, 2024.

David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, 1(2000):32, 2000.

Jurgen Drews. Drug discovery: a historical perspective. *science*, 287(5460):1960–1964, 2000.

Paul HC Eilers and Brian D Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, 11(2):89–121, 1996.

Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 2022.

Richard Phillips Feynman. *Feynman Lectures on Physics: Electrical and Magnetic Behavior. Volume 4*. Perseus Books, 1999.

Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. Overview of the transformer-based models for nlp tasks. In *2020 15th Conference on computer science and information systems (FedCSIS)*, pp. 179–183. IEEE, 2020.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *nature*, 405(6789):947–951, 2000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth, 2016. URL https://arxiv.org/abs/1603.09382.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: state of the art, current trends and challenges. *Multimedia tools and applications*, 82(3):3713–3744, 2023.

David Klahr and Herbert A Simon. Studies of scientific discovery: Complementary approaches and convergent findings. *Psychological Bulletin*, 125(5):524, 1999.

Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001.

Mario Köppen. On the training of a kolmogorov network. In *Artificial Neural Networks—ICANN 2002: International Conference Madrid, Spain, August 28–30, 2002 Proceedings 12*, pp. 474–479. Springer, 2002.

Alex Krizhevsky, Geoff Hinton, et al. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010.

T Lenhart, K Eckhardt, N Fohrer, and H-G Frede. Comparison of two different approaches of sensitivity analysis. *Physics and Chemistry of the Earth, Parts A/B/C*, 27(9-10):645–654, 2002.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12):6999–7019, 2021.

Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128:261–318, 2020.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Transferable representation learning with deep adaptation networks. *IEEE transactions on pattern analysis and machine intelligence*, 41(12):3071–3085, 2018.

Sridhar Narayan, Gene A Tagliarini, and Edward W Page. Enhancing mlp networks using a distributed data representation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):143–149, 1996.

William H Newell, Jay Wentworth, and David Sebberson. A theory of interdisciplinary studies. *Issues in Interdisciplinary Studies*, 2001.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023a.

OpenAI. Introducing chatgpt, 2023b. URL https://openai.com/blog/chatgpt.

OpenAI. Gpt-4o: Multimodal intelligence for text, audio, and vision in real time. *OpenAI Research Announcements*, 2024. URL https://www.openai.com/gpt4o. Accessed: 2024-05-13.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Saman Razavi, Bryan A Tolson, and Donald H Burn. Review of surrogate modeling in water resources. *Water Resources Research*, 48(7), 2012.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Iqbal H Sarker. Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN computer science*, 2(6):420, 2021.

Chaopeng Shen. A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resources Research*, 54(11):8558–8593, 2018.

Gregory Sliwoski, Sandeepkumar Kothiwale, Jens Meiler, and Edward W Lowe. Computational methods in drug discovery. *Pharmacological reviews*, 66(1):334–395, 2014.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

David A Sprecher and Sorin Draghici. Space-filling curves and kolmogorov superposition-based neural networks. *Neural Networks*, 15(1):57–67, 2002.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Cristian J Vaca-Rubio, Luis Blanco, Roberto Pereira, and Màrius Caus. Kolmogorov-arnold networks (kans) for time series analysis. *arXiv preprint arXiv:2405.08790*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3630–3638, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/90e1357833654983612fb05e3ec9148c-Abstract.html.

Jonathan D Wren, Raffi Bekeredjian, Jelena A Stewart, Ralph V Shohet, and Harold R Garner. Knowledge discovery by automated identification and ranking of implicit relationships. *Bioinformatics*, 20(3):389–398, 2004.

Dalei Wu, Andrew Morris, and Jacques Koreman. Mlp internal representation as discriminative features for improved speaker recognition. In *International Conference on Nonlinear Analyses and Algorithms for Speech Processing*, pp. 72–80. Springer, 2005.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, pp. 100211, 2024.

Jiahui Yu, Yuning Jiang, Zhangyang Wang, Zhimin Cao, and Thomas Huang. Unitbox: An advanced object detection network. In *Proceedings of the 24th ACM international conference on Multimedia*, pp. 516–520, 2016.

Runpeng Yu, Weihao Yu, and Xinchao Wang. Kan or mlp: A fairer comparison. *arXiv preprint arXiv:2407.16674*, 2024.

David Junhao Zhang, Kunchang Li, Yali Wang, Yunpeng Chen, Shashwat Chandra, Yu Qiao, Luoqi Liu, and Mike Zheng Shou. Morphmlp: An efficient mlp-like backbone for spatial-temporal representation learning. In *European Conference on Computer Vision*, pp. 230–248. Springer, 2022.

Jiawei Zhang. Rpn: Reconciled polynomial network towards unifying pgms, kernel svms, mlp and kan. *arXiv preprint arXiv:2407.04819*, 2024.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.

Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265–278, 2016.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

Blaz Zupan, Marko Bohanec, Ivan Bratko, and Janez Demsar. Machine learning by function decomposition. In *ICML*, pp. 421–429. Citeseer, 1997.

A Additional Implementation Details

Building on the transformer architecture, the input initially passes through the attention layer, where the number of attention heads is set to 8. Furthermore, our proposed MLP-KAN replaces the original MLP layer and consists of 8 experts (4 MLP experts and 4 KAN experts), with 2 experts dynamically selected for computation in each forward pass. Subsequently, an additive residual connection is applied before the attention and MLP-KAN layers. We also use the normalization layer to ensure a consistent numerical distribution across different feature dimensions. This improves both the stability during training and the overall performance of the model. We utilized a structure with 12 identical layers. To enhance model generalization, we employ Stochastic Depth (Huang et al., 2016), which randomly drops certain layers during training. The process is as follows:

- **Step 1**: Tokenize the input $\mathbf{X}$ into tokens $\mathbf{X}_i$:

$$\mathbf{X} = [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_m];$$

- **Step 2**: Apply the multi-head self-attention mechanism (MHA) and layer normalization (LN), obtaining:

$$\mathbf{X}' = \mathrm{MHA}\left(\mathrm{LN}(\mathbf{X})\right) + \mathbf{X}$$

- **Step 3**: Continue processing with MLP-KAN to obtain the following results:

$$\mathbf{X}'' = \mathrm{F}(\mathrm{LN}(\mathbf{X}')) + \mathbf{X}'$$

Typically, MLP-KAN, denoted as $\mathrm{F}()$, incorporates a Mixture of Experts (MoE) layer comprising multiple feed-forward networks (FFNs). These FFNs form a pool of experts $[\mathbf{e}_1, \mathbf{e}_2, \ldots]$. In this work, the MLP and KAN experts represent two distinct implementations within the FFN ensemble, together constituting the complete pool of experts. The gating mechanism, functioning as a linear layer, calculates the probability of each input token being assigned to a particular expert. Based on the router's output, the Top-K mechanism most probable experts are selected to process the input, and the outputs of these experts are weighted and summed to form the final result. The final representation is expressed as follows:

$$\alpha_i(\mathbf{X}) = \frac{\mathbf{e}^{g_i(\mathbf{X})}}{\sum_j^E \mathbf{e}^{g_j(\mathbf{X})}},$$

where $g(\mathbf{X}) = \mathbf{W} \cdot \mathbf{X}$ represents the logit produced by the gate, and the weights are normalized via a softmax function to yield the assignment probabilities for each input token across the experts. Through the Top-K operation, K experts with the highest probabilities are selected to process each input token.

Each selected expert processes the input, and the outputs are weighted according to softmax probabilities. These are then aggregated into a weighted sum to produce the final output, which can be described as follows:

$$\mathrm{F}(\mathbf{X}) = \sum_{i=1}^{k} \alpha_i(\mathbf{X}) \cdot \mathbf{e}_i(\mathbf{X}).$$

This mechanism allows each token to be effectively processed by only a few relevant experts, thereby achieving efficient computation and expanding the model's capacity.

B DATASETS

B.1 CIFAR-10 DATASET

The CIFAR-10 dataset is a labeled subset of the 80 million tiny images dataset, containing 60,000 32x32 color images distributed across 10 mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each class contains 6,000 images, and the dataset is divided into 50,000 training images and 10,000 test images. The training images are split into five batches, each consisting of 10,000 images, while the test batch contains 10,000 randomly selected images. The dataset provides a diverse representation of objects, and the classes are non-overlapping; for instance, "automobile" includes small vehicles like sedans and SUVs, while "truck" includes only larger vehicles like big trucks.

Each image is represented by a 1x3072 array of pixel values, where the first 1024 entries correspond to the red channel, the second 1024 to the green channel, and the last 1024 to the blue channel, stored in row-major order. The dataset is widely used for image classification benchmarks, and baseline results using convolutional neural networks have achieved test error rates of 18% without data augmentation and 11% with augmentation. The dataset is commonly accessed in Python, Matlab, or binary formats, with convenient tools for loading and processing the images for machine learning tasks. The structure of the CIFAR10 dataset as shown in Table 6.

Table 6: CIFAR-10 Dataset Structure

Data	Shape	Description
train_x	(50000, 32, 32, 3)	Training Samples
train_y	(50000, 1)	Training Labels
test_x	(10000, 32, 32, 3)	Testing Samples
test_y	(10000, 1)	Testing Labels

B.2 CIFAR-100 DATASET

The CIFAR-100 dataset shares the same general structure as CIFAR-10 but is more granular, containing 100 classes of objects, each represented by 600 images, with 500 training images and 100 test images per class. The dataset introduces a hierarchical structure where the 100 fine-grained classes are grouped into 20 superclasses (coarse labels). For example, the superclass "aquatic mammal" includes beaver, dolphin, otter, seal, and whale, while the superclass "vehicles 1" contains bicycle, bus, motorcycle, pickup truck, and train.

Similar to CIFAR-10, CIFAR-100 images are stored as 1x3072 arrays, with two label bytes for each image: one for the coarse label and one for the fine label. This dataset is often used for fine-grained classification tasks, presenting a more challenging problem due to its increased number of classes and hierarchical structure. Both the CIFAR-10 and CIFAR-100 datasets have been extensively used in the computer vision community for benchmarking the performance of image classification algorithms. The structure of CIFAR-100 as shown in Table 7.

Table 7: Classification Table

Category	**Subcategory**
Aquatic Mammals	Beaver, Dolphin, Otter, Seal, Whale
Fish	Aquarium Fish, Flounder, Ray, Shark, Trout
Flowers	Orchid, Poppy, Rose, Sunflower, Tulip
Food Containers	Bottle, Bowl, Can, Cup, Plate
Fruits and Vegetables	Apple, Mushroom, Orange, Pear, Bell Pepper
Household Appliances	Clock, Computer Keyboard, Lamp, Phone, TV
Household Furniture	Bed, Chair, Sofa, Table, Wardrobe
Insects	Bee, Beetle, Butterfly, Caterpillar, Cockroach
Large Carnivores	Bear, Leopard, Lion, Tiger, Wolf
Large Man-made Outdoor Things	Bridge, Castle, House, Road, Skyscraper
Large Natural Outdoor Scenes	Cloud, Forest, Mountain, Plain, Sea
Large Omnivores and Herbivores	Camel, Cow, Chimpanzee, Elephant, Kangaroo
Medium-sized Mammals	Fox, Porcupine, Opossum, Raccoon, Skunk
Non-insect Invertebrates	Crab, Lobster, Snail, Spider, Worm
People	Baby, Boy, Girl, Man, Woman
Reptiles	Crocodile, Dinosaur, Lizard, Snake, Turtle
Small Mammals	Hamster, Mouse, Rabbit, Shrew, Squirrel
Trees	Maple, Oak, Palm, Pine, Willow
Vehicles	Bicycle, Bus, Motorcycle, Van, Train

B.3 FEYNMAN DATASET

The Feynman dataset is a collection of physics equations sourced from the Feynman Lectures on Physics (Feynman, 1999), designed as a benchmark for symbolic regression tasks. It comprises 120 formulas, primarily drawn from classical physics, including key concepts from mechanics, electromagnetism, and thermodynamics. For our purposes, we focus on the Feynman_no_units subset, specifically equations involving at least two variables, which reduce to one-dimensional splines. An example is the relativistic velocity addition formula, $f(u, v) = \frac{u+v}{1+uv}$, where u and v are sampled from the range (-1, 1), and the network is trained to predict f based on these inputs. The dataset

serves to evaluate the ability of neural networks and other symbolic regression methods to model and predict underlying physical laws from empirical data.

B.4 MINI-INMAGENET DATASET

Mini-Imagenet is a small-scale dataset extracted from the ImageNet dataset by the Google DeepMind team in 2016, primarily used for research in the field of few-shot learning. The total size of the dataset is approximately 3GB and contains 60,000 images divided into 100 classes, with 600 images per class. These images are of varying sizes and are saved in .jpg format.

Compared to the full ImageNet dataset, Mini-Imagenet significantly reduces the data volume, making it more accessible for researchers with limited hardware resources. It is suitable for rapid prototyping and evaluating a model's classification performance, especially in few-shot learning scenarios.

The dataset is structured as follows:

Table 8: Mini-Imagenet Dataset Structure

Directory	Description
`mini-imagenet/`	Root directory of the dataset
`images/`	Folder containing all the images
`train.csv`	Label file for the training set
`val.csv`	Label file for the validation set
`test.csv`	Label file for the test set

It is important to note that when this dataset was created, the labels were not evenly sampled from each class, which adds an additional challenge for models designed for few-shot learning. Researchers can use these CSV files to obtain image labels and perform training, validation, and testing.

B.5 SST-2 DATASET

The Stanford Sentiment Treebank (SST) is a linguistically annotated dataset designed to enable detailed analysis of sentiment composition in natural language. Derived from movie reviews, this dataset includes 11,855 individual sentences, which were parsed into syntactic structures using the Stanford parser. The resulting parse trees consist of 215,154 unique phrases, all annotated by human judges to capture nuanced sentiment at various granularities.

A distinctive feature of the SST dataset is its ability to support research on compositional sentiment analysis, as each sub-phrase in a sentence is independently labeled for sentiment. This allows for a deeper understanding of how sentiment is constructed and expressed through the combination of linguistic elements.

In the context of binary sentiment classification tasks, a simplified version of the dataset, known as SST-2, is often used. In SST-2, neutral sentences are excluded, and the remaining sentences are categorized into either negative or positive classes. This binary classification setup has become a widely adopted benchmark for evaluating sentiment analysis models.

C ADDITIONAL RESULTS

C.1 ATTENTION MECHANISM VISUALIZATION

In this section, we present a comparative analysis of attention mechanisms across different models on CIFAR-100. As shown in Figure 4, the attention maps of our proposed MLP-KAN approach are visually comparable to those generated by MLP, which achieves the best performance on CIFAR-100. This indicates that our method effectively captures critical features similar to the most successful architecture. In contrast, the KAN model exhibits distinct attention patterns and performs poorly on CIFAR-100, likely due to its limitations in handling image-based tasks.

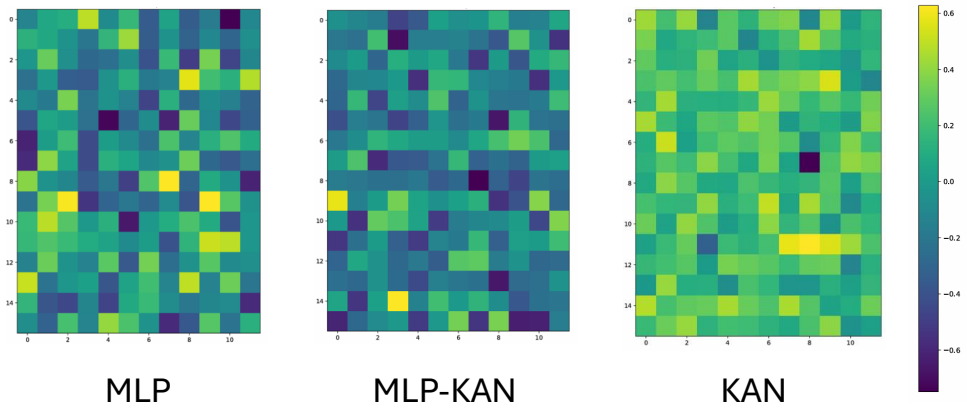

Figure 4: Visualization of attention mechanisms for the first batch of CIFAR-100. The attention maps are generated using MLP, MLP-KAN, and KAN models, showcasing distinct patterns and feature focuses across the different architectures.

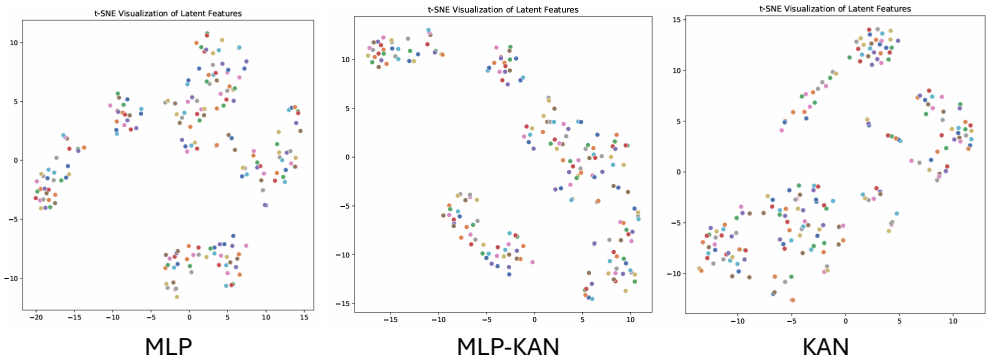

Figure 5: t-SNE visualizations of latent features extracted by MLP, MLP-KAN, and KAN models, showcasing the distinct clustering patterns and feature separability achieved by each approach.

Overall, the results demonstrate that MLP-KAN not only aligns closely with the attention dynamics of the best-performing model (MLP) but also surpasses KAN in adapting to the characteristics of CIFAR-100, highlighting its effectiveness and adaptability for this dataset.

C.2 Latent Feature Visualization

In this subsection, we analyze the quality of latent feature representations learned by MLP, MLP-KAN, and KAN models through t-SNE visualizations, as shown in Figure 5. These visualizations provide insights into how well the models capture meaningful structure in the latent space during representation learning on the CIFAR-100 dataset.

As illustrated, the MLP model shows relatively scattered clusters with weaker separability, indicating limited ability to encode meaningful and distinct latent representations. In contrast, our proposed MLP-KAN model demonstrates significantly improved clustering patterns, with more compact and well-separated groups of features. This suggests that the combination of MLPs for representation learning and KANs for functional learning synergistically enhances the model's ability to learn structured and meaningful latent features.

These results confirm that MLP-KAN not only improves task performance but also enables the learning of semantically meaningful latent features, aligning closely with the underlying data structure.

This demonstrates the potential of MLP-KAN in bridging representation and functional learning for real-world applications.