

Low Tensor-Rank Adaptation of Kolmogorov–Arnold Networks

Yihang Gao, Michael K. Ng, Vincent Y.F. Tan

Abstract—Kolmogorov–Arnold networks (KANs) have demonstrated their potential as an alternative to multi-layer perceptrons (MLPs) in various domains, especially for science-related tasks. However, transfer learning of KANs remains a relatively unexplored area. In this paper, inspired by Tucker decomposition of tensors and evidence on the low tensor-rank structure in KAN parameter updates, we develop low tensor-rank adaptation (LoTRA) for fine-tuning KANs. We study the expressiveness of LoTRA based on Tucker decomposition approximations. Furthermore, we provide a theoretical analysis to select the learning rates for each LoTRA component to enable efficient training. Our analysis also shows that using identical learning rates across all components leads to inefficient training, highlighting the need for an adaptive learning rate strategy. Beyond theoretical insights, we explore the application of LoTRA for efficiently solving various partial differential equations (PDEs) by fine-tuning KANs. Additionally, we propose Slim KANs that incorporate the inherent low-tensor-rank properties of KAN parameter tensors to reduce model size while maintaining superior performance. Experimental results validate the efficacy of the proposed learning rate selection strategy and demonstrate the effectiveness of LoTRA for transfer learning of KANs in solving PDEs. Further evaluations on Slim KANs for function representation and image classification tasks highlight the expressiveness of LoTRA and the potential for parameter reduction through low tensor-rank decomposition.

Index Terms—Low tensor-rank adaptation, transfer learning, fine-tuning, Kolmogorov–Arnold networks, physics-informed machine learning, partial differential equations.

I. INTRODUCTION

The Kolmogorov–Arnold representation theorem (KART) states that any multivariate continuous function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ can be decomposed into a sum of univariate functions. Specifically, there exists a set of univariate functions $\{\phi_{p,q}\}_{p=1,q=1}^{n,2n+1}$ and $\{\Phi_q\}_{q=1}^{2n+1}$ such that

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

where $\mathbf{x} = [x_1, x_2, \dots, x_n]^\top$. Inspired by this theorem, researchers have explored novel network architectures, as an alternative to multi-layer perceptrons (MLPs) guaranteed by the Universal Approximability Theory (UAT). However, earlier network designs based on KART have been underwhelming, mainly because they are inherently limited to two layers and a width restricted to at most twice the input dimension, as prescribed by Equation (1). These models typically represent the implicit univariate functions $\{\phi_{p,q}\}_{p=1,q=1}^{n,2n+1}$ and $\{\Phi_q\}_{q=1}^{2n+1}$ using polynomials or splines. Although theoretically sound, such models often fail to perform competitively in practical

applications, lagging behind MLPs. Recently, Liu et al. [1] proposed a KART-based model known as Kolmogorov–Arnold Networks (KANs), which extends the architecture to support multiple layers and arbitrary width. This design overcomes the stringent limitations on width and depth inherent in traditional KART-based networks, enabling greater expressiveness and flexibility.

KANs have demonstrated remarkable empirical performance across a wide range of domains, including computer vision, time series forecasting, reinforcement learning, physics-informed machine learning, and large language models. Li et al. [2] developed U-KAN, which replaces the bottleneck layers of U-Net with KAN layers, achieving higher accuracy with reduced computational cost in medical image segmentation and generation tasks. This improvement is attributed to the potentially higher expressiveness of KANs compared to MLPs. Kich et al. [3] explored the application of KANs as function approximators in proximal policy optimization for reinforcement learning, demonstrating that KANs can match the performance of MLPs while requiring fewer parameters. Notably, KANs excel in symbolic and function representation tasks. In physics-informed machine learning, KANs consistently match or outperform MLPs, especially in approximating solutions of partial differential equations (PDEs), as reported by Wang et al. [4]. To model the separable properties inherent in some PDE solutions, a separable physics-informed KAN architecture was proposed [5]. This design processes each block independently using individual KAN models before synthesizing them in the final stage, significantly reducing model complexity by incorporating the natural separable structure of PDE solutions. More interesting applications and investigations of KANs can be found in [6]–[17].

As illustrated above, one of the most significant advantages of KANs over MLPs lies in their superior performance in science-related tasks, particularly in physics-informed machine learning. Here, we mainly focus on using KANs to approximate solutions of PDEs. However, in practice, even slight variations in physical parameters or conditions can result in changes to the solutions. Instead of solving a single PDE and storing the corresponding network, it is often necessary to solve a class of PDEs with varying physical parameters. This setting requires solving PDEs multiple times and storing numerous networks, which significantly increases computational costs and storage requirements.

However, the transfer learning of KANs has not been investigated, despite its potential importance in enhancing the efficiency of model training and storage, particularly in physics-informed machine learning applications. Inspired by the recently popular low-rank adaptation (LoRA) technique used in Transformer models, which significantly accelerates the fine-tuning process for large language models, and the

Department of Mathematics, National University of Singapore

Department of Mathematics, Hong Kong Baptist University

Department of Mathematics and Department of Electrical and Computer Engineering, National University of Singapore

evidence on low tensor-rank structure of parameter updates of KANs, we introduce the low tensor-rank adaptation (LoTRA) for the tensor parameters of KANs. In this approach, we first pre-train the KAN model on a given task with full tensor parameters (denoted as \mathcal{A}) being updated. For a new task, instead of updating the entire tensor parameter \mathcal{A} , we apply the Tucker decomposition to the tensor, adapting it as $\mathcal{A} + \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}$. Here, $\{\mathcal{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ are trainable parameters, providing a low tensor-rank adaptation for the tensor parameters of KANs. In this adaptation, \mathcal{G} is the core tensor, and $\{\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ are transformation matrices. If the core tensor \mathcal{G} is significantly smaller than \mathcal{A} , then the total parameter size of $\{\mathcal{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ is much smaller than that of \mathcal{A} , resulting in efficient fine-tuning and reduced storage. In this framework, the original tensor \mathcal{A} captures and retains the shared information across tasks from the pre-training stage, while $\{\mathcal{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ further adapt the model to task-specific characteristics in new tasks. This is the first work studying the transfer learning of KANs through low tensor rank adaptation, paving the way for more efficient and scalable applications. Our contributions are summarized as follows:

- (i) We introduce the low tensor-rank adaptation (LoTRA) for the transfer learning of KANs, using Tucker decomposition to achieve efficient and effective adaptation.
- (ii) We theoretically show the expressiveness of LoTRA and analyze the efficient training of KANs with LoTRA. Our theoretical results offer a learning rate selection strategy to fine-tune KANs with LoTRA efficiently.
- (iii) The proposed LoTRA method adapts well to physics-informed machine learning with KANs, enabling efficient training and significantly reduced storage for solving a class of PDEs. Additionally, the LoTRA framework motivates the development of Slim KANs, which adopt a low tensor-rank structure in parameter tensors to reduce model size while preserving performance.
- (iv) We conduct comprehensive experiments on solving a class of PDEs with varying parameters and solutions, validating the developed learning rate selection strategy and showing the effectiveness of LoTRA in fine-tuning KANs. Additional experiments on Slim KANs further demonstrate the expressiveness of LoTRA and the potential of integrating parameter-efficient models into advanced architectures for broader applications.

II. BACKGROUND AND PRELIMINARIES

In this section, we first introduce the notation used throughout the paper. We then review the mathematical definition and key concepts of KANs. Finally, we discuss the mathematical formulation of Tucker decomposition for tensors, which serves as a foundation for the methodology section.

A. Notation

In this paper, we use bold lowercase letters (e.g., \mathbf{x}), bold capital letters (e.g., \mathbf{A}), and calligraphic uppercase letters (e.g., \mathcal{A}) to denote vectors, matrices and tensors, respectively. Scalars are represented using regular (non-bold) letters (e.g.,

a). The reshape operation, denoted as $\text{reshape}(\mathcal{A}; i; a, b)$, restructures the tensor \mathcal{A} along mode i , transforming it into an $a \times b$ matrix. We use $[L]$ to denote the set $\{1, 2, \dots, L\}$ for positive integers L .

B. Kolmogorov–Arnold Networks

As defined in [1], the $(\ell + 1)$ -st layer $\mathbf{z}_{\ell+1} = [z_{\ell+1,1}, \dots, z_{\ell+1,n_{\ell+1}}]^\top$ of a KAN admits

$$z_{\ell+1,q} = \sum_{p=1}^{n_\ell} \phi_{\ell,p,q}(z_{\ell,p}), \quad q \in [n_{\ell+1}], \quad (2)$$

where n_ℓ is the width of the ℓ -th layer and $\phi_{\ell,p,q}$ denotes the (p, q) -th representation function in the ℓ -th layer. This formulation is consistent with the KART presented in Equation (1), but it introduces some key differences. Notably, KANs do not impose restrictions on width (n_ℓ can exceed twice the input dimension n) or depth (e.g., ℓ can be much greater than 2). However, the implicit representation functions $\{\phi_{\ell,p,q}\}_{\ell,p,q}$ in each layer are unknown, and must be approximated using practical function classes. In this sense, the development of KANs draws inspiration both from the theoretical principles of KART and the flexibility of MLPs.

In practice, the implicit representation functions are expressed by combinations of basis functions, i.e.,

$$\phi_{\ell,p,q}(z) = \sum_{k=1}^{n_d} a_{\ell,p,q,k} b_k(z),$$

where $\{b_k\}_{k \in [n_d]}$ is a set of basis functions and $\mathcal{A}^\ell := (a_{\ell,p,q,k}) \in \mathbb{R}^{n_\ell \times n_{\ell+1} \times n_d}$ denotes the parameter tensor at ℓ -th layer. The basis functions adopted can include Chebyshev polynomials [18], Legendre polynomials [19], Fourier series [20], wavelet functions [21], Bernoulli polynomials, Fibonacci polynomials, Jacobi polynomials [22], B-splines [1], as well as rational and fractional polynomials [23], [24], due to the universal approximability of those basis functions.

In summary, the transformation of KANs from ℓ -th layer to $(\ell + 1)$ -th layer is formulated as

$$z_{\ell+1,q} = \sum_{p=1}^{n_\ell} \sum_{k=1}^{n_d} a_{\ell,p,q,k} b_k(z_{\ell,p}), \quad q \in [n_{\ell+1}], \quad (3)$$

where $\mathbf{z}_\ell = (z_{\ell,1}, z_{\ell,2}, \dots, z_{\ell,n_\ell})^\top \in \mathbb{R}^{n_\ell}$ and $\mathbf{z}_{\ell+1} = (z_{\ell+1,1}, z_{\ell+1,2}, \dots, z_{\ell+1,n_{\ell+1}})^\top \in \mathbb{R}^{n_{\ell+1}}$ denote the neurons at the ℓ -th and $(\ell + 1)$ -th layers, respectively, and $\mathcal{A}^\ell := (a_{\ell,p,q,k}) \in \mathbb{R}^{n_\ell \times n_{\ell+1} \times n_d}$ represents the parameter tensor at ℓ -th layer.

C. Tucker Decomposition

Tensor decomposition methods have shown significant success in various domains, such as hyperspectral image processing [25]–[27], multidimensional time series analysis [28], [29], and high-dimensional machine learning [30], [31]. Tucker decomposition method [32] compresses the tensor into a smaller core tensor with transformation matrices applied along each mode, whose structure is similar to the singular value decomposition (SVD) of matrices. Other tensor decomposition methods include CANDECOMP/PARAFAC (CP) method [33],

tensor SVD [34], [35], CANDECOMP with linear constraints (CANDELINC) [36], and parallel factors for cross products (PARAFAC2) [37], among others.

In this paper, we make use of the Tucker decomposition of tensors. As shown in Equation (3), the parameters of KANs for each layer are represented by a third-order tensor, whereas the parameters of MLPs are matrices. Tucker decomposition is considered a generalization of matrix SVD to tensors. For a given third-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, Tucker decomposition expresses \mathcal{A} as a core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, and three transformation matrices applied to each mode, $\mathbf{U}^{(1)} \in \mathbb{R}^{n_1 \times r_1}$, $\mathbf{U}^{(2)} \in \mathbb{R}^{n_2 \times r_2}$ and $\mathbf{U}^{(3)} \in \mathbb{R}^{n_3 \times r_3}$, such that

$$\mathcal{A} = \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \mathbf{U}^{(3)}, \quad (4)$$

where \times_i denotes the mode- i product of the core tensor \mathcal{G} with the transformation matrices $\mathbf{U}^{(i)}$ (for $i = 1, 2, 3$). If the tensor \mathcal{A} has a low Tucker rank, then $r_i < n_i$ (for $i = 1, 2, 3$). This decomposition closely parallels matrix SVD, where the core tensor \mathcal{G} plays a role analogous to the singular values, while $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$ and $\mathbf{U}^{(3)}$ serve as basis matrices. The essential information of the tensor is compressed into the core tensor \mathcal{G} and can be fully reconstructed using the appropriate transformation matrices. If the tensor \mathcal{A} has an extremely low Tucker rank (e.g., $r_i \ll n_i$, for $i = 1, 2, 3$), then the parameter size of $\{\mathcal{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ is much smaller than that of the original tensor, leading to the effective compression by Tucker decomposition.

The higher-order singular value decomposition (HOSVD) method [38] provides a numerical approach to determine the Tucker decomposition for tensors. Let $\mathbf{A}^{(i)}$ denote the mode- i unfolding (or matricization) of the tensor \mathcal{A} , defined as

$$\mathbf{A}^{(i)} = \text{reshape} \left(\mathcal{A}; i, n_i, \frac{n_1 n_2 n_3}{n_i} \right).$$

In this unfolding, the mode- i fibers of \mathcal{A} are arranged as columns in $\mathbf{A}^{(i)}$. This operation reorganizes the tensor by flattening all dimensions except the i -th mode. The transformation matrices $\{\mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ are then obtained by performing SVDs on all mode- i unfolding matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$, and $\mathbf{A}^{(3)}$. If any of the unfolding matrices contain zero singular values, the core tensor \mathcal{G} has a smaller size than the original tensor \mathcal{A} , indicating the presence of a low-rank structure in \mathcal{A} .

III. METHODS

In this section, we first provide evidence supporting the potential low tensor-rank structure of the parameter updates, which motivates the development of the low tensor-rank adaptation (LoTRA) for transfer learning of KANs. We then theoretically analyze the approximation and expressiveness capabilities of LoTRA, using well-established results on Tucker decomposition. Furthermore, we propose a theoretically guided strategy for learning rate selection of each component of LoTRA to achieve efficient training using gradient descent. In contrast, we demonstrate theoretically that using the same learning rate scale for all components is inefficient for LoTRA.

A. Motivation and Evidence

The concept of low-rank adaptation is widely recognized in transfer learning and domain adaptation. In multi-task learning, the parameter matrix is typically decomposed into global patterns shared across tasks and task-specific adaptations within the shared subspace [39], [40]. This decomposition captures shared information while regularizing models to prevent overfitting. Similarly, in domain adaptation, both source and target domains are projected into a shared feature space through low-rank mappings, reflecting the existence of underlying shared low-rank subspaces [41], [42]. In the fine-tuning of large language models, Hu et al. [43] empirically demonstrated that weight updates during adaptation exhibit low intrinsic matrix rank. This observation led to the development of low-rank adaptation (LoRA) for Transformer models.

Based on their observations, we have reason to hypothesize that the update of KANs parameters has a low tensor-rank structure in transferring learning tasks, specifically, fine-tuning on new tasks based on the pre-trained models. The potential intrinsic low tensor-rank property has been seen in some empirical investigations on KANs. For example, function mappings in science domains usually involve specific structures that may lead to similarities between implicit representation functions $\{\phi_{\ell,p,q}\}$ in Equation (2). Specifically, the symbolic representation of the function $f(x_1, x_2, x_3, x_4) = \exp(\frac{1}{2}(\sin(\pi(x_1^2 + x_2^2)) + \sin(\pi(x_3^2 + x_4^2))))$ is discussed in Liu et al. [1], where a three-layer KAN fits the function well. However, the first layer involves mostly the square operation applied to all coordinates and the second layer only represents the sin function. The final layer produces the outputs after passing through the exponential function. This example shows the repeated operations in each layer, implying the possibility of compression of parameter tensors of KANs. Similar properties have been observed in the solutions of PDEs studied in physics-informed KANs, as shown in Shukla et al. [6] and Wang et al. [4], further supporting the possibility of low-rank structures in KAN parameter tensors, particularly in symbolic representation and physics-informed machine learning.

To further validate our hypothesis regarding the low-rank updates of KAN parameter tensors, we conduct a simple transfer learning experiment on function representation. We use a three-layer KAN with a hidden dimensions $n_1 = n_2 = 32$ and the number of basis functions $n_d = 32$ to represent and pre-train on the function $u(x_1, x_2) = \sin(\frac{\pi}{2}(1 - \sqrt{x_1^2 + x_2^2})^{2.5})$. We then fine-tune the KAN model on a new task with the modified objective function: $u(x_1, x_2) = \sin(\frac{\pi}{2}(1 - \sqrt{x_1^2 + x_2^2})^{2.5}) + \sin(\frac{\pi}{2}(1 - \sqrt{x_1^2 + x_2^2}))$. Our goal is to empirically investigate whether the parameter tensor updates in the fine-tuned model exhibit a low tensor-rank structure. We measure the tensor rank using the Tucker rank and determine it using HOSVD, where the singular values of the mode-unfolded matrices correspond to the Tucker rank of the tensors. Specifically, zero singular values in the unfolded matrices indicate a low Tucker rank. Our observations reveal that KANs at the pre-training stage already exhibit a low Tucker rank structure, where most of the singular values are small, as visualized in Figure 1(a). Furthermore, we compute the

parameter tensor updates during fine-tuning relative to the pre-trained model and parameters. As shown in Figure 1(b), the singular values confirm the low Tucker rank structure in the parameter tensor updates.

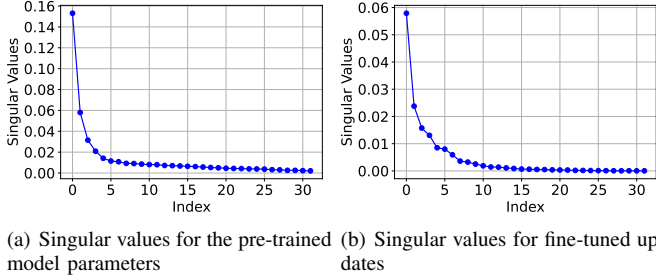


Fig. 1. Singular values for the pre-trained model parameters and fine-tuned updates, demonstrating the low Tucker rank structure.

B. Low Tensor-Rank Adaptation

Building on observations of low-rank structures in previous studies across various domains, as well as the empirical investigations of a simple transfer learning example, we develop an efficient and effective transfer learning method for fine-tuning KANs on new tasks, given a pre-trained model. The developed low tensor-rank adaptation (LoTRA) method preserves shared information across tasks and adapts to new tasks with integrated low-rank structures on parameter tensor updates. For simplicity, we express Equation (3) as

$$\mathbf{z}_{\ell+1} := \Phi_{\ell}(\mathbf{z}_{\ell}; \mathcal{A}_{\ell}),$$

where $\mathcal{A}_{\ell} \in \mathbb{R}^{n_{\ell} \times n_{\ell+1} \times n_d}$ is a third-order parameter tensor, $\ell \in [L]$, and L denotes the depth of the KAN. Therefore, the forward propagation of the KAN model can be formulated as

$$\begin{aligned} \Psi(\mathbf{x}; \{\mathcal{A}_{\ell}\}_{\ell \in [L]}) &:= \Phi_L(\Phi_{L-1} \dots (\Phi_1(\mathbf{x}; \mathcal{A}_1); \mathcal{A}_{L-1}) \mathcal{A}_L) \\ &= \Phi_L \circ \Phi_{L-1} \circ \dots \circ \Phi_1(\mathbf{x}; \{\mathcal{A}_{\ell}\}_{\ell \in [L]}). \end{aligned} \quad (5)$$

Suppose we first pre-train the KAN model on a given task, where the pre-trained model is denoted as $\Psi_{\text{pt}}(\mathbf{x}; \{\mathcal{A}_{\ell, \text{pt}}\})$, and $\{\mathcal{A}_{\ell, \text{pt}}\}$ represents the set of parameter tensors across layers. For a new task, the fine-tuned model is denoted as $\Psi_{\text{ft}}(\mathbf{x}; \{\mathcal{A}_{\ell, \text{ft}}\})$, where $\{\mathcal{A}_{\ell, \text{ft}}\}$ is the updated set of parameter tensors. We define the target model, which represents the optimal function within the KAN function class, as $\Psi_{\text{tg}}(\mathbf{x}; \{\mathcal{A}_{\ell, \text{tg}}\})$, where $\{\mathcal{A}_{\ell, \text{tg}}\}$ denotes the corresponding set of target parameter tensors. The parameter tensor $\mathcal{A}_{\ell, \text{ft}}$ of the fine-tuned model Ψ_{ft} updated using LoTRA follows:

$$\mathcal{A}_{\ell, \text{ft}} = \mathcal{A}_{\ell, \text{pt}} + \mathcal{G}_{\ell} \times_1 \mathbf{U}_{\ell}^{(1)} \times_2 \mathbf{U}_{\ell}^{(2)} \times \mathbf{U}_{\ell}^{(3)}, \quad (6)$$

where $\mathcal{G}_{\ell} \in \mathbb{R}^{r_{\ell,1} \times r_{\ell,2} \times r_{\ell,3}}$ denotes the core tensor, and $\mathbf{U}_{\ell}^{(1)} \in \mathbb{R}^{n_{\ell} \times r_{\ell,1}}$, $\mathbf{U}_{\ell}^{(2)} \in \mathbb{R}^{n_{\ell+1} \times r_{\ell,2}}$ and $\mathbf{U}_{\ell}^{(3)} \in \mathbb{R}^{n_d \times r_{\ell,3}}$ represent transformation matrices. Here, we require that $r_{\ell,1} \leq n_{\ell}$, $r_{\ell,2} \leq n_{\ell+1}$, and $r_{\ell,3} \leq n_d$, where $(r_{\ell,1}, r_{\ell,2}, r_{\ell,3})$ is the core tensor size for ℓ -th layer. Since the Tucker rank of the updates $(r_{\ell,1}, r_{\ell,2}, r_{\ell,3})$ is smaller than the original parameter size

$(n_{\ell}, n_{\ell+1}, n_d)$, LoTRA effectively reduces the model complexity with integrated low-rank structures on tensor updates while preserving essential task-specific information in $\mathcal{A}_{\ell, \text{ft}}$.

We first pre-train the model Ψ_{pt} on a given task, and obtain the corresponding parameter tensors $\{\mathcal{A}_{\ell, \text{pt}}\}_{\ell \in [L]}$. When solving a new task that shares similarities with the pre-training task, we aim to transfer shared information to the new task. To achieve this, we fine-tune the model using LoTRA, resulting in the fine-tuned model Ψ_{ft} with updated parameter tensors $\{\mathcal{A}_{\ell, \text{ft}}\}_{\ell \in [L]}$, following the update rule in Equation (6). In this adaptation, the shared information is retained in the pre-trained tensors $\{\mathcal{A}_{\ell, \text{pt}}\}_{\ell \in [L]}$ while the task-specific information is incorporated through the trainable parameters $\{\mathcal{G}_{\ell}, \mathbf{U}_{\ell}^{(1)}, \mathbf{U}_{\ell}^{(2)}, \mathbf{U}_{\ell}^{(3)}\}_{\ell \in [L]}$, which are the only parameters updated during fine-tuning in Ψ_{ft} . By using this adaptation approach, LoTRA eliminates the need to re-learn shared information from data, thereby significantly improving training efficiency compared to training a new model from scratch. Moreover, the task-specific information is effectively extracted, maintaining model expressiveness and performances on new tasks. Moreover, instead of fine-tuning all parameters on the new task, the explicitly introduced low-rank structure in LoTRA acts as a regularization, mitigating overfitting and filtering out noise arising from limited training data.

IV. THEORETICAL ANALYSIS

In this section, we study the theoretical expressiveness of LoTRA within the framework of low Tucker-rank approximations. To enable efficient fine-tuning using LoTRA with gradient descent, we propose a theoretically grounded learning rate selection strategy. Moreover, we prove that assigning identical learning rates to all trainable parameters is inefficient for training. These theoretical results not only enhance our understanding of LoTRA's expressiveness and capacity but also provide a deeper insight into the training process.

A. Expressiveness of LoTRA

We define the error tensor between the target parameter tensor and the pre-trained parameter tensor as:

$$\mathcal{E}_{\ell} = \mathcal{A}_{\ell, \text{tg}} - \mathcal{A}_{\ell, \text{pt}}.$$

Let $\mathbf{E}_{\ell}^{(1)} \in \mathbb{R}^{n_{\ell} \times (n_{\ell+1} n_d)}$, $\mathbf{E}_{\ell}^{(2)} \in \mathbb{R}^{n_{\ell+1} \times (n_{\ell} n_d)}$, and $\mathbf{E}_{\ell}^{(3)} \in \mathbb{R}^{n_d \times (n_{\ell} n_{\ell+1})}$ denote the corresponding mode- i unfolding matrices of \mathcal{E}_{ℓ} . The following lemma on the Tucker approximation of tensors plays a crucial role in characterizing the approximation capability of LoTRA, which integrates low Tucker-rank adaptation to parameter tensor updates.

Lemma 1 (Tucker Approximation [38]). *For each $\ell \in [L]$, there exist $(\mathcal{G}_{\ell}, \mathbf{U}_{\ell}^{(1)}, \mathbf{U}_{\ell}^{(2)}, \mathbf{U}_{\ell}^{(3)})$, such that*

$$\begin{aligned} \|\mathcal{A}_{\ell, \text{ft}} - \mathcal{A}_{\ell, \text{tg}}\|_F^2 &\leq \sum_{r=r_{\ell,1}+1}^{n_{\ell}} \sigma_r(\mathbf{E}_{\ell}^{(1)})^2 \\ &\quad + \sum_{r=r_{\ell,2}+1}^{n_{\ell+1}} \sigma_r(\mathbf{E}_{\ell}^{(2)})^2 + \sum_{r=r_{\ell,3}+1}^{n_d} \sigma_r(\mathbf{E}_{\ell}^{(3)})^2, \end{aligned}$$

where $\sigma_r(\cdot)$ denotes the r -th largest singular value of the given matrix, and $\mathcal{A}_{\ell,\text{ft}}$ follows Equation (6).

Assumption 1. Assume that the basis functions $b_k(\cdot)$ are uniformly bounded and smooth, with Lipschitz constant $L > 0$ and the bound $B > 0$. Moreover, the parameter tensors of both the target and pre-trained models are bounded by $M > 0$, i.e., $\|\mathcal{A}_{\ell,\text{tg}}\|_F \leq M$ and $\|\mathcal{A}_{\ell,\text{pt}}\|_F \leq M$, for $\ell \in [L]$.

The above assumption is mild and practically realizable for KAN models. The conditions of uniform boundedness and smoothness hold in common settings. For example, in KANs with B-spline basis functions, the input is restricted to a bounded domain, ensuring that the conditions are naturally satisfied [1]. Similarly, for KANs using polynomial basis functions, such as Chebyshev and Legendre polynomials, the domain is usually normalized to $[-1, 1]$ through an additional activation, such as the hyperbolic tangent function [18]. Therefore, the uniform boundedness and Lipschitz continuity conditions hold in practice. Based on Lemma 1 and Assumption 1, we establish the approximation capability of LoTRA, particularly its ability to approximate the target function Ψ_{tg} using the fine-tuned model Ψ_{ft} .

Theorem 1. Under Assumption 1, there exists a fine-tuned model Ψ_{ft} with LoTRA model that satisfies

$$\begin{aligned} & \|\Psi_{\text{ft}}(\mathbf{x}) - \Psi_{\text{tg}}(\mathbf{x})\|_2 \\ & \leq \sum_{\ell=1}^L C_\ell \cdot \left(\sum_{r=r_{\ell,1}+1}^{n_\ell} \sigma_r(\mathbf{E}_\ell^{(1)})^2 + \sum_{r=r_{\ell,2}+1}^{n_{\ell+1}} \sigma_r(\mathbf{E}_\ell^{(2)})^2 \right. \\ & \quad \left. + \sum_{r=r_{\ell,3}+1}^{n_d} \sigma_r(\mathbf{E}_\ell^{(3)})^2 \right)^{1/2}, \end{aligned}$$

for any $\mathbf{x} \in \mathbb{R}^n$, where the constant $C_\ell := (ML\sqrt{n_d})^{L-\ell} \cdot B\sqrt{n_\ell n_d}$ depends only on the constants in Assumption 1 and the model size.

The proof can be found in Appendix A. Theorem 1 indicates that when some of the singular values of the error tensor \mathcal{E}_ℓ between the pre-trained and target models are negligible, the fine-tuned model with LoTRA is capable of accurately approximating the target model. The quality of this approximation depends on the magnitude of the discarded singular values.

B. Efficient Training of LoTRA

Besides the approximation capability of the fine-tuned model with LoTRA, optimization efficiency with gradient descent is also a key focus and a critical consideration for real-world applications. For analysis convenience, we consider fine-tuning the model on a toy case, where we are given one training data (\mathbf{x}, \mathbf{y}) of a new task and the loss function is formulated as $\mathcal{L} = \frac{1}{2} \|\Psi_{\text{ft}}(\mathbf{x}) - \mathbf{y}\|_2^2$ with $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$. We first consider a one-layer model and the adaptation with core tensor size (r_1, r_2, r_3) . Suppose the pre-trained model is $\Psi_{\text{pt}}(\mathbf{x}; \mathcal{A}_{\text{pt}})$ and the fine-tuned model $\Psi_{\text{ft}}(\mathbf{x}; \mathcal{A}_{\text{ft}})$ satisfies $\mathcal{A}_{\text{ft}} = \mathcal{A}_{\text{pt}} + \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(1)} \times_3 \mathbf{U}^{(3)}$, where $\mathcal{A}_{\text{pt}} \in \mathbb{R}^{n \times m \times n_d}$, $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, $\mathbf{U}^{(1)} \in \mathbb{R}^{n \times r_1}$, $\mathbf{U}^{(2)} \in \mathbb{R}^{m \times r_2}$, and $\mathbf{U}^{(3)} \in \mathbb{R}^{n_d \times r_3}$. Here, we assume that (r_1, r_2, r_3)

is fixed and much smaller than (n, m, n_d) . The pre-trained parameter tensor \mathcal{A}_{pt} remains fixed and $\{\mathcal{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ are trainable. We denote the (p, q, k) -th element of the tensor \mathcal{G} as $g^{p,q,k}$ and the j -th column of $\mathbf{U}^{(i)} = [\mathbf{u}^{(i),1}, \dots, \mathbf{u}^{(i),r_i}]$ is denoted as $\mathbf{u}^{(i),j}$. All trainable parameters are optimized by the vanilla gradient descent, following the updates:

$$g_t^{p,q,k} = g_{t-1}^{p,q,k} - \eta_0 \cdot \frac{\partial \mathcal{L}_{t-1}}{\partial g^{p,q,k}}, \quad \mathbf{u}_t^{(i),j} = \mathbf{u}_{t-1}^{(i),j} - \eta_i \cdot \frac{\partial \mathcal{L}_{t-1}}{\partial \mathbf{u}^{(i),j}}, \quad (7)$$

for $p \in [n]$, $q \in [m]$, $k \in [n_d]$, $j \in [r_i]$, and $i \in \{1, 2, 3\}$. Here, the subscript t denotes the parameters after t steps of gradient descent, and $(\eta_0, \eta_1, \eta_2, \eta_3)$ are learning rates for $\{\mathcal{G}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}\}$ respectively. Similarly, we denote the fine-tuned model Ψ_{ft} at step t as $\Psi_{t,\text{ft}}$ with parameters $\{\mathcal{G}_t, \mathbf{U}_t^{(1)}, \mathbf{U}_t^{(2)}, \mathbf{U}_t^{(3)}\}$, and the corresponding loss function is denoted as \mathcal{L}_t .

Denote $\mathbf{X} \in \mathbb{R}^{n \times n_d}$, where the k -th column of \mathbf{X} is obtained by applying the basis function b_k elementwise to the input column vector \mathbf{x} . Then, the change in function values after one step of gradient descent satisfies

$$\begin{aligned} \Delta \Psi_{t,\text{ft}} &:= \Psi_{t,\text{ft}} - \Psi_{t-1,\text{ft}} \\ &\approx \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} \delta_{t,0}^{p,q,k} + \sum_{p=1}^{r_1} \delta_{t,1}^p + \sum_{q=1}^{r_2} \delta_{t,2}^q + \sum_{k=1}^{r_3} \delta_{t,3}^k, \end{aligned}$$

where

$$\begin{aligned} \delta_{t,0}^{p,q,k} &= -\eta_0 \cdot \left(\mathbf{v}_{t-1}^\top \mathbf{u}_{t-1}^{(2),q} \right) \cdot \left(\mathbf{u}_{t-1}^{(1),p^\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k} \right)^2 \cdot \mathbf{u}_{t-1}^{(2),q}, \\ \delta_{t,1}^p &= -\eta_1 \cdot \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} g_{t-1}^{p,q,k} \left(\sum_{q'=1}^{r_2} \sum_{k'=1}^{r_3} g_{t-1}^{p,q',k'} \left(\mathbf{v}_{t-1}^\top \mathbf{u}_{t-1}^{(2),q'} \right) \right. \\ &\quad \left. \cdot \left(\mathbf{u}_{t-1}^{(3),k'^\top} \mathbf{X}^\top \mathbf{X} \mathbf{u}_{t-1}^{(3),k} \right) \right) \cdot \mathbf{u}_{t-1}^{(2),q}, \\ \delta_{t,2}^q &= -\eta_2 \cdot \sum_{p=1}^{r_1} \sum_{k=1}^{r_3} g_{t-1}^{p,q,k} \left(\sum_{p'=1}^{r_1} \sum_{k'=1}^{r_3} g_{t-1}^{p',q,k'} \left(\mathbf{u}_{t-1}^{(1),p'^\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k'} \right) \right) \\ &\quad \cdot \left(\mathbf{u}_{t-1}^{(1),p^\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k} \right) \cdot \mathbf{v}_{t-1}, \\ \delta_{t,3}^k &= -\eta_3 \cdot \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} g_{t-1}^{p,q,k} \left(\sum_{p'=1}^{r_1} \sum_{q'=1}^{r_2} g_{t-1}^{p',q',k} \left(\mathbf{v}_{t-1}^\top \mathbf{u}_{t-1}^{(2),q'} \right) \right. \\ &\quad \left. \cdot \left(\mathbf{u}_{t-1}^{(1),p^\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_{t-1}^{(1),p'} \right) \right) \cdot \mathbf{u}_{t-1}^{(2),q}, \end{aligned}$$

where $\mathbf{v}_t := \frac{\partial \mathcal{L}_t}{\partial \Psi_{\text{ft}}} = \Psi_{t,\text{ft}}(\mathbf{x}) - \mathbf{y}$ denotes the partial derivative of the loss with respect to the output of the fine-tuned model. The detailed derivation of the linearization of $\Delta \Psi_{t,\text{ft}}$ can be found in Appendix B. Here, $\{\delta_{t,0}^{p,q,k}, \delta_{t,1}^p, \delta_{t,2}^q, \delta_{t,3}^k\}$ represent the first-order linearization terms of $\Delta \Psi_{t,\text{ft}}$, and the higher-order terms with respect to learning rates are neglected. The training efficiency of LoTRA is evaluated based on the magnitude of the first-order improvement in the function value, as defined in Definition 1.

Definition 1. The training of LoTRA is considered as efficient if $\{\|\delta_{t,0}^{p,q,k}\|_2\}$, $\{\|\delta_{t,1}^p\|_2\}$, $\{\|\delta_{t,2}^q\|_2\}$, and $\{\|\delta_{t,3}^k\|_2\}$ are $\Theta(1)$ with respect to the model size (n, m, n_d) , for $p \in [r_1]$, $q \in [r_2]$, $k \in [r_3]$, and $t \geq 2$.

The definition of training efficiency assumes that the first-order improvement in function values captures the training progress, as learning rates are typically set to be small. Efficient training requires that the one-step improvement remains independent of the model size. To illustrate this, consider a scenario where the one-step improvement follows the order $\Theta(n^{\alpha_1} m^{\alpha_2} n_d^{\alpha_3})$. For stable training, it must hold that $\alpha_1, \alpha_2, \alpha_3 \leq 0$, otherwise, the model exhibits value explosion, leading to instability and training failure. However, if $\alpha_1, \alpha_2, \alpha_3 < 0$, the one-step improvement scales negatively with model size, resulting in excessively slow training as the model scales up. This is undesirable in practice. Therefore, to achieve both stability and efficiency, we prefer the condition $\alpha_1 = \alpha_2 = \alpha_3 = 0$, as stated in Definition 1.

Before delving into the details of the analysis, it is crucial to consider the initialization setup of parameters, because the initialization together with the learning rates determines the scale of change of function values during training. We initialize all trainable parameters as follows:

$$g_0^{p,q,k} = 0, \quad \mathbf{u}_0^{(1),q} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{n} \mathbf{I}\right), \quad \mathbf{u}_0^{(3),k} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{n_d} \mathbf{I}\right), \\ \|\mathbf{u}_0^{(2),q}\|_2 = \Theta(1), \text{ and } \mathbf{u}_0^{(2),q\top} \mathbf{v}_0 = \Theta(1), \quad (8)$$

for $p \in [r_1]$, $q \in [r_2]$ and $k \in [r_3]$, where $\mathbf{v}_0 := \frac{\partial \mathcal{L}_0}{\partial \Psi_{\text{ft}}} = \Psi_{0,\text{ft}}(\mathbf{x}) - \mathbf{y} = \Psi_{\text{pt}}(\mathbf{x}) - \mathbf{y}$ denotes the partial derivative of the loss with respect to the output of the fine-tuned model at initialization. The first three initialization schemes are standard, except for the initialization of $\mathbf{u}_0^{(2),q}$. A natural choice that satisfies the given condition is to explicitly compute the vector \mathbf{v}_0 based on the pre-trained model and initialize all $\mathbf{u}_0^{(2),q}$ as $\mathbf{v}_0 / \|\mathbf{v}_0\|_2$.

The following theorem presents the main result of the efficient training analysis for LoTRA. The proof extends the work of Hayou et al. [44], who analyzed the efficient training of LoRA involving two parameter components. In contrast, the proposed LoTRA framework is significantly more complex, consisting of four parameter components. Therefore, extending the proof and analysis is nontrivial and requires substantial modifications.

Theorem 2. *Efficient training of a one-layer KAN with LoTRA is not achievable, if all learning rates are set to the same order of magnitude with respect to the model size (n, m, n_d) , under the initialization scheme in Equation (8). However, efficient training can be achieved, if the learning rates are set as follows: $\eta_0 = \Theta(1)$, $\eta_1 = \Theta(n^{-1})$, $\eta_2 = \Theta(1)$, and $\eta_3 = \Theta(n_d^{-1})$.*

The proof can be found in Appendix C. Theorem 2 provides theoretical insights into learning rate selection for LoTRA. As the input dimension increases, the learning rate η_1 for the mode-1 transformation matrix should be decreased accordingly. Similarly, the learning rate η_3 for the mode-3 transformation matrix should be adjusted based on the number of basis functions n_d . In contrast, the learning rates η_0 for the core tensor and η_2 for the mode-2 transformation matrix remain independent of the model size. This theorem highlights an important principle for learning rate selection,

providing valuable practical guidance for learning rate tuning. In practice, the hidden dimension (or input dimension) n is usually larger than the number of basis functions n_d , i.e., $n > n_d$. Based on our theoretical results, an appropriate learning rate setup might follow $\eta_0 \approx \eta_2 \geq \eta_3 > \eta_1$. These theoretical results significantly streamline the learning rate selection process, reducing the learning rate search space and facilitating a more efficient tuning.

The extension of Theorem 2 to multi-layer KANs is feasible. The definition of efficient training in Definition 1 can be adapted for multi-layer KANs by requiring that all first-order linearization terms of each layer remain $\Theta(1)$ with respect to the model size. The proof follows a similar structure to that of Theorem 2, with minor modifications on $\mathbf{v}_t^\ell := \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{\ell,\text{ft}}}$, which represents the partial derivative of the loss function with respect to the output of ℓ -th layer (denoted as $\mathbf{z}_{\ell,\text{ft}}$) of the fine-tuned model $\Psi_{t,\text{ft}}$. The initialization of all trainable parameters is given by

$$g_0^{\ell,p,q,k} = 0, \quad \mathbf{u}_0^{(1),\ell,q} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{n_\ell} \mathbf{I}\right), \quad \mathbf{u}_0^{(3),\ell,k} \sim \mathcal{N}\left(\mathbf{0}, \frac{1}{n_d} \mathbf{I}\right), \\ \|\mathbf{u}_0^{(2),\ell,q}\|_2 = \Theta(1), \text{ and } \mathbf{v}_0^{\ell\top} \mathbf{u}_0^{(2),\ell,q} = \Theta(1),$$

for $p \in [r_1]$, $q \in [r_2]$, $k \in [r_3]$, and $\ell \in [L]$. Here, the initialization for $\mathbf{u}_0^{(2),\ell,q}$ can be achieved by explicitly calculating \mathbf{v}_0^ℓ for each layer based on the fine-tuned model, and setting all $\mathbf{u}_0^{(2),\ell,q}$ to be $\mathbf{v}_0^\ell / \|\mathbf{v}_0^\ell\|_2$. Corollary 1 establishes that the theoretically optimal learning rates for each layer depend on both the input dimension and the number of basis functions.

Corollary 1. *Efficient training of multi-layer KANs with LoTRA is not achievable, if all learning rates are set to the same order of magnitude as the model size $\{n_\ell\}_{\ell \in [L]}$ and n_d . However, for the ℓ -th layer of the fine-tuned model, efficient training holds if the learning rates are chosen as follows: $\eta_0 = \Theta(1)$, $\eta_1 = \Theta(n_\ell^{-1})$, $\eta_2 = \Theta(1)$, and $\eta_3 = \Theta(n_d^{-1})$.*

V. APPLICATIONS

In this section, we discuss the potential applications of LoTRA in fine-tuning and training KANs.

A. Physics-Informed KANs

KANs have demonstrated superior performance over MLPs in some science-related tasks, making them promising models for physics-informed machine learning. We investigate the potential of LoTRA in training KANs for solving a class of PDEs, aiming to reduce computational cost and storage requirements with enhanced efficiency. Consider the problem of solving a class of PDEs with different physical parameters $\lambda \in \mathcal{S}$:

$$\mathcal{D}[\mathbf{u}_\lambda; \lambda] = f(\mathbf{x}; \lambda), \quad \mathbf{x} \in \Omega, \\ \mathcal{B}[\mathbf{u}_\lambda; \lambda] = g(\mathbf{x}; \lambda), \quad \mathbf{x} \in \partial\Omega,$$

where \mathcal{D} and \mathcal{B} are differential operators defined in the interior domain Ω and on its boundary $\partial\Omega$, respectively. The solution corresponding to physical constants λ is denoted as \mathbf{u}_λ .

In physics-informed machine learning, neural networks act as surrogates for PDE solutions. Given a KAN model $\Psi(\mathbf{x}; \{\mathcal{A}_\ell\}_{\ell \in [L]})$, the training loss is formulated as

$$\mathcal{L}(\{\mathcal{A}_\ell\}_{\ell \in [L]}) = \frac{\mu}{N} \sum_{i=1}^N \left\| \mathcal{D} \left[\Psi(\mathbf{x}_i; \{\mathcal{A}_\ell\}_{\ell \in [L]}) \right] - \mathbf{y}_i \right\|_2^2 + \frac{\mu_b}{N_b} \sum_{j=1}^{N_b} \left\| \mathcal{B} \left[\Psi(\hat{\mathbf{x}}_j; \{\mathcal{A}_\ell\}_{\ell \in [L]}) \right] - \mathbf{b}_j \right\|_2^2, \quad (9)$$

given some observations $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i \in [N]}$ with $\mathbf{y}_i = f(\mathbf{x}_i; \boldsymbol{\lambda})$ in the interior and $\{(\hat{\mathbf{x}}_j, \mathbf{b}_j)\}_{j \in [N_b]}$ with $\mathbf{b}_j = g(\hat{\mathbf{x}}_j; \boldsymbol{\lambda})$ on the boundary. Here, $\mu > 0$ and $\mu_b > 0$ are hyperparameters that balance the PDE residual in the interior domain and the boundary condition residual, respectively.

Our objective is to train KANs to efficiently approximate solutions $\{\mathbf{u}_\lambda\}_{\lambda \in \mathcal{S}}$ for the entire class of PDEs. Due to structural similarities in the PDE operators, we assume that the solutions exhibit shared underlying patterns. Instead of repeatedly learning this shared information from each individual PDE, we adopt LoTRA to retain the shared information across all PDEs and adapt to task-specific variations using a low tensor-rank adaptation framework. This enables efficient fine-tuning of different PDEs, reducing redundancy in training and significantly improving storage efficiency.

Given a PDE with physical parameters $\boldsymbol{\lambda}_0$, we first pre-train the full KAN model on this PDE, where the obtained model Ψ_{pt} is an approximation to the corresponding solution $\mathbf{u}_{\boldsymbol{\lambda}_0}$. The shared information across the class of PDEs is inherently captured in Ψ_{pt} . For a new PDE with different physical parameters $\hat{\boldsymbol{\lambda}} \in \mathcal{S}$, we fine-tune the KANs model by LoTRA and obtain Ψ_{ft} , which approximates the solution $\mathbf{u}_{\hat{\boldsymbol{\lambda}}}$. The fine-tuning process is efficient because the shared information has already been included, eliminating the need for redundant learning, and thus resulting in faster convergence.

Moreover, when storing KAN models for an entire class of PDEs, the storage requirements are significantly reduced if the core tensor \mathcal{G} is much smaller than the original parameter tensor. In this case, rather than storing the full model for each PDE, it suffices to retain the parameter tensors of Ψ_{pt} and the low tensor-rank components $\{(\mathcal{G}_\ell, \mathbf{U}_\ell^{(1)}, \mathbf{U}_\ell^{(2)}, \mathbf{U}_\ell^{(3)})\}_{\ell \in [L]}$ for each PDE. For notational simplicity, we use “cr” to denote the compression ratio of LoTRA. A compression ratio of $\text{cr} = 1/4$ implies that each mode of the core tensor \mathcal{G} is compressed by $1/4$, reducing the dimensionality of each mode and resulting in an overall parameter ratio of $(1/4)^3$ compared to the original parameter tensor. Mathematically, this is expressed as $(r_{\ell,1}, r_{\ell,2}, r_{\ell,3}) = \lceil \text{cr} \cdot (n_\ell, n_{\ell+1}, n_d) \rceil$, where the ceiling operator $\lceil \cdot \rceil$ is applied elementwise to ensure that each component is rounded up to the nearest integer. This approach significantly reduces parameter size while maintaining the expressiveness and adaptability of the model.

B. Slim KANs

Another potential application of LoTRA is slimming the model, enabling a slimmer KAN architecture with low tensor-rank structures on parameter tensors. Unlike transfer learning,

this approach directly imposes a low tensor-rank structure on the parameter tensors of KANs from the outset. Suppose no pre-trained model is available, and we initialize all pre-trained parameter tensors $\mathcal{A}_{\ell, \text{pt}}$ as zero tensors. This setup is equivalent to assuming that the parameter tensors themselves exhibit low tensor-rank properties. Similar assumptions have been validated in other domains, such as large language models [45]. Additionally, in Section III-A, we empirically demonstrate that KAN parameter tensors naturally exhibit a low tensor-rank structure in the function representation task. In this case, the only trainable parameters in the slim KAN model are $\{(\mathcal{G}_\ell, \mathbf{U}_\ell^{(1)}, \mathbf{U}_\ell^{(2)}, \mathbf{U}_\ell^{(3)})\}_{\ell \in [L]}$. If the core tensor is significantly smaller than the full parameter tensor, it results in a much smaller model with certain expressiveness, compared to the full KAN with the same width and depth. Besides the benefits of reduced parameter size, the integrated low-rank structure of parameters acts as a form of regularization, preventing overfitting and improving generalization. In our experiments, we applied the slim KAN model to function representation and image classification tasks. The results show that it achieves comparable performance to vanilla KANs, demonstrating the potential of slim KANs constructed by LoTRA.

VI. EXPERIMENTS

In this section, we conduct comprehensive experiments on transfer learning of KANs using LoTRA in solving a class of PDEs. Additionally, we validate the learning rate selection strategy derived in Theorem 2. Slim KANs are also evaluated and compared with MLPs on function representation and image classification tasks. To explore the impact of different basis functions, we consider Chebyshev polynomials, Legendre polynomials, Taylor polynomials, and Fourier series as basis functions, and denote the corresponding KAN models as ChebyKAN, LegendreKAN, TaylorKAN, and FourierKAN, respectively. For all pre-training and fine-tuning models, we adopt the Adam optimizer with its default hyperparameters.

A. Transfer Learning of KANs

In this experiment, we apply KANs with LoTRA to solve a class of PDEs, as detailed in Section V-A. We consider three types of second-order PDEs: elliptic, parabolic, and hyperbolic equations. For a given PDE in the class, we first pre-train a KAN model to obtain Ψ_{pt} . For new PDE tasks within the same class, we fine-tune the pre-trained model using LoTRA, resulting in the fine-tuned model Ψ_{ft} . To evaluate the quality of the obtained models, we compute the relative error (rel) between the predicted solution $\Psi(\mathbf{x})$ and the exact solution $\mathbf{u}(\mathbf{x})$. Given a test dataset $\{(\mathbf{x}_i, \mathbf{u}(\mathbf{x}_i))\}_{i \in N_t}$, the relative error is defined as $\text{rel} = \frac{\sum_{i=1}^{N_t} \|\Psi(\mathbf{x}_i) - \mathbf{u}(\mathbf{x}_i)\|_2^2}{\sum_{i=1}^{N_t} \|\mathbf{u}(\mathbf{x}_i)\|_2^2}$. In the figures, the method labeled “full” refers to the vanilla transfer learning approach, where all parameters are fully updated. The method labeled “zero” represents training from scratch without leveraging any information from a pre-trained model.

As discussed in Section V-A, we denote the compression ratio of LoTRA as “cr.” A compression ratio of $\text{cr} = 1/4$

implies that each mode of the core tensor \mathcal{G} is reduced to $1/4$ of its original size, leading to an overall parameter reduction of $(1/4)^3$ compared to the original parameter tensor. The mathematical formulation follows $(r_{\ell,1}, r_{\ell,2}, r_{\ell,3}) = \lceil \text{cr} \cdot (n_\ell, n_{\ell+1}, n_d) \rceil$, where the ceiling operator $\lceil \cdot \rceil$ is applied elementwise to ensure that each component is rounded up to the nearest integer. In our experiments, we utilize LoTRA with three compression ratios, namely $\text{cr} = 1/2, 1/4$, and $1/8$. We adopt three-layer KANs with a fixed hidden dimension $n_\ell = 64$ and the number of basis functions set to $n_d = 8$. We consider PDEs with two-dimensional spatial domains. Since the input dimension and the output dimension studied here are significantly smaller than the hidden dimension, we apply LoTRA only to the hidden layer, while fine-tuning the input and the output layers normally without compression.

1) *Elliptic Equations*: We consider a class of elliptic equations with varying parameters $\lambda \in \mathbb{R}$:

$$-\nabla \cdot (a(\mathbf{x}) \cdot \nabla u(\mathbf{x}; \lambda)) + \|\nabla u(\mathbf{x}; \lambda)\|_2^2 = f(\mathbf{x}; \lambda), \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}; \lambda) = g(\mathbf{x}; \lambda), \quad \mathbf{x} \in \partial\Omega, \quad (10)$$

where the domain is defined as $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_2 \leq 1\}$, and the coefficient function is given by $a(\mathbf{x}) = 1 + \frac{1}{2} \|\mathbf{x}\|_2^2$. The exact solution $u(\mathbf{x}; \lambda)$ with the parameter λ is defined as $u(\mathbf{x}; \lambda) = \sin(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)^{2.5}) + \lambda \cdot \sin(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2))$. We first pre-train a KAN model on the PDE with $\lambda = 0$, and then fine-tune it on PDEs with $\lambda = 0.1$ and $\lambda = 1$.

We first validate the developed theorem on the learning rate selection by designing four different learning rate strategies. In the first strategy, we set the learning rates based on Theorem 2 considering the hidden dimension and the number of basis functions. Specifically, we choose $(\eta_0, \eta_1, \eta_2, \eta_3) = (1\text{e-}2, 2\text{e-}4, 1\text{e-}2, 1\text{e-}3)$, which is consistent with the theorem if the constant in $\Theta(\cdot)$ is assumed to be $1\text{e-}2$. For the second and third strategies, we set learning rates for all components of LoTRA to be $3\text{e-}3$ and $1\text{e-}3$, respectively. In the fourth strategy, we deliberately violate the theoretical conditions by making η_1 the largest learning rate. Specifically, we set $(\eta_0, \eta_1, \eta_2, \eta_3) = (2\text{e-}4, 1\text{e-}2, 1\text{e-}3, 1\text{e-}2)$. The fine-tuning trajectories of KANs with LoTRA under these learning rates strategies are illustrated in Figure 2 (as well as in Figures S1 and S3 in the supplementary material). In these figures, “LR-1” to “LR-4” correspond to the four respective learning rate strategies. Across different transfer learning tasks, including varying parameter values λ , different compression ratios, and various KAN variants, we observe that the first strategy, designed from our theoretical results, achieves the lowest training loss in most cases. This empirical observation validates Theorem 2. In some cases, the first strategy exhibits slightly lower performance. This is primarily due to the elliptic PDE examples being relatively easy for KANs to learn, leading to an overly simple fine-tuning process where differences between strategies are not as pronounced.

We compare the fine-tuned KANs using the introduced LoTRA with several baseline methods: the vanilla transfer learning approach, which updates all parameters based on the pre-trained model; KANs trained from scratch without

transfer learning; and MLPs using the vanilla transfer learning method. The visualized results are shown in Figure 3 (as well as in Figures S2 and S4 in the supplementary material). The results demonstrate that KAN models with LoTRA are highly competitive and perform comparably with the vanilla transfer learning method updating all parameters, with significantly reduced parameter size and maintained performance. We further observe that compression ratios of $\text{cr} = 1/2$ and $\text{cr} = 1/4$ exhibit comparable performances, with the latter occasionally outperforming the former. This is likely due to the simplicity of the example, where the compression ratio remains too large for $\text{cr} = 1/2$, making $\text{cr} = 1/4$ a suitable compression ratio for this case. To further evaluate performance, we compute the relative error of the fine-tuned models, as shown in Tables I and II. The results indicate that fine-tuned models with LoTRA perform comparably to, and in some cases outperform, full fine-tuning models. Additionally, KANs consistently outperform MLPs, highlighting their potential in physics-informed machine learning tasks.

Method		rel (%)	Method		rel (%)
Type	cr		Type	cr	
ChebyKAN -LoTRA	1/2	0.05	TaylorKAN -LoTRA	1/2	0.20
	1/4	0.06		1/4	0.25
	1/8	0.08		1/8	0.23
ChebyKAN-full		0.05	TaylorKAN-full		0.26
ChebyKAN-zero		0.08	TaylorKAN-zero		1.34
LegendreKAN -LoTRA	1/2	0.08	FourierKAN -LoTRA	1/2	0.14
	1/4	0.08		1/4	0.11
	1/8	0.09		1/8	0.13
LegendreKAN-full		0.08	FourierKAN-full		0.14
LegendreKAN-zero		0.14	FourierKAN-zero		0.22
MLP-full		0.30	MLP-zero		0.80

TABLE I
RELATIVE ERROR (REL) OF KANS WITH LoTRA, FULLY UPDATED KANS AND MLPs, AND KANS AND MLPs TRAINED FROM SCRATCH WITHOUT TRANSFER LEARNING, FOR SOLVING ELLIPTIC EQUATIONS WITH PARAMETER $\epsilon = 0.1$.

Method		rel (%)	Method		rel (%)
Type	cr		Type	cr	
ChebyKAN -LoTRA	1/2	0.08	TaylorKAN -LoTRA	1/2	0.19
	1/4	0.09		1/4	0.22
	1/8	0.10		1/8	0.20
ChebyKAN-full		0.04	TaylorKAN-full		0.19
ChebyKAN-zero		0.04	TaylorKAN-zero		0.55
LegendreKAN -LoTRA	1/2	0.14	FourierKAN -LoTRA	1/2	0.16
	1/4	0.20		1/4	0.17
	1/8	0.33		1/8	0.39
LegendreKAN-full		0.10	FourierKAN-full		0.10
LegendreKAN-zero		0.15	FourierKAN-zero		0.18
MLP-full		0.25	MLP-zero		0.28

TABLE II
RELATIVE ERROR (REL) OF KANS WITH LoTRA, FULLY UPDATED KANS AND MLPs, AND KANS AND MLPs TRAINED FROM SCRATCH WITHOUT TRANSFER LEARNING, FOR SOLVING ELLIPTIC EQUATIONS WITH PARAMETER $\epsilon = 1.0$.

2) *Allen-Cahn Equations*: We consider a class of Allen-Cahn equations, which are nonlinear parabolic PDEs, with

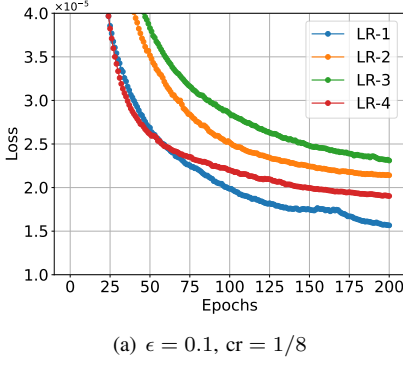
(a) $\epsilon = 0.1$, $cr = 1/8$

Fig. 2. Fine-tuning trajectories of Chebyshev KANs using LoTRA under four strategies of learning rate selection (denoted as “LR-1” to “LR-4”) for solving elliptic equations.

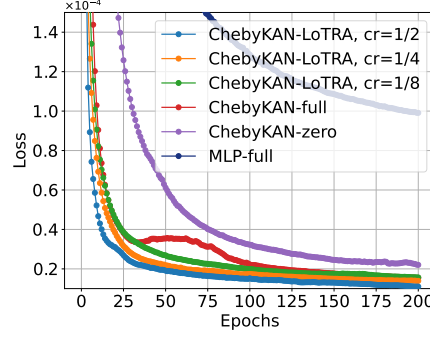
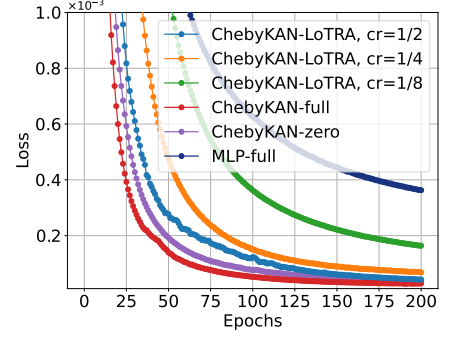
(a) $\epsilon = 0.1$ (b) $\epsilon = 1.0$

Fig. 3. Fine-tuning trajectories of Chebyshev KANs using LoTRA, compared to fully updated KANs and MLPs, for solving elliptic equations.

varying parameters $\lambda \in \mathbb{R}$:

$$\begin{aligned} \frac{\partial u(t, \mathbf{x}; \lambda)}{\partial t} - \Delta u(t, \mathbf{x}; \lambda) - u(t, \mathbf{x}; \lambda)^3 + u(t, \mathbf{x}; \lambda) \\ = f(t, \mathbf{x}; \lambda), \quad (t, \mathbf{x}) \in [0, 1] \times \Omega, \\ u(t, \mathbf{x}; \lambda) = g(t, \mathbf{x}; \lambda), \quad (t, \mathbf{x}) \in [0, 1] \times \partial\Omega, \\ u(0, \mathbf{x}; \lambda) = h(\mathbf{x}; \lambda), \quad \mathbf{x} \in \Omega, \end{aligned} \quad (11)$$

where the temporal and the spatial domains are defined as $[0, 1]$ and $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_2 \leq 1\}$, respectively. The exact solution $\mathbf{u}(\mathbf{x}; \lambda)$ with the parameter λ is defined as $u(\mathbf{x}; \lambda) = e^{-t} \sin\left(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)^{2.5}\right) + \lambda \cdot e^{-t} \sin\left(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)\right)$. We first pre-train a KAN model on the PDE with $\lambda = 0$, and then fine-tune it on PDEs with $\lambda = 0.1$ and $\lambda = 1$.

We conduct similar experiments as in Section VI-A1 for solving Allen-Cahn equations using KANs with LoTRA, adopting four different learning rate strategies to verify the derived Theorem 2. The four strategies follow those in Section VI-A1, where the first strategy is consistent with the theorem, while the other three deviate from the theoretically optimal learning rate selection. The fine-tuning trajectories of KANs with LoTRA under these learning rates strategies are illustrated in Figures 4 (as well as in Figures S5 and S7 in the supplementary material). Across different transfer learning tasks, including varying parameter values λ , different compression ratios, and various KAN variants, we observe that the first strategy, derived from our theoretical results, consistently achieves the lowest training loss in all cases. This empirical observation validates Theorem 2, demonstrating the effectiveness of the proposed learning rate selection strategy. The advantages of using the first strategy are particularly obvious in tasks with $\epsilon = 1.0$, as these cases are relatively more challenging. Therefore, the benefits of the theoretically guided learning rate selection become more noticeable compared to the results obtained for elliptic equations in Figure 2.

We compare the fine-tuned KANs using the introduced LoTRA with several baseline methods for solving Allen-Cahn equations. The baseline models are the same as those examined in Section VI-A1. The visualized results are presented in Figures 5 (as well as in Figures S6 and S8 in the supplementary material). The results demonstrate that KAN models with Lo-

TRA outperform and sometimes comparably with the vanilla fine-tuning method with full parameter updates. Additionally, they significantly outperform models trained from scratch without transfer learning. We further observe that LoTRA with larger compression ratios (i.e., larger core tensors) achieve lower training loss, benefiting from greater expressiveness. This observation differs slightly from the results for elliptic equations, mainly because Allen-Cahn equations are more challenging to solve. Therefore, when applying LoTRA, it is crucial to balance the trade-off between compression ratio and model expressiveness, depending on the specific application requirements. To further evaluate performance, we compute the relative error of the fine-tuned models, as shown in Tables SI and SII in the supplementary material. The results indicate that fine-tuned models with LoTRA outperform or perform comparably with full fine-tuning models and models trained from scratch. Moreover, KANs consistently outperform MLPs in solving Allen-Cahn equations, showing their potential in science-related tasks.

3) *Hyperbolic Equations*: We consider a class of hyperbolic equations with varying parameters $\lambda \in \mathbb{R}$:

$$\begin{aligned} \frac{\partial^2 u(t, \mathbf{x}; \lambda)}{\partial t^2} - \Delta u(t, \mathbf{x}; \lambda) = f(t, \mathbf{x}; \lambda), \quad (t, \mathbf{x}) \in [0, 1] \times \Omega, \\ u(t, \mathbf{x}; \lambda) = g(t, \mathbf{x}; \lambda), \quad (t, \mathbf{x}) \in [0, 1] \times \partial\Omega, \\ u(0, \mathbf{x}; \lambda) = h(\mathbf{x}; \lambda), \quad \mathbf{x} \in \Omega, \\ \frac{u(0, \mathbf{x}; \lambda)}{\partial t} = \bar{h}(\mathbf{x}; \lambda), \quad \mathbf{x} \in \Omega, \end{aligned} \quad (12)$$

where the temporal and the spatial domains are defined as $[0, 1]$ and $\Omega = \{\mathbf{x} \in \mathbb{R}^2 : \|\mathbf{x}\|_2 \leq 1\}$, respectively. The exact solution $\mathbf{u}(\mathbf{x}; \lambda)$ with the parameter λ is defined as $u(\mathbf{x}; \lambda) = (e^{t^2} - 1) \sin\left(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)^{2.5}\right) + \lambda \cdot (e^{t^2} - 1) \sin\left(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)\right)$. We first pre-train a KAN model on the PDE with $\lambda = 0$, and then fine-tune it on PDEs with $\lambda = 0.1$ and $\lambda = 1$.

We conduct similar experiments as in Sections VI-A1 and VI-A2 for solving hyperbolic equations using KANs with LoTRA, adopting four different learning rate strategies to verify the theoretical results in Theorem 2. The four strategies follow those in Sections VI-A1 and VI-A2, where the first strategy is consistent with the theorem, while the other three deviate from the theoretically optimal learning rate selection.

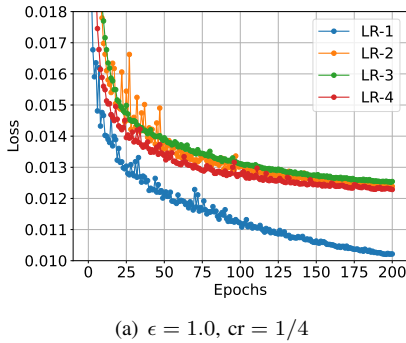


Fig. 4. Fine-tuning trajectories of Chebyshev KANs using LoTRA under four strategies of learning rate selection (denoted as “LR-1” to “LR-4”) for solving Allen-Cahn equations.

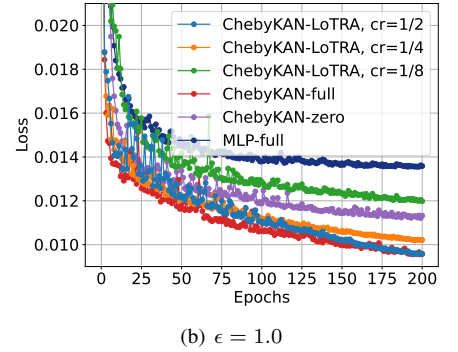
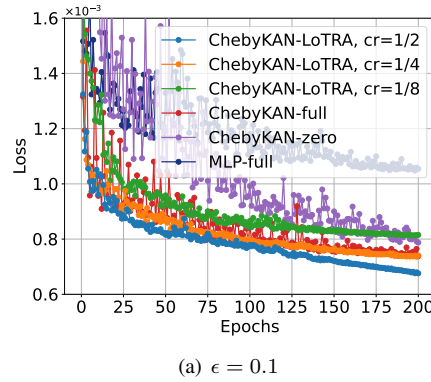


Fig. 5. Fine-tuning trajectories of Chebyshev KANs using LoTRA, compared to fully updated KANs and MLPs, for solving Allen-Cahn equations.

The fine-tuning trajectories of KANs with LoTRA under these learning rates strategies are illustrated in Figures S9 and S11 in the supplementary material. Across different transfer learning tasks, including varying parameter values λ , different compression ratios, and various KAN variants, we observe that the first strategy, derived from our theoretical results, consistently achieves the lowest training loss in most cases. In some instances, the first strategy may exhibit slightly lower performance. This is primarily due to our naive assumption of setting the constant in $\Theta(\cdot)$ to be $1e-2$ for all learning rates, which is not the best value. However, given the large number of variables involved, we must fix the constant term for practical feasibility and comparison. This empirical observation further validates Theorem 2, demonstrating the effectiveness of the proposed learning rate selection strategy. The superiority of the first learning rate strategy are even more evident in hyperbolic equations compared to those observed in elliptic and Allen-Cahn equations. This is mainly due to the higher complexity of hyperbolic equations, which involve second-order derivatives on the temporal variable, making them more challenging to solve. Therefore, the benefits of the theoretically guided learning rate selection are more apparent when applied to hyperbolic equations than to elliptic and Allen-Cahn equations.

We compare the fine-tuned KANs using the proposed LoTRA with several baseline methods for solving hyperbolic equations. The baseline models are the same as those examined in Sections VI-A1 and VI-A2. The visualized results are presented in Figures S10 and S12 in the supplementary material. The results demonstrate that KAN models with LoTRA and $cr = 1/2$ outperform the vanilla fine-tuning method with full parameter updates. Additionally, they significantly outperform models trained from scratch without transfer learning. We also observe that LoTRA with larger cr (i.e., larger core tensors) achieve lower training loss due to increased expressiveness. This observation differs slightly from the results for elliptic equations but is consistent with those for Allen-Cahn equations. Notably, the performance gap in hyperbolic equations becomes even more obvious and amplified across different core tensor sizes. This is primarily because hyperbolic equations, which involve higher-

order derivatives in the temporal domain, are inherently more challenging and demand higher model expressiveness. To further evaluate performance, we compute the relative error of the fine-tuned models, as shown in Tables SIII and SIV in the supplementary material. The results indicate that fine-tuned models with LoTRA outperform or perform comparably with full fine-tuning models and models trained from scratch. Moreover, KANs consistently and significantly outperform MLPs in solving hyperbolic equations, further showing their potential in science-related tasks. Both LoTRA-based fine-tuning and full parameter updates outperform models trained from scratch in most cases, suggesting that transfer learning effectively captures shared information from the pre-training task, leading to faster convergence.

B. Slim KANs

In the previous section, we demonstrated the effectiveness of LoTRA in transfer learning for KANs, successfully solving three classes of PDEs with faster convergence and significantly less parameter size. In this subsection, we further evaluate slim KANs, developed based on the LoTRA framework without pre-trained parameters, on function representation and image classification tasks.

1) Trigonometric Function: We evaluate the expressiveness of slim KANs on a simple function representation task. Specifically, we aim to represent a two-dimensional trigonometric function given by $u(x, y) = \sin(\pi x) \sin(\pi y)$. To introduce complexity, we add Gaussian noise with zero mean and a standard deviation of 0.05. We experiment with various compression ratios (core tensor size) and basis functions. For this task, we use three-layer KANs with a hidden dimension of $n_\ell = 32$ and $n_d = 8$. Given the small input and output dimensions, we only apply compression to the hidden layers. We use the MLP with a similar parameter size (hidden dimension to be 128) as the baseline model to compare its performance with KANs on the trigonometric function representation task.

The training loss with respect to the compression ratio (cr) is visualized in Figure 6. From the results, we observe that the training loss decreases monotonically as the compression ratio increases, highlighting the improved expressiveness of slim KANs with larger core tensors. This observation is consistent

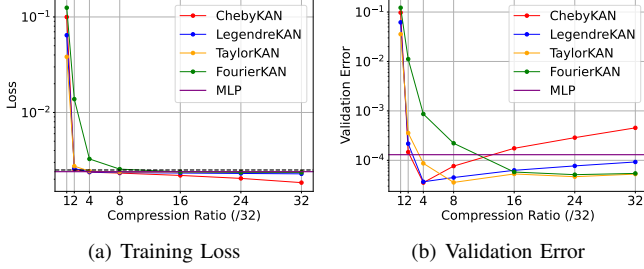


Fig. 6. Relationship between the compression ratio and the performances on representing a trigonometric function. (a) Training Loss: The training loss decreases with an increasing compression ratio. (b) Validation Error: The validation error shows a U-shape, indicating a balance between model complexity and generalization.

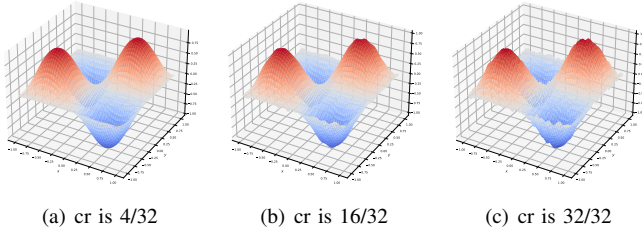


Fig. 7. Visualization of the generated function by slim KAN models with different compression ratios for representing a trigonometric function. The figures illustrate that models with a larger compression ratio tend to overfit the noise, producing less smooth results. In contrast, models with smaller compression ratios generate smoother functions, reflecting better generalization.

with our intuition. However, when considering the validation error, the curve exhibits a U-shape, indicating overfitting in slim KANs with larger core tensor sizes, which is also within our expectations. Here, in the simple example, we observe that the MLP demonstrates comparable performance to KANs. Furthermore, the dashed line in the figure represents the expected training loss due to the introduced noise. Interestingly, slim KANs with larger core tensors achieve training losses smaller than this threshold, implying that they are fitting the noise. This phenomenon is further corroborated by the observed increase in validation error when the model achieves lower training losses than the noise level. The visualization of generated functions is shown in Figure 7. Models with larger compression ratios tend to overfit the noise, resulting in a nonsmooth surface for the generated shape. In summary, our observations confirm that Slim KANs are highly expressive for function representation tasks. Additionally, the low-rank structure introduced by LoTRA acts as an effective regularization mechanism to mitigate overfitting.

2) *Nonsmooth and Sharp Function*: We evaluate the performance of slim KANs on a more challenging task of representing a two-dimensional nonsmooth and sharp function. Due to the growing complexity of the function, we use a larger KAN model with $n_\ell = 128$ hidden dimensions and $n_d = 8$ basis functions. As a baseline for comparison, we use an MLP with a hidden dimension of 1024.

The relationship between the compression ratio (core tensor size) and model performance is visualized in Figure 8. The

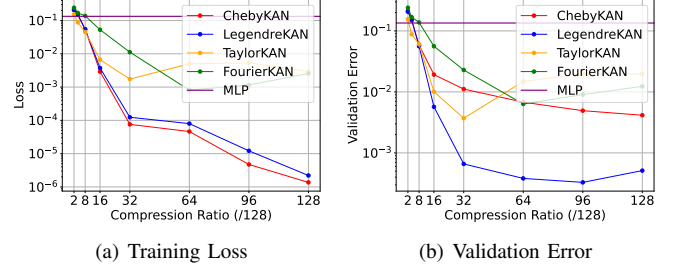


Fig. 8. Relationship between the compression ratio and the performances on representing a nonsmooth function. (a) Training Loss: The training loss decreases with an increasing compression ratio. (b) Validation Error: The validation error shows a similar trend, indicating higher expressiveness with a larger compression ratio.

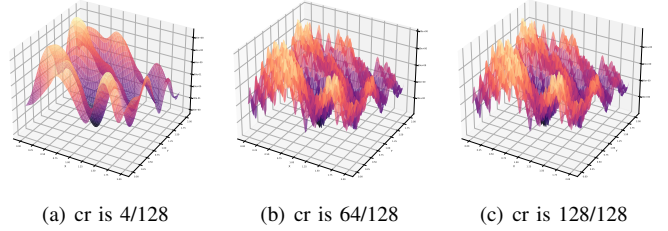


Fig. 9. Visualization of the generated function by KAN models with different compression ratios for representing a nonsmooth function. The figures highlight that models with larger compression ratios exhibit higher expressiveness.

figure demonstrates that larger compression ratios correspond to higher model complexity and expressiveness, leading to lower training loss and validation error. This behavior differs slightly from the results for the trigonometric function, as the trigonometric function is relatively simple, whereas the nonsmooth function is significantly more challenging. Even with an enlarged model size, the KAN model does not overfit the nonsmooth function, and the characteristic U-shape observed in the trigonometric function task does not appear. Additionally, the MLP fails to accurately represent the nonsmooth function, on the contrary, KANs achieve substantially lower training loss and validation error, showing the superior ability of KANs in function representation compared to MLP. The visualization of the generated functions is shown in Figure 9. The model with $cr = 4/64$ fails to capture the oscillatory trajectories of the function due to its limited expressiveness. In contrast, the slim KAN model with a larger core tensor size effectively captures more intricate details of this complicated nonsmooth function.

3) Image Classification: MNIST and CIFAR-10 Datasets:

We further apply Slim KANs to image classification tasks on the MNIST and CIFAR-10 datasets. For the MNIST dataset, we adopt the three-layer feed-forward Slim KANs model, where the input and hidden layers are compressed, while the output layer remains uncompressed. The input dimension, hidden dimension, and output dimension are in the size of 28×28 , 128, and 10, respectively. Following the experimental settings in [18], the number of basis functions is set to $n_d = 4$. For the CIFAR-10 dataset, which is more complex, we use a one-layer ResNet [46] consisting of three convolutional layers, with the

hidden feed-forward and the output layer replaced by slim KAN layers. The hidden dimension of the slim KAN layer is set to $n_\ell = 128$ and the number of basis functions is $n_d = 4$. After the convolutional layers, the input to the slim KAN layer has a dimension of 256. Compression is applied only to this layer, while the output layer remains as a standard KAN layer. Since n_d is much smaller than the hidden dimension, we do not compress the third mode of the tensor, meaning the third mode of the core tensor matches the original tensor. We also compare slim KANs against MLPs as baseline models with a comparable parameter size (denoted as “MLP-small”) and significantly larger parameter size (denoted as “MLP-large”, with approximately four times more parameters). The performance of Slim KANs with varying compression ratios and different basis functions is summarized in Tables III and IV.

We observe from Table III that models with larger compression ratios (i.e., larger core tensors) demonstrate greater expressiveness and achieve higher accuracy. However, in this example, slim KANs and original KANs perform slightly worse than MLPs of similar parameter sizes. A similar observation was reported in [47], which conducted a comprehensive comparison between KANs and MLPs across various tasks. The study found that while KANs often outperform MLPs in function representation tasks, they tend to slightly underperform in computer vision and machine learning tasks. Our results are consistent with these observations. Despite this, we emphasize that slim KANs, with significantly fewer parameters, maintain competitive and satisfying performance, demonstrating their potential for parameter-efficient tasks. This emphasizes the potential and promise of slim KANs in applications where reducing parameter sizes is crucial without significantly compromising accuracy.

Table IV further highlights the expressiveness and potential of slim KANs models when integrated with ResNet. Similarly, we observe that a larger compression ratio enhances expressiveness and leads to higher accuracy. Notably, when combined with a one-layer ResNet, the model outperforms the baseline MLP models with comparable parameter sizes. These results demonstrate the potential of slim KANs when integrated with advanced architectures in specific applications.

Method		Accuracy (%)	Method		Accuracy (%)
Type	cr (/128)		Type	cr (/128)	
Chebyshev	8	94.48	Taylor	8	95.15
	16	96.21		16	96.77
	32	96.59		32	96.86
	64	96.72		64	96.64
	96	96.57		96	96.36
	128	97.18		128	96.68
Legendre	8	95.05	Fourier	8	95.01
	16	95.79		16	96.20
	32	96.45		32	96.48
	64	96.76		64	96.81
	96	97.42		96	97.54
	128	97.73		128	97.57
MLP-Small		98.08	MLP-Large		98.47

TABLE III

ACCURACY OF SLIM KANs, MLP WITH COMPARABLE PARAMETERS (MLP-SMALL), AND MLP WITH AN ENLARGED MODEL SIZE (MLP-LARGE) FOR THE IMAGE CLASSIFICATION TASK ON THE MNIST DATASET.

Method		Accuracy (%)	Method		Accuracy (%)
Type	cr (/128)		Type	cr (/128)	
Chebyshev	8	69.67	Taylor	8	69.83
	16	73.43		16	73.33
	32	74.92		32	75.33
	48	75.92		48	75.82
	64	76.24		64	76.10
	96	76.50		96	76.24
	128	76.63		128	76.27
Legendre	8	69.88	Fourier	8	68.95
	16	73.48		16	72.96
	32	75.44		32	74.41
	48	75.55		48	75.49
	64	75.39		64	75.63
	96	75.62		96	76.50
	128	76.13		128	76.27
MLP-Small		73.29	MLP-Large		78.33

TABLE IV

ACCURACY OF SLIM KANs, MLP WITH COMPARABLE PARAMETERS (MLP-SMALL), AND MLP WITH AN ENLARGED MODEL SIZE (MLP-LARGE), WHEN COMBINED WITH A ONE-LAYER RESNET, FOR THE IMAGE CLASSIFICATION TASK ON THE CIFAR-10 DATASET.

VII. CONCLUSION

In this paper, we introduce the concept of low tensor-rank adaptation (LoTRA) for the transfer learning of KANs, inspired by Tucker decomposition in tensors and the success of LoRA for matrix parameter updates. We begin by empirically observing that both KAN parameters and fine-tuning updates exhibit a low tensor-rank structure, which motivates us to develop LoTRA as an efficient parameter update method. We then theoretically establish the expressiveness of LoTRA based on Tucker decomposition approximations. Additionally, we propose a theoretically grounded learning rate selection strategy for efficient training of LoTRA, providing theoretical insights for practical implementation. Our analysis further reveals that applying identical learning rates to all LoTRA components is inefficient. Beyond theoretical insights, we explore the practical applications of LoTRA, particularly in fine-tuning KANs for solving PDEs and slimming KANs models. Experimental results validate our proposed learning rate selection strategy and demonstrate the effectiveness of LoTRA in fine-tuning KANs for solving PDEs. Furthermore, we evaluate Slim KANs in function representation and image classification tasks, showing that slim KANs maintain satisfying performance and significantly reduce the number of parameters. This is the first paper studying the transfer learning and fine-tuning of KANs with LoTRA.

Although our study mainly focuses on fine-tuning KANs using LoTRA for solving various PDEs, further exploration of LoTRA for broader transfer learning tasks remains an important direction. A deeper theoretical analysis of LoTRA is needed to enhance our understanding of its underlying properties, leading to better model interpretability and practical implementation. Future research could further refine its theoretical foundations and explore its practical integration into more complex deep learning architectures.

APPENDIX

B. Derivation for $\Delta\Psi_{t,ft}$

A. Proof for Theorem 1

For notational simplicity, we denote $\mathbf{z}_{\ell,ft}$ and $\mathbf{z}_{\ell,tg}$ as the ℓ -th layer outputs of the fine-tuned model Ψ_{ft} and the target model Ψ_{tg} , respectively. Then, we have

$$\begin{aligned} & \|\Psi_{ft}(\mathbf{x}) - \Psi_{tg}(\mathbf{x})\|_2 \\ &= \|\Phi_L(\mathbf{z}_{\ell,ft}; \mathcal{A}_{L,ft}) - \Phi_L(\mathbf{z}_{\ell,tg}; \mathcal{A}_{L,tg})\|_2 \\ &\leq \|\Phi_L(\mathbf{z}_{\ell,ft}; \mathcal{A}_{L,ft}) - \Phi_L(\mathbf{z}_{\ell,ft}; \mathcal{A}_{L,tg})\|_2 \\ &\quad + \|\Phi_L(\mathbf{z}_{\ell,ft}; \mathcal{A}_{L,tg}) - \Phi_L(\mathbf{z}_{\ell,tg}; \mathcal{A}_{L,tg})\|_2. \end{aligned}$$

We define $\hat{\mathbf{z}}_\ell \in \mathbb{R}^{n_{\ell} n_d}$ as $\hat{\mathbf{z}}_{\ell,k+(p-1)n_d} = b_k(\phi(\mathbf{z}_{\ell,p}))$, for $k \in [n_d]$ and $p \in [n_\ell]$. Let $\mathbf{A}_\ell^{(2)}$ denote the mode-2 unfolding of the tensor \mathcal{A}_ℓ , then we have

$$\begin{aligned} & \|\Phi_L(\mathbf{z}_{\ell,ft}; \mathcal{A}_{L,ft}) - \Phi_L(\mathbf{z}_{\ell,tg}; \mathcal{A}_{L,tg})\|_2 \\ &= \left\| \left(\mathbf{A}_{L,ft}^{(2)} - \mathbf{A}_{L,tg}^{(2)} \right) \hat{\mathbf{z}}_{L,ft} \right\|_2 \\ &\leq \left\| \mathbf{A}_{L,ft}^{(2)} - \mathbf{A}_{L,tg}^{(2)} \right\|_F \cdot \|\hat{\mathbf{z}}_{L,ft}\|_2 \\ &= \left\| \mathcal{A}_{L,ft}^{(2)} - \mathcal{A}_{L,tg}^{(2)} \right\|_F \cdot \|\hat{\mathbf{z}}_{L,ft}\|_2 \\ &\leq B\sqrt{n_L n_d} \cdot \left(\sum_{r=r_{L,1}+1}^{n_L} \sigma_r \left(\mathbf{E}_L^{(1)} \right)^2 + \sum_{r=r_{L,2}+1}^{n_{L+1}} \sigma_r \left(\mathbf{E}_L^{(2)} \right)^2 \right. \\ &\quad \left. + \sum_{r=r_{L,3}+1}^{n_d} \sigma_r \left(\mathbf{E}_L^{(3)} \right)^2 \right)^{1/2}, \end{aligned}$$

where $\|\hat{\mathbf{z}}_{L,ft}\|_2$ is uniformly bounded by $B\sqrt{n_L n_d}$ under Assumption 1.

Furthermore,

$$\begin{aligned} & \|\Phi_L(\mathbf{z}_{\ell,ft}; \mathcal{A}_{L,tg}) - \Phi_L(\mathbf{z}_{\ell,tg}; \mathcal{A}_{L,tg})\|_2 \\ &= \left\| \mathbf{A}_{L,tg}^{(2)} (\hat{\mathbf{z}}_{\ell,ft} - \hat{\mathbf{z}}_{\ell,tg}) \right\|_2 \\ &\leq \left\| \mathbf{A}_{L,tg}^{(2)} \right\|_F \cdot \|\hat{\mathbf{z}}_{\ell,ft} - \hat{\mathbf{z}}_{\ell,tg}\|_2 \\ &= \|\mathcal{A}_{L,tg}\|_F \cdot \|\hat{\mathbf{z}}_{\ell,ft} - \hat{\mathbf{z}}_{\ell,tg}\|_2 \\ &\leq M \cdot L\sqrt{n_d} \|\mathbf{z}_{\ell,ft} - \mathbf{z}_{\ell,tg}\|_2. \end{aligned}$$

Applying induction from $\ell = 1$ to L , we obtain the desired result.

$$\begin{aligned} \Delta\Psi_{t,ft} &:= \Psi_{t,ft} - \Psi_{t-1,ft} \\ &= \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} g_t^{p,q,k} \cdot \left(\mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{u}_t^{(3),k} \right) \cdot \mathbf{u}_t^{(2),q} \\ &\quad - \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} g_{t-1}^{p,q,k} \cdot \left(\mathbf{u}_{t-1}^{(1),p\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k} \right) \cdot \mathbf{u}_{t-1}^{(2),q} \\ &= \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} \left[g_{t-1}^{p,q,k} - \eta_0 \cdot \left(\mathbf{v}_{t-1}^\top \mathbf{u}_{t-1}^{(2),q} \right) \cdot \left(\mathbf{u}_{t-1}^{(1),p\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k} \right) \right] \cdot \\ &\quad \cdot \left[\mathbf{u}_{t-1}^{(1),p} - \eta_1 \cdot \left(\sum_{q'=1}^{r_2} \sum_{k'=1}^{r_3} g_{t-1}^{p,q',k'} \cdot \left(\mathbf{v}_{t-1}^\top \mathbf{u}_{t-1}^{(2),q'} \right) \cdot \mathbf{X} \mathbf{u}_{t-1}^{(3),k'} \right) \right]^\top \\ &\quad \cdot \mathbf{X} \left[\mathbf{u}_{t-1}^{(3),k} - \eta_3 \cdot \left(\sum_{p'=1}^{r_1} \sum_{q'=1}^{r_2} g_{t-1}^{p',q',k} \cdot \left(\mathbf{v}_{t-1}^\top \mathbf{u}_{t-1}^{(2),q'} \right) \mathbf{X}^\top \mathbf{u}_{t-1}^{(1),p'} \right) \right] \\ &\quad \cdot \left[\mathbf{u}_{t-1}^{(2),k} - \eta_2 \cdot \left(\sum_{p'=1}^{r_1} \sum_{q'=1}^{r_3} g_{t-1}^{p',q,k'} \cdot \left(\mathbf{u}_{t-1}^{(1),p'\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k'} \right) \cdot \mathbf{v}_{t-1} \right) \right] \\ &\quad - \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} g_{t-1}^{p,q,k} \cdot \left(\mathbf{u}_{t-1}^{(1),p\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k} \right) \cdot \mathbf{u}_{t-1}^{(2),q} \\ &\approx \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{k=1}^{r_3} \delta_{t,0}^{p,q,k} + \sum_{p=1}^{r_1} \delta_{t,1}^p + \sum_{q=1}^{r_2} \delta_{t,2}^q + \sum_{k=1}^{r_3} \delta_{t,3}^k, \end{aligned}$$

C. Proof for Theorem 2

If r_1 , r_2 , and r_3 are fixed constants and n and n_d are sufficiently large, then with high probability, the initialization satisfies

$$\begin{aligned} & \left| \mathbf{u}_0^{(1),p\top} \mathbf{X} \mathbf{u}_0^{(3),k} \right| = \Theta(1), \\ & \left| \mathbf{u}_0^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_0^{(1),p} \right| = \Theta(n_d), \\ & \left| \mathbf{u}_0^{(3),k\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_0^{(3),k} \right| = \Theta(n), \\ & \left\| \mathbf{u}_0^{(2),q} \right\|_2 = \Theta(1), \quad \mathbf{v}_0^\top \mathbf{u}_0^{(2),q} = \Theta(1), \end{aligned} \tag{13}$$

for $p \in [r_1]$, $q \in [r_2]$, and $k \in [r_3]$. To achieve the efficient training of $g^{p,q,k}$ (i.e., $\delta_{t,0}^{p,q,k}$) in Definition 1, we require the condition to hold at $t = 2$, equivalently, $\eta_0 = \Theta(1)$. Under this setting, we obtain $\left| g_1^{p,q,k} \right| = \Theta(1)$. If all learning rates are set to the same order of magnitude, we have $\|\delta_{t,1}^p\|_2 = \Theta(n_d)$ and $\|\delta_{t,3}^k\|_2 = \Theta(n)$ (with $t = 2$), which contradicts the conditions required for efficient training. Furthermore, if we assume that relations in Equation (13) hold for $t \geq 0$, then we can similarly deduce that $\eta_0 = \Theta(1)$, leading to the violation of conditions for efficient training.

Now, we begin to verify that setting learning rates as $\eta_0 = \Theta(1)$, $\eta_1 = \Theta(n^{-1})$, $\eta_2 = \Theta(1)$, and $\eta_3 = \Theta(n_d^{-1})$, ensures

the satisfaction of the conditions of efficient training. We first prove by induction that Equation (13) holds for all $t \geq 0$, i.e.,

$$\begin{aligned} & \left| \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{u}_t^{(3),k} \right| = \Theta(1), \\ & \left| \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_t^{(1),p} \right| = \Theta(n_d), \\ & \left| \mathbf{u}_t^{(3),k\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_t^{(3),k} \right| = \Theta(n), \\ & \left\| \mathbf{u}_t^{(2),q} \right\|_2 = \Theta(1), \quad \mathbf{v}_t^\top \mathbf{u}_t^{(2),q} = \Theta(1), \end{aligned} \quad (14)$$

for $p \in [r_1]$, $q \in [r_2]$, $k \in [r_3]$, and $t \geq 0$. Suppose that those relations in Equation (14) are satisfied at $t \geq 0$. First, we analyze the update of $g_{t+1}^{p,q,k}$ with

$$\begin{aligned} & \left| g_{t+1}^{p,q,k} - g_t^{p,q,k} \right| \\ &= \eta_0 \cdot \left| \mathbf{v}_t^\top \mathbf{u}_t^{(2),q} \right| \cdot \left| \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{u}_t^{(3),k} \right| \\ &= \Theta(1). \end{aligned}$$

Therefore, we have $\left| g_{t+1}^{p,q,k} \right| = \Theta(1)$, for $p \in [r_1]$, $q \in [r_2]$, and $k \in [r_3]$.

For $\mathbf{u}_{t+1}^{(1),p}$, we obtain

$$\begin{aligned} \left\| \Delta \mathbf{u}_{t+1}^{(1),p} \right\|_2 &= \left\| \mathbf{u}_{t+1}^{(1),p} - \mathbf{u}_t^{(1),p} \right\|_2 \\ &\leq \eta_1 \cdot \sum_{q'=1}^{r_2} \sum_{k'=1}^{r_3} \left| g_t^{p,q',k'} \right| \cdot \left| \mathbf{v}_t^\top \mathbf{u}_t^{(2),q'} \right| \cdot \left\| \mathbf{X} \mathbf{u}_t^{(3),k'} \right\|_2 \\ &= \Theta\left(n^{-1/2}\right), \end{aligned}$$

then

$$\begin{aligned} & \left| \mathbf{u}_{t+1}^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_{t+1}^{(1),p} - \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_t^{(1),p} \right| \\ &\leq 2 \left\| \Delta \mathbf{u}_{t+1}^{(1),p} \right\|_2 \cdot \left\| \mathbf{X} \right\|_2 \cdot \sqrt{\left| \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_t^{(1),p} \right|} \\ &\quad + \left\| \Delta \mathbf{u}_{t+1}^{(1),p} \right\|_2^2 \cdot \left\| \mathbf{X} \right\|_2^2 \\ &\leq \Theta(n_d), \end{aligned}$$

which guarantees that $\left| \mathbf{u}_{t+1}^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_{t+1}^{(1),p} \right| = \Theta(n_d)$. Similarly, we have

$$\begin{aligned} & \left\| \Delta \mathbf{u}_{t+1}^{(3),k} \right\|_2 \\ &= \left\| \mathbf{u}_{t+1}^{(3),k} - \mathbf{u}_t^{(3),k} \right\|_2 \\ &= \eta_3 \cdot \sum_{p'=1}^{r_1} \sum_{q'=1}^{r_2} \left| g_t^{p',q',k} \right| \cdot \left| \mathbf{v}_t^\top \mathbf{u}_t^{(2),q'} \right| \cdot \left\| \mathbf{X}^\top \mathbf{u}_t^{(1),p'} \right\|_2 \\ &= \Theta\left(n_d^{-1/2}\right), \end{aligned}$$

and

$$\begin{aligned} & \left| \mathbf{u}_{t+1}^{(3),k\top} \mathbf{X}^\top \mathbf{X} \mathbf{u}_{t+1}^{(3),k} - \mathbf{u}_t^{(3),k\top} \mathbf{X}^\top \mathbf{X} \mathbf{u}_t^{(3),k} \right| \\ &\leq 2 \left\| \Delta \mathbf{u}_{t+1}^{(3),k} \right\|_2 \cdot \left\| \mathbf{X} \right\|_2 \cdot \sqrt{\left| \mathbf{u}_t^{(3),k\top} \mathbf{X}^\top \mathbf{X} \mathbf{u}_t^{(3),k} \right|} \\ &\quad + \left\| \Delta \mathbf{u}_{t+1}^{(3),k} \right\|_2^2 \cdot \left\| \mathbf{X} \right\|_2^2 \\ &\leq \Theta(n_\ell), \end{aligned}$$

which ensures that $\left| \mathbf{u}_{t+1}^{(3),k\top} \mathbf{X}^\top \mathbf{X} \mathbf{u}_{t+1}^{(3),k} \right| = \Theta(n)$. Moreover,

$$\begin{aligned} & \left| \mathbf{u}_{t+1}^{(1),p\top} \mathbf{X} \mathbf{u}_{t+1}^{(3),k} - \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{u}_t^{(3),k} \right| \\ &\leq \left\| \Delta \mathbf{u}_{t+1}^{(1),p} \right\|_2 \cdot \sqrt{\left| \mathbf{u}_t^{(3),k\top} \mathbf{X}^\top \mathbf{X} \mathbf{u}_t^{(3),k} \right|} \\ &\quad + \left\| \Delta \mathbf{u}_{t+1}^{(3),k} \right\|_2 \cdot \sqrt{\left| \mathbf{u}_t^{(1),p\top} \mathbf{X} \mathbf{X}^\top \mathbf{u}_t^{(1),p} \right|} \\ &\quad + \left\| \Delta \mathbf{u}_{t+1}^{(1),p} \right\|_2 \cdot \left\| \Delta \mathbf{u}_{t+1}^{(3),k} \right\|_2 \cdot \left\| \mathbf{X} \right\|_2 \\ &= \Theta(1), \end{aligned}$$

we have $\mathbf{u}_{t+1}^{(1),p\top} \mathbf{X} \mathbf{u}_{t+1}^{(3),k} = \Theta(1)$. Finally, we consider $\mathbf{u}_t^{(2)}$ with

$$\begin{aligned} & \left\| \Delta \mathbf{u}_{t+1}^{(2),q} \right\|_2 \\ &= \left\| \mathbf{u}_{t+1}^{(2),q} - \mathbf{u}_t^{(2),q} \right\|_2 \\ &= \eta_2 \cdot \sum_{p'=1}^{r_1} \sum_{k'=1}^{r_3} \left| g_{t-1}^{p',q,k'} \right| \cdot \left| \mathbf{u}_{t-1}^{(1),p'\top} \mathbf{X} \mathbf{u}_{t-1}^{(3),k'} \right| \cdot \left\| \mathbf{v}_{t-1} \right\|_2 \\ &= \Theta(1), \end{aligned}$$

which implies $\left\| \mathbf{u}_{t+1}^{(2),q} \right\|_2 = \Theta(1)$. Note that

$$\Delta \Psi_{t+1,\text{ft}} = \sum_{q=1}^{r_2} a_{t,q} \mathbf{u}_t^{(2),q} + c_t \mathbf{v}_t,$$

with some numbers $a_{t,q} = \Theta(1)$ and $c_t = \Theta(1)$, then we have

$$\begin{aligned} \mathbf{v}_{t+1}^\top \mathbf{u}_{t+1}^{(2),q} &= (\mathbf{v}_t + \Delta \Psi_{t+1,\text{ft}})^\top \left(\mathbf{u}_t^{(2),q} + \Delta \mathbf{u}_{t+1}^{(2),q} \right) \\ &= \mathbf{v}_t^\top \mathbf{u}_t^{(2),q} + \Theta(1) \\ &= \Theta(1). \end{aligned}$$

Therefore, all conditions in Equation (14) hold for $t+1$ as well. By induction, it follows that these conditions are satisfied for all t .

Under the conditions in Equation (14) and the learning rates setup, where $\eta_0 = \Theta(1)$, $\eta_1 = \Theta(n^{-1})$, $\eta_2 = \Theta(1)$, and $\eta_3 = \Theta(n_d^{-1})$, we can verify that the quantities $\left\{ \left\| \delta_{t,0}^{p,q,k} \right\|_2 \right\}$, $\left\{ \left\| \delta_{t,1}^p \right\|_2 \right\}$, $\left\{ \left\| \delta_{t,2}^q \right\|_2 \right\}$, and $\left\{ \left\| \delta_{t,3}^k \right\|_2 \right\}$ are all of order $\Theta(1)$ with respect to the model size (n, m, n_d) , for all $i \in \{0, 1, 2, 3\}$, $p \in [r_1]$, $q \in [r_2]$, $k \in [r_3]$, and $t \geq 2$.

REFERENCES

- [1] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov–Arnold networks," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=Ozo7qJ5vZi>
- [2] C. Li, X. Liu, W. Li, C. Wang, H. Liu, Y. Liu, Z. Chen, and Y. Yuan, "U-KAN makes strong backbone for medical image segmentation and generation," *arXiv preprint arXiv:2406.02918*, 2024.
- [3] V. A. Kich, J. A. Bottega, R. Steinmetz, R. B. Grando, A. Yorozu, and A. Ohya, "Kolmogorov-Arnold networks for online reinforcement learning," in *2024 24th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2024, pp. 958–963.
- [4] Y. Wang, J. Sun, J. Bai, C. Anitescu, M. S. Eshaghi, X. Zhuang, T. Rabczuk, and Y. Liu, "A physics-informed deep learning framework for solving forward and inverse problems based on Kolmogorov–Arnold Networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 433, p. 117518, 2025.
- [5] B. Jacob, A. A. Howard, and P. Stinis, "SPIKANs: Separable physics-informed Kolmogorov-Arnold networks," *arXiv preprint arXiv:2411.06286*, 2024.
- [6] K. Shukla, J. D. Toscano, Z. Wang, Z. Zou, and G. E. Karniadakis, "A comprehensive and fair comparison between MLP and KAN representations for differential equations and operator networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 431, p. 117290, 2024.
- [7] Y. Hou and D. Zhang, "A comprehensive survey on Kolmogorov Arnold networks (KAN)," *arXiv preprint arXiv:2407.11075*, 2024.
- [8] S. Somvanshi, S. A. Javed, M. M. Islam, D. Pandit, and S. Das, "A survey on Kolmogorov-Arnold network," *arXiv preprint arXiv:2411.06078*, 2024.
- [9] N. Firsov, E. Myasnikov, V. Lobanov, R. Khabibullin, N. Kazanskiy, S. Khonina, M. A. Butt, and A. Nikonorov, "HyperKAN: Kolmogorov–Arnold networks make hyperspectral image classifiers smarter," *Sensors (Basel, Switzerland)*, vol. 24, no. 23, p. 7683, 2024.
- [10] Q. Zhou, C. Pei, F. Sun, J. Han, Z. Gao, D. Pei, H. Zhang, G. Xie, and J. Li, "KAN-AD: Time series anomaly detection with Kolmogorov-Arnold networks," *arXiv preprint arXiv:2411.00278*, 2024.
- [11] Y. Wang, J. W. Siegel, Z. Liu, and T. Y. Hou, "On the expressiveness and spectral bias of KANs," *arXiv preprint arXiv:2410.01803*, 2024.
- [12] A. Mehrabian, P. M. Adi, M. Heidari, and I. Hacıhaliloglu, "Implicit neural representations with Fourier Kolmogorov-Arnold networks," *arXiv preprint arXiv:2409.09323*, 2024.
- [13] A. A. Howard, B. Jacob, S. H. Murphy, A. Heinlein, and P. Stinis, "Finite basis Kolmogorov-Arnold networks: domain decomposition for data-driven and physics-informed problems," *arXiv preprint arXiv:2406.19662*, 2024.
- [14] S. Rigas, M. Papachristou, T. Papadopoulos, F. Anagnostopoulos, and G. Alexandridis, "Adaptive training of grid-dependent physics-informed Kolmogorov-Arnold networks," *IEEE Access*, 2024.
- [15] A. Kundu, A. Sarkar, and A. Sadhu, "Kanas: Kolmogorov-Arnold network for quantum architecture search," *EPJ Quantum Technology*, vol. 11, no. 1, p. 76, 2024.
- [16] A. D. Bodner, A. S. Tepsich, J. N. Spolski, and S. Pourceau, "Convolutional Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.13155*, 2024.
- [17] I. Drokina, "Kolmogorov-Arnold convolutions: Design principles and empirical studies," *arXiv preprint arXiv:2407.01092*, 2024.
- [18] S. SS, K. AR, A. KP *et al.*, "Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation," *arXiv preprint arXiv:2405.07200*, 2024.
- [19] Anonymous, "Legendre-KAN : High accuracy KA network based on Legendre polynomials," 2025. [Online]. Available: <https://openreview.net/forum?id=Bb1ddVX8rL>
- [20] J. Xu, Z. Chen, J. Li, S. Yang, W. Wang, X. Hu, and E. C.-H. Ngai, "FourierKAN-GCF: Fourier Kolmogorov-Arnold network—an effective and efficient feature transformation for graph collaborative filtering," *arXiv preprint arXiv:2406.01034*, 2024.
- [21] Z. Bozorgasl and H. Chen, "Wav-KAN: Wavelet Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.12832*, 2024.
- [22] S. T. Seydi, "Exploring the potential of polynomial basis functions in Kolmogorov-Arnold networks: A comparative study of different groups of polynomials," *arXiv preprint arXiv:2406.02583*, 2024.
- [23] A. A. Aghaei, "rKAN: Rational kolmogorov-arnold networks," *arXiv preprint arXiv:2406.14495*, 2024.
- [24] —, "fKAN: Fractional Kolmogorov-Arnold networks with trainable Jacobi basis functions," *Neurocomputing*, p. 129414, 2025.
- [25] L. Zhuang, X. Fu, M. K. Ng, and J. M. Bioucas-Dias, "Hyperspectral image denoising based on global and nonlocal low-rank factorizations," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 12, pp. 10438–10454, 2021.
- [26] J. Xue, Y. Zhao, W. Liao, and J. C.-W. Chan, "Nonlocal low-rank regularized tensor decomposition for hyperspectral image denoising," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 7, pp. 5174–5189, 2019.
- [27] T. Yokota, Q. Zhao, and A. Cichocki, "Smooth PARAFAC decomposition for tensor completion," *IEEE Transactions on Signal Processing*, vol. 64, no. 20, pp. 5423–5436, 2016.
- [28] X. Chen and L. Sun, "Bayesian temporal factorization for multidimensional time series prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4659–4673, 2021.
- [29] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.
- [30] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.
- [31] Y. Shen, B. Baingana, and G. B. Giannakis, "Tensor decompositions for identifying directed graph topologies and tracking dynamic networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 14, pp. 3675–3687, 2017.
- [32] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in Measuring Change*, vol. 15, no. 122-137, p. 3, 1963.
- [33] H. A. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 14, no. 3, pp. 105–122, 2000.
- [34] G. Song, M. K. Ng, and X. Zhang, "Robust tensor completion using transformed tensor singular value decomposition," *Numerical Linear Algebra with Applications*, vol. 27, no. 3, p. e2299, 2020.
- [35] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 2011.
- [36] J. D. Carroll, S. Pruzansky, and J. B. Kruskal, "CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters," *Psychometrika*, vol. 45, no. 1, pp. 3–24, 1980.
- [37] R. A. Harshman *et al.*, "PARAFAC2: Mathematical and technical notes," *UCLA Working Papers in Phonetics*, vol. 22, no. 3044, p. 122215, 1972.
- [38] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.
- [39] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.
- [40] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2021.
- [41] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation with multiple sources," *Advances in Neural Information Processing systems*, vol. 21, 2008.
- [42] H. Daumé III, "Frustratingly easy domain adaptation," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, A. Zaenen and A. van den Bosch, Eds. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 256–263. [Online]. Available: <https://aclanthology.org/P07-1033/>
- [43] E. J. Hu, Yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022. [Online]. Available: <https://openreview.net/forum?id=nZeVKeeFYf9>
- [44] S. Hayou, N. Ghosh, and B. Yu, "LoRA+: Efficient low rank adaptation of large models," in *International Conference on Machine Learning*, 2024. [Online]. Available: <https://openreview.net/forum?id=NEv8YqBR00>
- [45] A. Aghajanyan, L. Zettlemoyer, and S. Gupta, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," *arXiv preprint arXiv:2012.13255*, 2020.
- [46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
- [47] R. Yu, W. Yu, and X. Wang, "KAN or MLP: A fairer comparison," *arXiv preprint arXiv:2407.16674*, 2024.