[Team 12] Final Report
LoKi: Low-dimensional KAN for Efficient Fine-tuning Image Models

Yoonseo Gu
Department of Data Science
2024320362
gys0821@korea.ac.kr

Hyunseo Son
Division of Life Sciences
2020140012
soner2009@korea.ac.kr

Youngjun Ji
Department of Computer Science and Engineering
2021320070
jiyj1080@korea.ac.kr

1. Introduction

This report is our analysis of the paper "LoKi: Low-dimensional KAN for Efficient Fine-tuning Image Models"[1], focused on its novelty, methodology, claims, and critiques.

1.1. Purposed problem

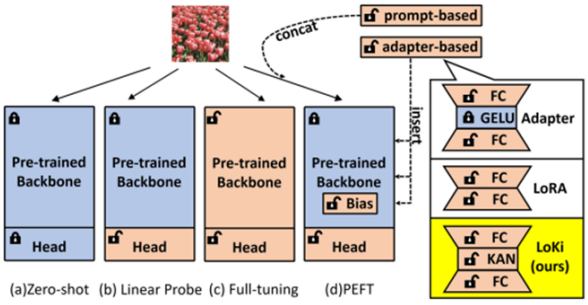

Figure 1. Common fine-tuning methods of pre-trained models[1]

One of the current trend in computer vision(CV) models is fine-tuning universal pre-trained model for downstream tasks.

Especially, PEFT(Parameter Efficient Fine Tuning), the method that limits the number of parameters that need to be tuned, is getting attention for its efficiency and ensured performance.

However, existing PEFT methods based on MLPs (e.g. adapter, LoRA) have some limitations:

- First, the fixed activation function (e.g. ReLU) limits expressiveness of the model, therefore requires extensive linear combinations.
- Second, these methods are showing high bottleneck ratios, which causes a generalization problem for various downstream tasks.

1.2. The novelty of the paper

To address these problems, the authors proposed LoKi, the adapter-based method with low rank, sandwiched Kolmogorov-Arnold network(KAN) structure.

The authors suggested the advantages of LoKi from two perspectives:

- Compared to existing MLP-based PEFT methods, the learnable activation functions of LoKi could allow for more flexibility with the same or even a smaller number of parameters than fully connected layers(achieving a lower bottleneck ratio).
- Compared to plain KAN, the size of the KAN layer is minimized in LoKi, so it could avoid slow training speed and high memory consumption.

1.3. Implemented tasks and conducted experiments

The authors applied LoKi at the pre-trained ViT model in adapter-like manner, to carry out image classification and video action recognition. For video action recognition, the authors add frame-related components (time embedding, time attention, and time LoKi layers) to try utilizing the image ViT model for video tasks.

Then, the authors compared its performance with other models, including PEFT methods and full-tuning, on various datasets.

1.4. The Pros and Cons claimed by the authors

According to the results, the authors claimed that LoKi showed the advantages such that:

- LoKi outperformed other PEFT and full-tuning methods on the most of the datasets, with the significantly small amount of parameters that need to be tuned.
- LoKi showed clearer attention maps on the image classification task compared to other PEFT and full-tuning methods.

And the authors discussed about potential shortcomings such that:

- Low scalability: On a large dataset, LoKi underperformed a simple adapter-based model. The authors suggested the reason, was that due to the large size of the dataset, the overfitting-prone nature of the adapter had diluted.
- Tightness of the parameter space: LoKi has a smaller amount of parameters, which can limit the ability to adapt multiple downstream tasks.

1.5. Our critiques and suggestions

We discovered another potential shortcoming of LoKi that the authors did not mentioned: computational costs.

Despite of limited size of the KAN layer in LoKi, the high computational cost of its activation function (B-spline) may require longer time to train or infer compared to MLP-based models. But the authors only argued the advantage of LoKi with respect to only the number of parameters, not to actual computational costs.

So we suggest some improvements to reduce costs:

1. **Dimension reduction**: By introducing dimension-manipulating methods like matrix factorization or 1x1 convolution, we can reduce the input dimension of LoKi, and thus reduce the number of parameters further.
2. **Regularized KAN**: By implementing regularization on training KAN layer, we can simplify the model and can speed up the inference speed of a distributed model.

2. Related works

2.1. Image pre-trained models

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1. The size of ViT models proposed[2]

The popular choices for the pre-trained model in CV tasks are ViTs and its variants. In the paper, the authors chose ViT-B/16. ViT-B/16 is the smallest model proposed at the reference paper[2], with the patch size 16x16. The authors used both of pre-trained weights of ViT(IN21K)[2] and CLIP(ViT-B/16)[7].

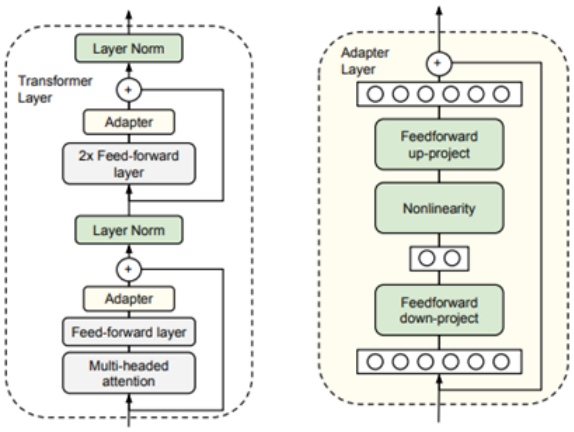

Figure 2. insertion of adapter layer on pre-trained model[3]

$$h = w_0 x + \Delta x = W_0 x + BAx$$

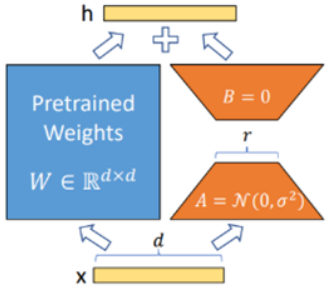

Figure 3. summation of decomposing weights with pre-trained weights[4]

2.2. PEFT

2.2.1. PEFT methods

There are some representative PEFT methods:

1. adapter-based methods: inserts bottleneck structure into the pre-trained model
   - adapter[3]: inserts MLP
   - LoRA(low-rank adaption)[4]: add weights generated by low-rank decomposition(mathematically, a single linear layer)
2. prompt tuning[5]: add learnable parameters at only a input layer
3. bias tuning(e.g. Bitfit)[10]: select existing bias parameters to tune, instead of adding new parameters

2.2.2. Adapting pre-trained image models for video tasks

There are several works tried to adapt pre-trained iamge models for video tasks, with PEFT. For example, AIM[8] is a video action recognition model, based on pre-trained ViT and using adapter for fine-tuning.

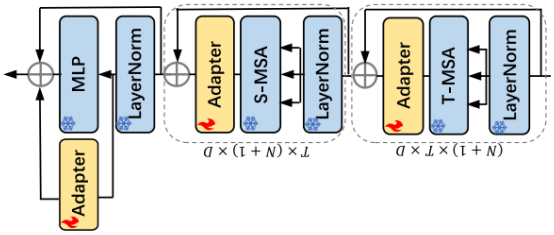

Figure 4. The structure of AIM[8]

2.3. KAN

2.3.1. Kolmogorov-Arnold representation theorem

Kolmogorov-Arnold representation theorem is stated like: For a arbitrary smooth $f : [0, 1]^n \to \mathbb{R}$

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

where $\phi_{q,p} : [0, 1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$
This theorem states that with a 2-layer combination of univariate functions, we can approximate a arbitrary smooth multivariate function.

But for machine learning perspective, this theorem is quite meaningless, because of no restrictions on ϕ, Φ, whether the model adopted the theorem is learnable or optimizable is not guaranteed.

2.3.2. The structure of KAN

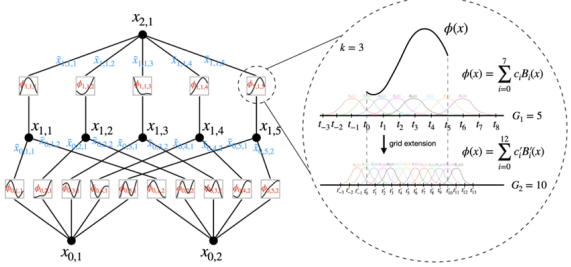

Figure 5. The structure of KAN[6]

The main ideas constructing KAN based on are:

- Generalize the width and depths of the network compared to the statement of the original theorem
- Adopt a linear combination of B-splines with learnable coefficients for ϕ

In detail, for the l-th layer of total L layers $\mathbf{x}_l = \{x_{l,i}\}$ $i = 1, 2, \cdots, n_l$ and learnable activation functions $\phi_{l,j,i}$

$$x_{l+1,j} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i})$$

In matrix form,

$$\mathbf{x}_{l+1} = \begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \cdots & \cdots & \ddots & \cdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix} \mathbf{x}_l = \mathbf{\Phi}_l \mathbf{x}_l$$

where $\mathbf{\Phi}_l = \{\phi_{l,j,i}\}$ $\quad i = 1, \cdots, n_l, \quad j = 1, \cdots, n_{l+1}$

Then the output of KAN is

$$\text{KAN}(\mathbf{x}) = (\mathbf{\Phi}_{L-1} \circ \mathbf{\Phi}_{L-2} \circ \cdots \circ \mathbf{\Phi}_1 \circ \mathbf{\Phi}_0)\mathbf{x}$$

Each activation functions $\phi_{l,j,i}$ are defined as

$$\phi(x) = w_b b(x) + w_s \text{spline}(x)$$

where

$$b(x) = \text{silu}(x) = \frac{x}{1 + e^{-x}}$$

$$\text{spline}(x) = \sum_{i}^{G+K-1} c_i B_i(x)$$

where w_b, w_s, c_i's are learnable parameters.

$B_i(x) = B_{i,k}(x)$ is a B-spline basis function, computed with the De Boor's algorithm.
For the control points

$$t_i = -1 + (i/G) \times 2$$
$$i = -K, -K+1, \cdots, 0, 1, \cdots, G, G+1, \cdots, G+K$$

$$B_{i,0} := \begin{cases} 1 & \text{if } t_i \le x < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,p}(x) := \frac{x - t_i}{t_{i+p} - t_i} B_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(x)$$

where G, K's are hyperparameters.

G is the number of spline intervals(grids), and K is the degree of splines. Usually $K = 3$ to ensure C2 continuity.

2.3.3. Several works implementing KAN

There are several works tried to utilize KAN for various tasks, by directly replacing some components of well-known models. For instance, KAT[9] is a model that MLP components of the transformer are replaced with KAN.

3. Details

3.1. Methodology: LoKi Module Architecture

LoKi is an adapter-style module designed to introduce learnable non-linearity into pre-trained models with high

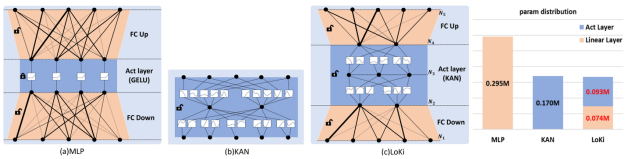

Figure 6. Structural comparison and parameter distribution of MLP, KAN, and LoKi modules[1]

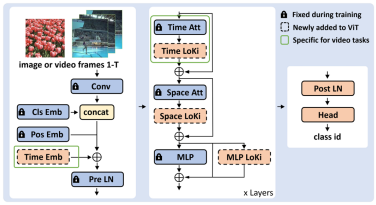

Figure 7. Integration of LoKi into the Vision Transformer architecture[1]

parameter efficiency. The architecture consists of three sequential components: an encoder, a learnable activation layer, and a decoder.

Encoder and Decoder: To construct a bottleneck structure, single-layer linear networks are utilized for both the encoder and decoder. The encoder (W_{down}) projects the input feature $\mathbf{x} \in \mathbb{R}^D$ to a lower-dimensional space $\mathbb{R}^d$, where $d = r_1 \times D$. Here, r_1 is the bottleneck ratio, which is set significantly lower than standard adapters (e.g., $r_1 \approx 0.0625$) to minimize parameter count. Conversely, the decoder (W_{up}) projects the features back to the original dimension D.

Learnable Activation Layer: Unlike traditional adapters that use fixed activation functions (e.g., GELU), LoKi employs a learnable activation layer based on a minimal Kolmogorov-Arnold Network (KAN). This layer consists of two KAN linear layers (Φ_0, Φ_1) to approximate complex non-linear mappings.

The operation of LoKi can be formulated as:

$$\text{LoKi}(\mathbf{X}) = (\mathbf{W}_1 \circ \Phi_1 \circ \Phi_0 \circ \mathbf{W}_0)\mathbf{X}$$

Crucially, the non-linear transformation function ϕ within the KAN layers is defined as a combination of a base activation and a learnable B-spline term:

$$\phi(x) = \text{SiLU}(x) + \sum_i c_i B_i(x)$$

where $\text{SiLU}(x) = x/(1 + e^{-x})$ serves as the base function, $B_i(x)$ represents the B-spline basis functions, and c_i are learnable coefficients. This design decouples the activation function from the linear weights, allowing the model to fit target non-linear functions more flexibly without increasing the depth of the linear layers[1].

3.2. Integration into Vision Transformers

For Vision Transformer (ViT) backbones, LoKi is integrated into each transformer block to modulate feature representations. The module is inserted **after the self-attention layer** and operates **alongside the frozen MLP block**. The output of the LoKi module is added to the output of the MLP block via a residual connection:

$$\mathbf{x}_{out} = \mathbf{x}_{in} + \text{MLP}(\mathbf{x}_{in}) + f_{LoKi}(\mathbf{x}_{in})$$

This parallel configuration allows LoKi to learn task-specific features and non-linearities independently, preserving the pre-trained knowledge encapsulated in the frozen MLP weights. For video tasks, a temporal variant (Time LoKi) is additionally inserted to capture time-dimension features, sharing the same architectural principles[1].

3.3. Experimental Results

The authors evaluate LoKi on image classification and video action recognition using the ViT-B/16 backbone. When compared to standard PEFT baselines (including Adapter, LoRA, and Linear Probe), LoKi achieves the highest average Top-1 accuracy across seven datasets. It demonstrates superior generalization stability, particularly on datasets with significant distribution shifts (e.g., medical imaging), where traditional adapters tend to overfit.

In video action recognition tasks, the temporal variant (Time LoKi) matches or outperforms state-of-the-art methods like AIM while requiring only 56.9% of the tunable parameters. Furthermore, ablation studies confirm that LoKi converges faster and exhibits greater training stability than vanilla KANs, maintaining high efficiency with approximately 56.7% of the parameters of standard Adapters[1].

4. Critiques

While LoKi demonstrates impressive performance as a parameter-efficient fine-tuning method, several limitations warrant further discussion.

First, the paper emphasizes efficiency primarily in terms of parameter count and training memory, but does not report inference-time metrics such as FLOPs or latency. Since KAN-based B-spline operations may introduce irregular memory access patterns and higher computational overhead than standard MLPs, parameter efficiency does not necessarily imply faster inference. A comparative evaluation of runtime performance would be necessary to fully validate LoKi's efficiency claims.

Second, LoKi exhibits limited scalability on large-scale datasets. The authors report that on datasets such as Kinetics-400 and SSv2, LoKi underperforms compared to standard Adapter methods. This suggests that LoKi functions primarily as a strong regularizer in low-data regimes,

potentially limiting its representational capacity when abundant data is available.

Finally, although LoKi is described as a plug-and-play method requiring minimal tuning, the internal KAN hyperparameters (grid size G and spline order K) are fixed throughout all experiments. The paper does not include a sensitivity analysis to verify whether these hyperparameters are robust across datasets with different characteristics. A sensitivity analysis would further strengthen the claim that LoKi requires no additional tuning.

5. Suggestions

Based on our critiques of the LoKi architecture and its limitations, we propose key improvements to enhance its efficiency and generalization capabilities.

5.1. Factorized LoKi

While LoKi significantly reduces the number of trainable parameters compared to standard Adapters by utilizing a bottleneck structure, the linear projection layers (Encoder and Decoder) still rely on dense matrix multiplication. According to the parameter calculation[1],

$$N_L = 2r_1 N_1(N_1 + 1)$$

the number of parameters in the linear part, denoted as N_L, is proportional to the square of the input dimension (N_1^2). This quadratic term dominates the total parameter count, limiting further efficiency gains.

To address this limitation and push parameter efficiency to the limit, we propose **Factorized LoKi**. This approach applies Low-Rank Decomposition to the dense linear layers of the Encoder and Decoder. Instead of using a full-rank weight matrix, we decompose it into two low-rank matrices, where the rank is significantly smaller than the input/output dimensions. And the decomposition can be formulated as:$W \approx A \cdot B$

By incorporating this factorization directly into the LoKi module, we can reduce the parameter complexity. This modification allows LoKi to achieve extreme parameter efficiency, making the model suitable for deployment on highly resource-constrained edge devices without significantly compromising the expressiveness provided by the central KAN layer.

5.2. Conv-LoKi

The current implementation of LoKi is tailored for Vision Transformers (ViT), specifically designed to work alongside MLP blocks. However, Convolutional Neural Networks (CNNs) such as ResNet are still widely used in various computer vision applications.

We suggest extending the applicability of LoKi by introducing **Conv-LoKi**. Since a 1×1 convolution is mathematically equivalent to a linear projection across the channel dimension, we can replace the linear layers in LoKi with 1×1 convolutional layers. This adaptation would allow LoKi to be integrated into CNN architectures as a plug-and-play module, providing a learnable activation mechanism to spatial feature maps and broadening the impact of the proposed method beyond Transformer-based models.

5.3. Regularized KAN

According to the reference paper of KAN[6], a regularized loss function for KAN is suggested like:

$$l_{\text{total}} = l_{\text{pred}} + \lambda \left(\mu_1 \sum_{l=0}^{L-1} |\Phi_l|_1 + \mu_2 \sum_{l=0}^{L-1} S(\Phi_l) \right)$$

where

$$\text{L1 norm of } \phi : |\phi_{l,j,i}|_1 \equiv \frac{1}{N} \sum_{s=1}^{N} \phi_{l,j,i}(x_{l,i}^{(s)}) = \text{avg}(\phi_{l,j,i})$$

$$\text{L1 norm of } \Phi : |\Phi_l|_1 \equiv \sum_{j=1}^{n_{l+1}} \sum_{i=1}^{n_l} |\phi_{l,j,i}|_1$$

$$\text{Entropy of } \Phi : S(\Phi_l) \equiv -\sum_{j=1}^{n_{l+1}} \sum_{i=1}^{n_l} \frac{|\phi_{l,j,i}|_1}{|\Phi_l|_1} \log \frac{|\phi_{l,j,i}|_1}{|\Phi_l|_1}$$

With this regularization, the KAN tends to be learned sparsely(less nodes and active functions matters), and the KAN can be pruned after training to create a simpler network.

We suggest to utilize this regularization & pruning method during training process of LoKi to simplify a model and reduce the inference time of the distributed version of the model.

But we should consider possible shortcomings:

- Generalization problem caused by increased bottleneck ratio
- Computational cost during training process

A. Appendix

A.1. Answers to Additional Questions from Peers

1. **The paper suggests that LoKi prevents models from overfitting. What specific features of LoKi make this possible?**
   LoKi achieves this by decoupling the learnable activation layer (KAN) from the linear projection layers. Unlike standard Adapters, which typically require a high bottleneck ratio ($r_1 \geq 0.5$) to maintain performance, LoKi employs a significantly lower bottleneck ratio ($r_1 \approx 0.0625$)[1]. This design drastically reduces the parameter count in the linear layers while relying on the KAN layer for efficient nonlinear fitting, effectively preventing overfitting on small downstream datasets.

Type	Method	Tunable Params(M)	Mem (G)	CIFAR 10	Food 101	CIFAR 100	MNIST	DTD	FGVC Aircraft	DDSM	Average
-	Zero-shot	-	-	91.6 -	89.2 -	68.7 -	56.0 -	46.0 -	27.1 -	9.2 -	- -
	Full-tuning	86	5.9	70.0 (53.6)	47.0 (75.0)	52.8 (67.9)	91.9 (15.0)	38.4 (48.9)	40.7 (67.3)	53.1 (4.7)	56.2 (47.5)
	Linear Probe	<1	2.4	94.9 (3.0)	87.9 (7.1)	80.8 (7.2)	95.7 (2.8)	73.4 (5.6)	47.2 (19.8)	56.6 (1.6)	76.6 (6.7)
PEFT	Bias Tuning [87]	<1	4.5	98.4 (0.7)	90.3 (4.8)	89.9 (3.9)	99.0 (0.4)	76.9 (4.5)	69.8 (11.0)	67.4 (5.6)	84.5 (4.4)
	Prompt Tuning [33]	<1	6.4	97.6 (0.7)	88.7 (6.6)	87.8 (4.2)	98.3 (0.8)	73.6 (2.4)	63.0 (20.9)	61.1 (2.4)	81.4 (5.4)
	LoRA [31]	<1	4.7	97.7 (1.6)	87.3 (1.0)	87.4 (5.3)	98.9 (0.3)	74.9 (5.2)	49.4 (29.5)	39.8 (3.3)	76.5 (6.6)
	Adapter [30]	7	4.8	86.7 (24.1)	63.3 (53.2)	65.1 (55.3)	98.6 (1.8)	47.8 (53.4)	46.5 (65.5)	67.5 (16.1)	67.9 (38.5)
	KAN [55]	4	32.9	80.2 (37.3)	56.3 (65.5)	60.6 (63.7)	97.3 (3.7)	48.5 (52.7)	43.7 (52.7)	67.0 (8.9)	64.8 (40.6)
	LoKi	**4**	**10.4**	98.1 (0.0)	89.4 (4.9)	89.1 (1.5)	99.3 (0.0)	76.0 (3.8)	71.6 (4.5)	74.2 (1.4)	85.4 (2.3)

Table 2. Comparison of accuracy and generalization among different fine-tuning methods on various image datasets. The zero-shot accuracy is reported by[7] (except that DDSM is reproduced in our own), which is used to measure data distribution discrepancies between the downstream dataset and the pre-trained data. In the corresponding results for each dataset, the number on top represents the **average** top-1 acc obtained using IN21K and CLIP pre-trained weights. The number in parentheses below represents the **absolute difference** in top-1 acc between the two pre-trained weights, which measures the generalization ability. Red represents the best results, and blue represents the second-best results.

2. **How does the decoder in LoKi ensure that the feature dimensions of LoKi are consistent with the pre-trained model? Why is this process necessary?**
   The decoder consists of a single linear layer that projects the features from the internal dimension of the activation layer back to the original hidden dimension (D) of the pre-trained backbone (i.e., $N_{out} = N_{in} = D$). This dimension alignment is necessary to perform an element-wise addition (residual connection) between the LoKi module's output and the pre-trained model's output, ensuring seamless integration into the existing architecture.

3. **What are the examples of the shortcomings of the classic methods such as Adapter and LoRA the authors mentioned they reproduced for the experiment in the paper?**
   The authors demonstrated that classic methods like Adapter and LoRA are prone to severe overfitting, particularly on datasets with data distributions distinct from the pre-training data.
   As shown in Tab. 2, LoKi demonstrates strong generalization and excellent fitting capabilities than other classic methods. On the datasets with the greatest distribution differences, DDSM and FGVC Aircraft, LoKi's accuracy and generalization are the best.

A.2. Each Teammate's Contribution

Team Member	Contribution Details
Yoonseo Gu	Wrote **Sec. 3 & 4**. Conducted in-depth LoKi architecture analysis and identified weaknesses.
Hyunseo Son	Wrote **Sec. 1 & 2 & 5.3**. Summarized PEFT/KAN background and established problem setup.
Youngjun Ji	Wrote **Sec. 5 & A**. Proposed potential improvements and compiled peer Q&A responses.

Table 3. Breakdown of contributions by each team member.

References

[1] Xuan Cai, Renjie Pan, and Hua Yang. Loki: Low-dimensional kan for efficient fine-tuning image models. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 14869–14880, 2025. 1, 4, 5

[2] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[3] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019. 2

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 2

[5] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European conference on computer vision*, pages 709–727. Springer, 2022. 2

[6] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024. 3, 5

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 2, 6

[8] Taojiannan Yang, Yi Zhu, Yusheng Xie, Aston Zhang, Chen Chen, and Mu Li. Aim: Adapting image models for efficient video action recognition. *arXiv preprint arXiv:2302.03024*, 2023. 2, 3

[9] Xingyi Yang and Xinchao Wang. Kolmogorov-arnold transformer. *arXiv preprint arXiv:2409.10594*, 2024. 3

[10] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, 2022. 2