TuckA: Hierarchical Compact Tensor Experts for Efficient Fine-Tuning

Qifeng Lei[1], Zhiyong Yang[1*], Qianqian Xu[2], Cong Hua[2], Peisong Wen[1], Qingming Huang[1,2*]

[1]School of Computer Science and Technology, University of Chinese Academy of Sciences
[2]State Key Laboratory of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences
leiqifeng24@mails.ucas.ac.cn, yangzhiyong21@mails.ucas.ac.cn, qmhuang@ucas.ac.cn

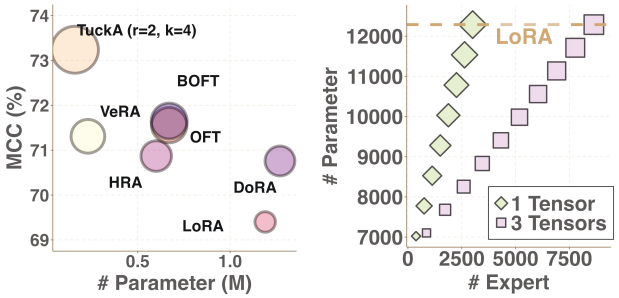
Abstract

Efficiently fine-tuning pre-trained models for downstream tasks is a key challenge in the era of foundation models. Parameter-efficient fine-tuning (PEFT) presents a promising solution, achieving performance comparable to full fine-tuning by updating only a small number of adaptation weights per layer. Traditional PEFT methods typically rely on a single expert, where the adaptation weight is a low-rank matrix. However, for complex tasks, the data's inherent diversity poses a significant challenge for such models, as a single adaptation weight cannot adequately capture the features of all samples. To address this limitation, we explore how to integrate multiple **small** adaptation experts into a compact structure to defeat a **large** adapter. Specifically, we propose Tucker Adaptation (TuckA), a method with four key properties: (i) We use Tucker decomposition to create a compact 3D tensor where each slice naturally serves as an expert. The low-rank nature of this decomposition ensures that the number of parameters scales efficiently as more experts are added. (ii) We introduce a hierarchical strategy that organizes these experts into groups at different granularities, allowing the model to capture both local and global data patterns. (iii) We develop an efficient batch-level routing mechanism, which reduces the router's parameter size by a factor of L compared to routing at every adapted layer (where L is the number of adapted layers) (iv) We propose data-aware initialization to achieve loss-free expert load balancing based on theoretical analysis. Extensive experiments on benchmarks in natural language understanding, image classification, and mathematical reasoning speak to the efficacy of TuckA, offering a new and effective solution to the PEFT problem.

Code — https://github.com/LQF39466/TuckA
Extended version — https://arxiv.org/pdf/2511.06859

1 Introduction

The emergence of large-scale pre-trained models (Radford et al. 2019; Bai et al. 2023; Liu et al. 2024a) has fundamentally reshaped the machine learning landscape. However, their ever-growing size makes full fine-tuning no longer affordable for most users. To address this challenge, Parameter-Efficient Fine-Tuning (**PEFT**) methods (Hu et al. 2022; Qiu et al. 2023; Guo et al. 2025; Hua et al. 2025; Han et al. 2025) have emerged. These methods adapt foundation models by training only a small fraction of their parameters while keeping the original model weights frozen, enabling efficient learning and lightweight deployment.



(a) **Performace Tradeoff** (b) **Parameter Scaling**

Figure 1: (a) The performance and parameter efficiency tradeoff of TuckA and other PEFT methods on the CoLA benchmark, where the top-left is ideal. (b) Parameter scaling under different configurations of TuckA's hierarchical multi-expert framework. The plot shows the number of trainable parameters as a function of expert numbers when adapting a single linear layer in DeBERTa-v3-base model.

Hitherto, mainstream PEFT methods have made significant progress by injecting a single, compact adaptation weight into layers of the pre-trained model. By adding or multiplying a low-rank matrix to the weights, approaches such as Low-Rank Adaptation (LoRA) (Hu et al. 2022) and Orthogonal Fine-Tuning (OFT) (Qiu et al. 2023) achieve leading parameter efficiency without performance compromises. Although effective, these single-expert approaches share a common weakness. For complex tasks, the inherent diversity of the data presents a substantial challenge, as a single adaptation weight cannot sufficiently capture the features of all samples. This representational bottleneck means that a single adapter may struggle to learn the varied patterns present in a dataset, limiting overall performance.

A simple solution is to combine multiple adapters into a Mixture-of-Experts (MoE) framework (Mu and Lin 2025; Hua et al. 2024), allowing different experts to specialize

*Corresponding Author

in different data. However, adding more experts increases the number of parameters, defeating the goal of PEFT. This leads to a key question: *Given the **same or even smaller** computational budget, how can we integrate **multiple small** adaptation experts into a compact structure to beat a **single large** expert?*

Seeking an answer to the question, we propose **Tucker Adaptation (TuckA)**, a novel PEFT framework that organizes multiple experts within a higher-order tensor, where elements are distributed in different columns, rows, and slices. This tensor-based design allows each slice of the tensor to act as a separate expert. Therefore, experts can scale up efficiently if the expression of the tensor itself is compact. Consequently, we employ the **Tucker decomposition** (Tucker 1966) to construct this compact 3D tensor adapter, where the core tensor leverages a low multi-linear rank across 3 dimensions, instead of 2, and achieves better scalability.

To capture features at different abstraction levels, we further design a **hierarchical multi-expert framework** based on the tensor structure. Experts are grouped, with each group forming an adaptation tensor. A proper routing scheme ensures that experts in the same group are selected together. This allows each group to capture global patterns, while individual experts within the group learn finer details. To keep the model scalable, the tensors across groups differ only by a small core tensor. With this hierarchical tensor-based framework, the number of experts can now scale efficiently. According to Figure 1, one can efficiently organize small experts to achieve better performance than using a large expert.

Building on this hierarchical setup, we propose a **batch-level routing** mechanism to reduce the computational cost of selection. Expert assignment is computed once per batch and shared across all adapted layers. However, this design increases the router's sensitivity to initialization. Through theoretical analysis based on high-dimensional probability, we demonstrate that standard methods, such as Kaiming uniform initialization (He et al. 2015), lead to imbalanced expert loads. To prevent expert collapse, we introduce a new **data-aware initialization** strategy that anchors expert centroids within the data manifold, ensuring balanced use from the beginning without relying on complex auxiliary losses.

We perform extensive experiments on a wide range of tasks, including natural language understanding (six tasks in GLUE), image classification (CIFAR-100, Food-101, Caltech-256), and mathematical reasoning (MATH, GSM-8K). Across multiple model architectures, TuckA consistently achieves state-of-the-art performance in the parameter-performance tradeoff, proving its efficiency and effectiveness.

Our contributions are threefold:

1. We propose **TuckA**, a novel tensor-based PEFT framework that leverages Tucker decomposition to create a compact ensemble of adaptation experts.
2. We develop a synergistic mechanism of **hierarchical routing** and a theoretically-grounded **data-aware initialization** strategy that together ensure efficient and stable expert activation.
3. Through **extensive empirical validation**, we demonstrate that TuckA significantly advances the state of the art, achieving superior performance with fewer parameters than existing methods.

2 Preliminaries

LoRA (Hu et al. 2022) and OFT (Qiu et al. 2023) are two influential lines of work in the field of PEFT. A recent paper proposed using Householder Reflection to construct the adaptation weights (Yuan, Liu, and Xu 2024). This Householder adapter can be viewed as both a multiplicative adapter and an additive adapter, which inspired this work. Refer to **Appendix D** for the related work.

Additive Adapters. LoRA and its variants (Zhang et al. 2023; Kopiczko, Blankevoort, and Asano 2024; Liu et al. 2024b) add its constructed matrix $\boldsymbol{T}_{\text{add}} \in \mathbb{R}^{d\times d'}$ to the original weights $\boldsymbol{W}$. The resulting weight matrix, $\boldsymbol{W}' \in \mathbb{R}^{d\times d'}$, is calculated as:

$$\boldsymbol{W}' = \boldsymbol{T}_{\text{add}} + \boldsymbol{W}.$$

Multiplicative Adapters. OFT and its variants (Liu et al. 2024c) apply an orthogonal transformation matrix $\boldsymbol{T}_{\text{orth}} \in \mathbb{R}^{d\times d}$ to the original weights $\boldsymbol{W}$. The resulting weight matrix, $\boldsymbol{W}' \in \mathbb{R}^{d\times d'}$, is calculated as:

$$\boldsymbol{W}' = \boldsymbol{T}_{\text{orth}}\boldsymbol{W}.$$

Householder Reflection Adaptation. Householder Reflection Adaptation (HRA) constructs $\boldsymbol{T}$ by applying Householder reflections multiple times, which yields:

$$\boldsymbol{T} = \prod_{i=1}^{r}(\boldsymbol{I} - 2\boldsymbol{u}_i\boldsymbol{u}_i^\top) = \left(\boldsymbol{I} + \boldsymbol{U}_r\boldsymbol{\Gamma}_r\boldsymbol{U}_r^\top\right), \qquad (1)$$

where $\boldsymbol{U}_r = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_r] \in \mathbb{R}^{d\times r}$ and $\boldsymbol{\Gamma}_r \in \mathbb{R}^{r\times r}$ is constructed recursively as follows:

$$\boldsymbol{\Gamma}_1 = -2,\ \boldsymbol{\Gamma}_r = \begin{bmatrix} \boldsymbol{\Gamma}_{r-1} & -2\boldsymbol{\Gamma}_{r-1}\boldsymbol{U}_{r-1}^\top\boldsymbol{u}_r \\ -2\boldsymbol{u}_r^\top & -2 \end{bmatrix}. \qquad (2)$$

In this sense, one can simultaneously adopt both additive and multiplicative modifications of the original weights.

3 Method Formulation

In this section, we introduce **TuckA**, an adapter architecture based on Tucker decomposition, designed to ensemble multiple low-rank adaptation experts while maintaining parameter efficiency. We show an overview of TuckA in Figure 2. We first formulate our tensor adapter, then propose grouped experts and their routing mechanism. Finally, we address the critical issues of initialization and load balancing.

3.1 Tensor Adapter as MoE

As mentioned in the introduction, our primary goal is to construct a tensor-based PEFT framework to leverage an efficient expert ensemble across its slices. To do this, inspired by the Householder adapter in HRA (Sec. 2), we employ a highly compressed tensor to capture both the additive and multiplicative characteristics. We observe that the

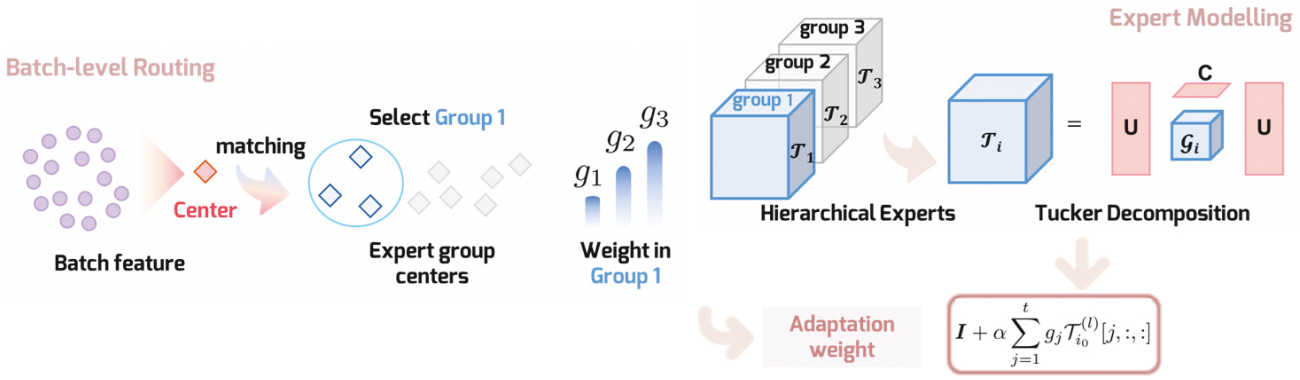


Figure 2: An overview of the proposed TuckA architecture. It is primarily composed of hierarchical experts constructed through Tucker decomposition and a batch-level routing mechanism, which determines how the experts are combined to produce the adaptation weight.

matrix form in HRA (Eq. 1) is analogous to Singular Value Decomposition (SVD), which approximates a matrix with a low-rank representation. This analogy naturally leads us to leverage the Tucker decomposition, a higher-order generalization of SVD, to achieve this.

When replacing $\mathbf{\Gamma}$ with a 3D tensor $\mathcal{G}$, and introducing a factor matrix $\boldsymbol{C}$ for the third dimension, our tensor adapter $\mathcal{T} \in \mathbb{R}^{t \times d \times d}$ is constructed as:

$$\mathcal{T} = \mathcal{G} \times_1 \boldsymbol{C} \times_2 \boldsymbol{U} \times_3 \boldsymbol{U}, \tag{3}$$

where:

- $\mathcal{G} \in \mathbb{R}^{p \times r \times r}$ is the core tensor. The tuple (p, r, r) is referred to as the **multilinear rank**.
- $\boldsymbol{C} \in \mathbb{R}^{t \times p}$ and $\boldsymbol{U} \in \mathbb{R}^{d \times r}$ denotes the factor matrix for a specific mode.
- $\times_n$ signifies the **mode-n product** between a tensor and a matrix. The entries of the resulting tensor $\mathcal{Y} = \mathcal{X} \times_n \boldsymbol{A}$ are calculated by multiplying the mode-n fibers of $\mathcal{X}$ with the rows of the matrix $\boldsymbol{A}$. Specifically, each element of $\mathcal{Y}$ is given by the formula:

$$(\mathcal{Y})_{i_1 i_2 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_n \ldots i_N} a_{j i_n},$$

where the index i_n is used for the summation over the n-th mode of $\mathcal{X}$, j is the new index for the n-th mode of $\mathcal{Y}$, I_n is the size of the n-th dimension of $\mathcal{X}$, and $a_{j i_n}$ is the element in the j-th row and i_n-th column of matrix $\boldsymbol{A}$.

In this formulation, the factor matrices $\boldsymbol{C}$ and $\boldsymbol{U}$ can be interpreted as the mode-specific principal components for the tensor $\mathcal{T}$. The core tensor $\mathcal{G}$ captures the interactions between these components. Akin to SVD, the multilinear rank (p, r, r) can be chosen to be substantially smaller than the original dimensions (t, d, d) while still capturing the salient latent patterns, owing to the inherent redundancy in the full parameter space.

To perform adaptation, the tensor $\mathcal{T}$ is conceptualized as a stack of t matrices, each of size $d \times d$. Each matrix adapts a pretrained weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$ as follows:

$$\boldsymbol{W}' = (\boldsymbol{I} + \alpha \mathcal{T}[j, :, :])\boldsymbol{W}, \tag{4}$$

where $\mathcal{T}[j, :, :]$ denotes the j-th frontal slice of the tensor $\mathcal{T}$, and α is a scaling factor analogous to that used in LoRA. If we regard different slices $\mathcal{T}[j, :, :]$ as different experts, then we can obtain a compact MoE framework based on the tensor formulation. Specifically, experts in a tensor are synthesized from a shared, compact set of parameters: the core tensor $\mathcal{G}$ and factor matrices $\boldsymbol{U}$ and $\boldsymbol{C}$.

This shared expert construction ensures that parameters for all experts are learned collectively from a compressed representation. Finally, the adaptation weight can be obtained by aggregating the elements in the tensor with proper expert allocation schemes.

3.2 Grouped Experts

To achieve a better performance-efficiency trade-off, we further formulate the experts with multiple tensors under a clear hierarchy. Specifically, we divide the experts into different groups. Each group of experts corresponds to a specific adaptation tensor $\mathcal{T}_i$. To better remain parameter-efficiency, we construct $\mathcal{T}_i$ as a series of k distinct tensor in the following:

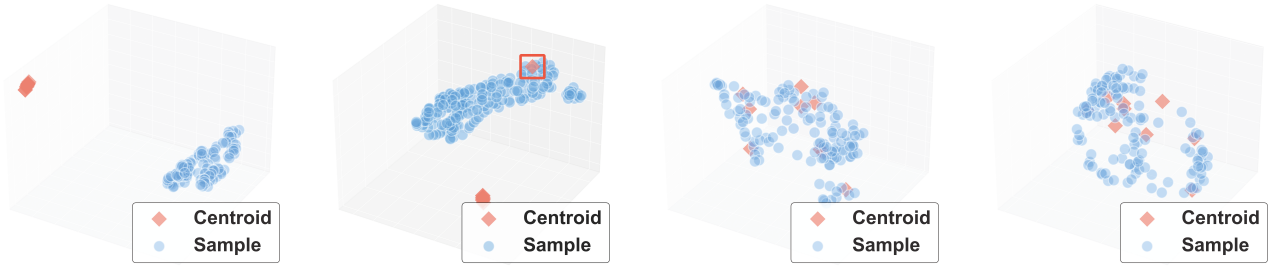
$$\mathcal{T}_i = \mathcal{G}_i \times_1 \boldsymbol{C} \times_2 \boldsymbol{U} \times_3 \boldsymbol{U}, \quad i = 1, 2, \ldots, k. \tag{5}$$

In this way, the tensors for different groups only differ from each other by the core tensor $\mathcal{G}_i$. Replacing the single core tensor $\mathcal{G}$ with a set of tensors $\{\mathcal{G}_i\}_{i=1}^{k}$ modulates the interaction between modes, affording greater independence between expert groups while still constraining them to the low-rank subspace defined by the shared factor matrices $\boldsymbol{U}$ and $\boldsymbol{C}$. With this hierarchical framework, the model can now capture **global variations** via allocating to different groups, while capturing **local** ones by choosing different expert weights within a group.

Expert Complexity. With the compact grouping strategy, if k expert groups are employed and there are M experts in total, the overall number of trainable parameters is in the order of:

$$\mathcal{O}\left(\frac{p}{k}M + kpr^2 + dr\right).$$

When $M, p, k, r << d^{1/2}$, this complexity is significantly **lower** than that of LoRA, which is $\mathcal{O}(2dr)$. The equation



(a) **Initial Step (Kaiming init.)** (b) **Last Step (Kaiming init.)** (c) **Initial Step (DAI)** (d) **Last Step (DAI)**

Figure 3: UMAP visualization of expert centroids and input embeddings. The figure compares the distribution of expert centroids (red diamonds) and CIFAR-100 input embeddings (blue circles) in ViT-base model. (a), (b) Kaiming initialization. (c), (d) Our proposed data aware initialization (DAI).

also suggests that TuckA can achieve a **more compact** structure by reducing r and increasing M (number of experts), therefore our selection of r is typically **smaller** than LoRA's. Through experiments in Sec. 4, we prove such an ensemble of **small** experts can defeat a **single large** expert empirically.

3.3 Routing

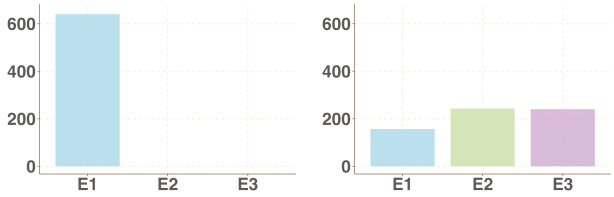
Having formulated our grouped experts in the previous section, we now examine the problem of expert routing.

Inspired by (Liu et al. 2024a), we quantify the similarity between the input and the available experts by assigning each expert a **trainable centroid vector**, $\boldsymbol{e}_{ij}$, for the j-th frontal slice of the i-th expert group. We then compute **affinity scores**, s_{ij}, between the input and expert centroids. These scores determine how the experts are utilized.

However, traditional MoE architecture routes individual samples or tokens to expert networks at each layer. For a model with M experts, this approach costs a trainable parameter number of $\sum_{l=1}^{L} Md^{(l)}$ and computational cost of $\sum_{l=1}^{L} BMd^{(l)}$, where $d^{(l)}$ is the size of input embedding at the l-th layer and B is the batch size. Considering the scale of $d^{(l)}$ and L, this method is **expensive** for PEFT. Furthermore, using different adaptation weights for each sample incurs the calculation of B distinct $d \times d$ matrices, which is a **computationally and memory-intensive** task.

To overcome this limitation, we propose a **batch-level routing** strategy. Our method makes a single routing decision for an entire batch of inputs, which is then propagated across all subsequent adapted layers, substantially reducing computational overhead while maintaining model performance. To achieve this, we calculate a single centroid vector named **routing feature**, $\bar{\boldsymbol{h}} = \frac{1}{B}\sum_{i=1}^{B} \boldsymbol{h}_i$, from the input embeddings of the entire batch. Subsequently, we **reduce** the number of trainable parameters in router to $Md^{(1)}$ and avoid calculating multiple adaptation weights.

To complement our hierarchical expert architecture, we implement a group-wise activation strategy. Specifically, we identify the expert with the highest affinity score across all other experts and activate its entire group to capture global variations. Within the activated group, experts are then combined based on their normalized affinity scores to capture local variations.



(a) **Kaiming Init.** (b) **Data-Aware Init.**

Figure 4: Comparison of expert activation on CIFAR-100. The bar charts show the total number of activations for each expert (E1, E2, E3) during the fine-tuning process. (a) With Kaiming initialization. (b) Our proposed data-aware initialization.

Based on the preceding description, we formulate TuckA as follows:

$$s_{ij} = \text{sigmoid}(\bar{\boldsymbol{h}}^{(1)} \cdot \boldsymbol{e}_{ij}), \tag{6}$$

$$(i_0, j_0) = \underset{i\in\{1,\dots,k\}, j\in\{1,\dots,t\}}{\arg\max} s_{ij}, \tag{7}$$

$$g_j = \frac{s_{i_0 j}}{\sum_{m=1}^{t} s_{i_0 m}}, \tag{8}$$

$$\boldsymbol{Y}^{(l)} = \boldsymbol{X}^{(l)} \left(\boldsymbol{I} + \alpha \sum_{j=1}^{t} g_j \mathcal{T}_{i_0}^{(l)}[j,:,:] \right) \boldsymbol{W}^{(l)}, \tag{9}$$

where for each adapted layer l, $\boldsymbol{X}^{(l)}$ and $\boldsymbol{Y}^{(l)}$ denote its batched input and output. These expert centroids are used in the first adapted layer of the model to determine the routing for every subsequent adapter. These equations, together with Eq. (5), form our proposed method, where the universal centroids $\boldsymbol{e}_{ij}$, and the layer-specific parameters $\mathcal{G}_i^{(l)}$, $\boldsymbol{C}^{(l)}$ and $\boldsymbol{U}^{(l)}$ are trainable parameters.

Method	# Par.	CoLA Mcc ↑	MRPC Acc/F1 ↑	QNLI Acc ↑	QQP Acc/F1 ↑	RTE Acc ↑	SST-2 Acc ↑	Avg.
LoRA$_{r=8}$	1.19M	69.40	92.19	92.48	83.58	<u>87.73</u>	94.38	86.63
DoRA$_{r=8}$	1.27M	70.76	92.20	91.40	84.59	**88.09**	92.55	86.60
VeRA$_{r=1024}$	<u>0.23M</u>	71.31	91.55	94.47	<u>89.78</u>	87.00	96.10	88.37
OFT$_{b=8}$	0.67M	71.56	92.45	93.59	89.68	84.84	95.64	87.96
BOFT$_{b=8}$	0.67M	71.65	91.85	93.98	**89.97**	81.59	95.87	87.49
HRA$_{r=8}$	0.60M	70.87	**92.87**	94.45	87.30	87.00	96.10	88.10
TuckA$_{r=2,k=4}$	**0.16M**	**73.24**	91.78	<u>94.49</u>	88.90	86.64	<u>96.33</u>	88.56
TuckA$_{r=6,k=2}$	0.46M	<u>72.13</u>	92.11	**94.55**	89.25	<u>87.73</u>	**96.44**	<u>88.70</u>
TuckA$_{r=8,k=2}$	0.64M	71.64	<u>92.82</u>	94.36	89.00	**88.09**	**96.44**	**88.73**

Table 1: Performance comparison of various PEFT methods applied to the DeBERTa-v3-base model on the GLUE benchmark. Details of these baselines can be found in **Appendix B**. We report the mean of accuracy and F1 for MRPC and QQP, Matthews correlation for CoLA, and accuracy for the rest. The best result for each column is in **bold**, and the second best is <u>underlined</u>.

3.4 Initialization and Load Balancing

In this paper, we find that the initialization of expert centroids plays a crucial role in achieving load balance in the MoE. Specifically, we start with the `Kaiming uniform` initialization. This strategy is a popular choice with its stability proven in many literature. However, we find this approach fails to create expert diversity as shown in Theorem 1.

Theorem 1 (**Imbalanced Load, Informal**). *Based on a proper setting, for simplicity, we assume that only one expert is employed within a group* ***without the score normalization*** *in Eq.8, and that* $\frac{\partial \mathcal{L}}{\partial s_i} < 0$ *for activated experts, and* ***`Kaiming Uniform`*** *initialization is employed for* $\boldsymbol{e}_i$*s. If the routing features are Gaussian with a large mean and a geometrically compact covariance matrix, then* ***the load is imbalanced*** *for different experts with high probability.*

proof sketch. If $i^\star$ is selected initially, we will prove that it will be picked again by most of the rest batches. By observing the gradient rule, we find that there is a positive-feedback phenomenon, in which the activation advantage of $i^\star$ will be continuously accumulated. Mathematically, denote the routing center for the 1st and 2nd batch as $\bar{\boldsymbol{h}}, \bar{\boldsymbol{h}}'$. Then, we prove that: the advantage gap $\langle \boldsymbol{e}_{i^\star} - \boldsymbol{e}_j, \bar{\boldsymbol{h}} \rangle + \langle \bar{\boldsymbol{h}}, \bar{\boldsymbol{h}}' \rangle$ is **sufficiently large to be positive** with high probability based on concentration inequalities in high-dimensional space. The results for the remaining batches follow a simple recursion and union bounds. □

The detailed setting and proof can be found in **Appendix A**. The assumption that $\frac{\partial \mathcal{L}}{\partial s_i} < 0$ is reasonable when all the expert embeddings are initially far apart from the sample distribution and the sample distribution only has one mode. In this sense, the activating expert must keep reducing the loss for most samples. Hence, keeping $\frac{\partial \mathcal{L}}{\partial s_i} < 0$ for one (or a few) experts can quickly lead to convergence. We also provide empirical validation in **Appendix C**.

To further support our theory, we visualize the input embedding and expert centroids with UMAP (Ghojogh et al. 2021). As demonstrated in Figure 3(a), initializing all expert centroids with a Kaiming distribution concentrates them within a small, dense cluster. This cluster is located very far from the routing feature distribution in the high-dimensional embedding space, and the covariance of the sample routing feature is relatively compact. This is consistent with the assumption in our theory. Moreover, we empirically observe expert collapse, where a single expert is over-trained while the others remain systematically neglected, as illustrated in Figure 3(b) and Figure 4(a). This again validates our theory.

While prior work has attempted to mitigate this issue through sophisticated load-balancing mechanisms, the efficacy of such methods can be limited, particularly under conditions of extreme initial imbalance. Notably, the DeepSeek-V3 model (Liu et al. 2024a), despite its auxiliary-loss-free strategy, still employs a complementary, sequence-wise auxiliary loss to prevent extreme imbalance. To circumvent these issues, we propose a **data-aware initialization (DAI)** strategy that directly addresses the imbalance from the perspective of initialization.

Specifically, we first compute the centroid, $\boldsymbol{e}$, of a small, randomly sampled batch of input token sequences. Proceeding from the assumption that tokens within a dataset are drawn from a locally consistent distribution, we initialize the set of M expert centroids by introducing small perturbations around this data centroid:

$$\boldsymbol{E}_{\text{init}} = [\boldsymbol{e}, \boldsymbol{e} + \boldsymbol{\epsilon}_1, \ldots, \boldsymbol{e} + \boldsymbol{\epsilon}_{M-1}]. \tag{10}$$

Here, each component of the perturbation vector $\boldsymbol{\epsilon}_i$ is sampled independently from a uniform distribution $\mathcal{U}(-a, a)$, where a is a scalar hyperparameter controlling the perturbation magnitude. This strategy positions the expert centroids within a region of high data density, rather than relying on the training process to relocate them from a suboptimal starting position. By selecting a proper hyperparameter a, our method can achieve effective load balancing from the outset, thereby obviating the need for auxiliary load-balancing loss terms, as demonstrated in Figure 3(c)(d) and Figure 4(b).

4 Experiments

We conduct a comprehensive empirical evaluation to assess the performance, parameter efficiency, and scal-

Method	# Par.	CIFAR100 Acc ↑	Food101 Acc ↑	Caltech256 Acc ↑	Avg.	Peak GPU Memory CIFAR100, MB ↓
LoRA$_{r=8}$	1.18M	82.71	72.94	88.43	81.36	**4419**
DoRA$_{r=8}$	1.25M	82.36	72.96	87.62	80.98	7995
VeRA$_{r=512}$	**0.10M**	75.89	69.89	87.40	77.73	5813
OFT$_{b=8}$	0.66M	82.73	72.52	87.21	80.82	5505
BOFT$_{b=8}$	0.66M	82.01	72.38	86.94	80.44	5541
HRA$_{r=8}$	0.59M	83.45	73.05	89.73	82.08	9530
TuckA$_{r=2,k=5}$	<u>0.16M</u>	84.65	73.33	<u>89.76</u>	82.58	4500
TuckA$_{r=6,k=1}$	0.45M	**84.81**	<u>73.43</u>	**89.90**	**82.71**	4514
TuckA$_{r=8,k=1}$	0.60M	<u>84.76</u>	**73.55**	89.54	<u>82.62</u>	4522

Table 2: Comparison of various PEFT methods on the ViT-Base model across three image classification benchmarks, including their peak GPU memory consumption on CIFAR100. All experiments were conducted in **few-shot** settings, utilizing only **10 training examples per class**. Details of these baselines can be found in **Appendix B**. The best result of each column is in **bold** and the second best is <u>underlined</u>.

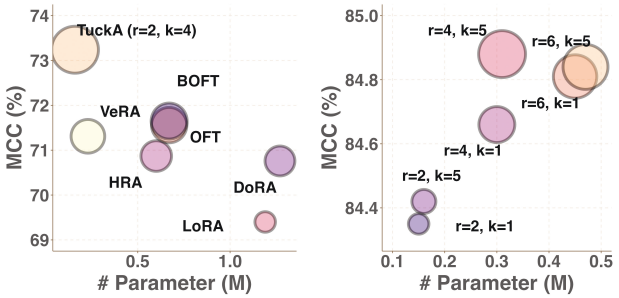
ability of our proposed method, TuckA. We benchmark TuckA against a suite of established PEFT techniques including **LoRA** (Hu et al. 2022), **DoRA** (Liu et al. 2024b), **VeRA** (Kopiczko, Blankevoort, and Asano 2024), **OFT** (Kopiczko, Blankevoort, and Asano 2024), **BOFT** (Liu et al. 2024c) and **HRA** (Yuan, Liu, and Xu 2024) across three diverse domains, demonstrating its broad applicability. For our experiments, we select representative Transformer architectures as base models: **DeBERTa-v3-base** (He, Gao, and Chen 2023) for natural language understanding, **Vision-Transformer-Base-Patch16** (Dosovitskiy et al. 2021; Li, Qi, and Geng 2025) for image classification, and **Llama2-7B** (Touvron et al. 2023) for mathematical reasoning. All baselines are sourced from the Hugging Face PEFT library (Mangrulkar et al. 2022) to ensure standardized and reproducible comparisons.

A key feature of our work is TuckA's architectural scalability. We investigate the effect of this by configuring TuckA with varying ranks and numbers of expert groups, creating variants that trade off between the complexity of individual experts and the number of expert groups. All experiments were performed on a system with eight NVIDIA RTX 4090 24G GPUs. Further implementation details, including hyperparameters, are provided in the **Appendix B**.

4.1 Natural Language Understanding

For the natural language understanding domain, we finetune **DeBERTa-v3-base** on the General Language Understanding Evaluation (GLUE) benchmark (Wang et al. 2019). Our methodology follows the established protocol from the original LoRA paper (Hu et al. 2022), encompassing six tasks that evaluate single-sentence classification, sentence-pair similarity, and natural language inference. We apply PEFT methods to the query, value, and dense layers, while the classification head remains fully trainable.

As presented in Table 1, TuckA demonstrates a clear and consistent performance advantage, outperforming all baseline methods on the majority of GLUE tasks. These results highlight TuckA's ability to capture rich linguistic features more effectively than existing approaches. Notably, our most



(a) **CoLA** (b) **CIFAR100**

Figure 5: Tradeoff between Performance and Parameter Efficiency. The ideal position is the top-left corner, representing high performance with high parameter efficiency. (a) A comparison of TuckA with other PEFT methods on the CoLA benchmark. (b) Impact of expert numbers and rank on CIFAR-100 dataset under **few-shot** setting.

parameter-frugal variant ($r = 2$, 4 expert groups), with merely 0.16 million trainable parameters, not only achieves competitive results but also surpasses several baselines. This underscores TuckA's exceptional efficiency. When allocated a slightly larger parameter budget, the TuckA's 0.64M variant also achieves the best average performance among all evaluated PEFTs, confirming the method's potent scalability and effectiveness.

4.2 Image Classification

Following the image classification evaluation protocol of (Kopiczko, Blankevoort, and Asano 2024; Wang et al. 2025; Ma et al. 2025), we assess TuckA on three standard datasets: CIFAR-100 (Krizhevsky, Hinton et al. 2009), Food-101 (Bossard, Guillaumin, and Van Gool 2014), and Caltech-256 (Griffin et al. 2007). We use a Vision Transformer pretrained on ImageNet-21K (Deng et al. 2009) as our base model. For each dataset, the training set is constructed with a challenging 10-shot setup (10 images per

class), and evaluation is performed on the full validation set. PEFT methods are applied to the query, value, and dense layers, with a trainable classification head.

TuckA's superiority persists in the vision domain, as detailed in Table 2. Across all three datasets, TuckA variants consistently outperform the baseline methods. Specifically, the $r = 6$ variant with a single expert group achieves the highest overall performance, securing top results with a remarkably low parameter size of just 0.45 million.

A critical aspect of PEFT is not just reducing parameter size but also managing computational overhead. We investigate the peak GPU memory consumption during training on CIFAR-100. As shown in Table 2, a reduction in trainable parameters does not guarantee lower memory usage; some methods with fewer parameters than LoRA still consume more memory due to the need to store complex intermediate activations and gradients. In contrast, TuckA exhibits memory consumption comparable to LoRA. This demonstrates that the additional computations for Tucker decomposition and expert routing in TuckA are efficiently implemented, imposing no substantive memory penalty and overcoming a key limitation of other advanced PEFT architectures.

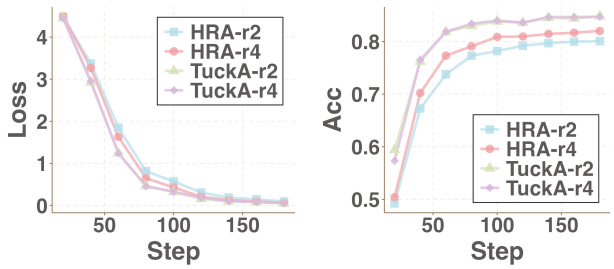
Method	# Par.	MATH Acc↑	GSM-8K Acc ↑
LoRA$_{r=32}$	16.8M	6.62	<u>50.34</u>
DoRA$_{r=32}$	17.0M	6.70	48.67
TuckA$_{r=16,k=4}$	**4.4M**	<u>7.26</u>	<u>50.34</u>
TuckA$_{r=32,k=4}$	<u>10.7M</u>	**9.02**	**53.75**

Table 3: **Low Bit (BF16)** performance of Llama2-7B fine-tuned on the **first 100,000** examples of the MetamathQA dataset for **2 epochs**. The models are evaluated on the MATH and GSM-8K benchmarks. The best result of each column is in **bold** and the second best is <u>underlined</u>.

4.3 Mathematical Reasoning

To challenge TuckA on a complex, large-scale reasoning task (Wu et al. 2025a,b), we fine-tune the Llama2-7B model on the MetaMathQA dataset (Yu et al. 2024). Following the setup from (Meng, Wang, and Zhang 2024), we apply adapters to the query and value layers and train for two epochs on the first 100,000 examples.

The sheer scale of Llama2-7B presented significant hardware challenges, making it impossible to run several memory-intensive baselines on our 24G GPUs, even with memory optimizations such as DeepSpeed (Rasley et al. 2020) and BFloat16 precision. The successful execution of TuckA highlights its memory efficiency. As shown in Table 3, TuckA achieves higher scores than both LoRA and DoRA under our experimental constraints. By employing the same rank as LoRA but distributing the capacity across four expert groups, TuckA achieves strong mathematical reasoning performance while utilizing significantly less trainable parameters. This result illustrates TuckA's potential for scaling effectively to large models, particularly in resource-constrained environments where other methods may be infeasible.



(a) **Train Loss** (b) **Accuracy**

Figure 6: CIFAR100 few-shot performance of TuckA and HRA. TuckA is using the setting $M = t = p = 1$.

4.4 Finer-grained Analysis

We provide a series of more detailed analyses to explore the properties of TuckA further.

First, we analyze the trade-off between increasing the rank and the number of expert groups. As illustrated in Figure 5(b), increasing the number of experts consistently improves performance with only a marginal increase in the number of parameters. This suggests that distributing representational capacity across more, simpler expert groups is a highly parameter-efficient scaling strategy. Conversely, while increasing the rank also boosts performance, it does so at a greater cost to the parameter budget and, as observed, a very large rank can increase the risk of overfitting to the few-shot training data.

Second, we compare TuckA to HRA, which is a constrained special case of TuckA when the expert number is set to one. By constraining TuckA to a single expert group and specific factor dimensions ($t = 1, p = 1$ in Eq. 3), it becomes an unrestricted variant of HRA. Figure 6 shows that while both methods converge similarly on the training set, TuckA achieves higher evaluation accuracy. This result supports our hypothesis that the unconstrained factor matrices in TuckA can provide a richer representational capacity, potentially leading to enhanced generalization.

Moreover, we also conduct ablation studies to demonstrate the impact of the initialization method. Due to the limited space, please refer to **Appendix C** for more details.

5 Conclusion

In this paper, we introduce TuckA, a PEFT method that integrates multiple adaptation experts into a compact structure. Specifically, we ensemble a collection of small adapters using Tucker decomposition and a hierarchical MoE structure to handle the representational limitation and parameter redundancy of large, single-expert adapters. We also propose carefully designed data-aware initialization and universal routing mechanisms to alleviate expert collapse common in MoE architectures. Experiments on diverse benchmarks demonstrate the effectiveness of TuckA in enhancing adaptation capacity in PEFT.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China: 62525212, 62236008, 62441232, U21B2038, U23B2051, 92370102, and 62176260, in part by Youth Innovation Promotion Association CAS, in part by the Strategic Priority Research Program of the Chinese Academy of Sciences, Grant No. XDB0680201, in part by the Fundamental Research Funds for the Central Universities, and in part by the China National Postdoctoral Program for Innovative Talents, Grant No. BX20250377.

References

Bai, J.; Bai, S.; Chu, Y.; Cui, Z.; Dang, K.; Deng, X.; Fan, Y.; Ge, W.; Han, Y.; Huang, F.; Hui, B.; Ji, L.; Li, M.; Lin, J.; Lin, R.; Liu, D.; Liu, G.; Lu, C.; Lu, K.; Ma, J.; Men, R.; Ren, X.; Ren, X.; Tan, C.; Tan, S.; Tu, J.; Wang, P.; Wang, S.; Wang, W.; Wu, S.; Xu, B.; Xu, J.; Yang, A.; Yang, H.; Yang, J.; Yang, S.; Yao, Y.; Yu, B.; Yuan, H.; Yuan, Z.; Zhang, J.; Zhang, X.; Zhang, Y.; Zhang, Z.; Zhou, C.; Zhou, J.; Zhou, X.; and Zhu, T. 2023. Qwen Technical Report. Technical report, Qwen Team, Alibaba Group.

Bossard, L.; Guillaumin, M.; and Van Gool, L. 2014. Food-101 – Mining Discriminative Components with Random Forests. In *ECCV 2014*.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR 2009*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR 2021*.

Ghojogh, B.; Ghodsi, A.; Karray, F.; and Crowley, M. 2021. Uniform Manifold Approximation and Projection (UMAP) and its Variants: Tutorial and Survey. arXiv:2109.02508.

Griffin, G.; Holub, A.; Perona, P.; et al. 2007. Caltech-256 object category dataset. Technical report, California Institute of Technology Pasadena.

Guo, S.; Hong, I.; Balmaseda, V.; Yu, C.; Qiu, L.; Liu, X.; Jiang, H.; Zhao, T.; and Yang, T. 2025. Discriminative Fine-tuning of Generative Large Language Models without Reward Models and Human Preference Data. In *Proceedings of the 42nd International Conference on Machine Learning*, 20825–20842.

Han, B.; Xu, Q.; Bao, S.; Yang, Z.; Zi, K.; and Huang, Q. 2025. LightFair: Towards an Efficient Alternative for Fair T2I Diffusion via Debiasing Pre-trained Text Encoders.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *ICCV 2015*.

He, P.; Gao, J.; and Chen, W. 2023. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. In *ICLR 2023*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR 2022*.

Hua, C.; Xu, Q.; Bao, S.; Yang, Z.; and Huang, Q. 2024. ReconBoost: Boosting Can Achieve Modality Reconcilement. In *International Conference on Machine Learning*, 19573–19597.

Hua, C.; Xu, Q.; Yang, Z.; Wang, Z.; Bao, S.; and Huang, Q. 2025. OpenworldAUC: Towards Unified Evaluation and Optimization for Open-world Prompt Tuning. *CoRR*, abs/2505.05180.

Kopiczko, D. J.; Blankevoort, T.; and Asano, Y. M. 2024. VeRA: Vector-based Random Matrix Adaptation. In *ICLR 2024*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Li, L.; Qi, L.; and Geng, X. 2025. One-Shot Knowledge Transfer for Scalable Person Re-Identification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 668–677.

Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; Lu, C.; Zhao, C.; Deng, C.; Zhang, C.; Ruan, C.; Dai, D.; Guo, D.; Yang, D.; Chen, D.; Ji, D.; Li, E.; Lin, F.; Dai, F.; Luo, F.; Hao, G.; Chen, G.; Li, G.; Zhang, H.; Bao, H.; Xu, H.; Wang, H.; Zhang, H.; Ding, H.; Xin, H.; Gao, H.; Li, H.; Qu, H.; Cai, J. L.; Liang, J.; Guo, J.; Ni, J.; Li, J.; Wang, J.; Chen, J.; Chen, J.; Yuan, J.; Qiu, J.; Li, J.; Song, J.; Dong, K.; Hu, K.; Gao, K.; Guan, K.; Huang, K.; Yu, K.; Wang, L.; Zhang, L.; Xu, L.; Xia, L.; Zhao, L.; Wang, L.; Zhang, L.; Li, M.; Wang, M.; Zhang, M.; Zhang, M.; Tang, M.; Li, M.; Tian, N.; Huang, P.; Wang, P.; Zhang, P.; Wang, Q.; Zhu, Q.; Chen, Q.; Du, Q.; Chen, R. J.; Jin, R. L.; Ge, R.; Zhang, R.; Pan, R.; Wang, R.; Xu, R.; Zhang, R.; Chen, R.; Li, S. S.; Lu, S.; Zhou, S.; Chen, S.; Wu, S.; Ye, S.; Ye, S.; Ma, S.; Wang, S.; Zhou, S.; Yu, S.; Zhou, S.; Pan, S.; Wang, T.; Yun, T.; Pei, T.; Sun, T.; Xiao, W. L.; and Zeng, W. 2024a. DeepSeek-V3 Technical Report. Technical report, Deepseek-AI.

Liu, S.; Wang, C.; Yin, H.; Molchanov, P.; Wang, Y. F.; Cheng, K.; and Chen, M. 2024b. DoRA: Weight-Decomposed Low-Rank Adaptation. In *ICML 2024*.

Liu, W.; Qiu, Z.; Feng, Y.; Xiu, Y.; Xue, Y.; Yu, L.; Feng, H.; Liu, Z.; Heo, J.; Peng, S.; Wen, Y.; Black, M. J.; Weller, A.; and Schölkopf, B. 2024c. Parameter-Efficient Orthogonal Finetuning via Butterfly Factorization. In *ICLR 2024*.

Ma, T.; Chen, H.; Hu, J.; Zhu, Y.; and Li, X. 2025. Forming Auxiliary High-confident Instance-level Loss to Promote Learning from Label Proportions. arXiv:2411.10364.

Mangrulkar, S.; Gugger, S.; Debut, L.; Belkada, Y.; Paul, S.; and Bossan, B. 2022. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft.

Meng, F.; Wang, Z.; and Zhang, M. 2024. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. In *NeurIPS 2024*.

Mu, S.; and Lin, S. 2025. A Comprehensive Survey of Mixture-of-Experts: Algorithms, Theory, and Applications. arXiv:2503.07137.

Qiu, Z.; Liu, W.; Feng, H.; Xue, Y.; Feng, Y.; Liu, Z.; Zhang, D.; Weller, A.; and Schölkopf, B. 2023. Controlling Text-to-Image Diffusion by Orthogonal Finetuning. In *NeurIPS 2023*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. Technical report, OpenAI.

Rasley, J.; Rajbhandari, S.; Ruwase, O.; and He, Y. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *ACM SIGKDD 2020*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Canton-Ferrer, C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardas, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. Technical report, GenAI, Meta.

Tucker, L. R. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3): 279–311.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *ICLR 2019*.

Wang, W.; Ma, T.; Xie, M.-K.; Niu, G.; and Sugiyama, M. 2025. Rethinking Consistent Multi-Label Classification under Inexact Supervision. arXiv:2510.04091.

Wu, J.; Feng, M.; Zhang, S.; Jin, R.; Che, F.; Wen, Z.; and Tao, J. 2025a. Boosting multimodal reasoning with mcts-automated structured thinking. *arXiv e-prints*, arXiv–2502.

Wu, J.; Liao, C.; Feng, M.; Zhang, S.; Wen, Z.; Shao, P.; Xu, H.; and Tao, J. 2025b. Thought-augmented policy optimization: Bridging external guidance and internal capabilities. *arXiv preprint arXiv:2505.15692*.

Yu, L.; Jiang, W.; Shi, H.; Yu, J.; Liu, Z.; Zhang, Y.; Kwok, J. T.; Li, Z.; Weller, A.; and Liu, W. 2024. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. In *ICLR 2024*.

Yuan, S.; Liu, H.; and Xu, H. 2024. Bridging The Gap between Low-rank and Orthogonal Adaptation via Householder Reflection Adaptation. In *NeurIPS 2024*.

Zhang, Q.; Chen, M.; Bukharin, A.; He, P.; Cheng, Y.; Chen, W.; and Zhao, T. 2023. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *ICLR 2023*.