DARS : ROBUST SPARSE FINE-TUNING WITH REGULARIZED SUBSPACE DISALIGNMENT

Sumin Park[1], **Noseong Park**[1]
[1]Department of School of Computing, KAIST
{psmiz,noseong}@kaist.ac.kr

ABSTRACT

Recent works have identified the alignment, which measures a layerwise weight correlation, as a novel yet crucial mechanism for feature learning. We investigate an underlying connection between the alignment learning and the structural fitting of a network to the training data span. Based on this insight, we further demonstrate that fine-tuning on out-of-distribution (OOD) data disrupts this well-aligned structure fitted during the pre-training phase, degrading generalization performance. To address this, we propose **DARS**, **D**is**A**lignment-**R**egularized **S**parse fine-tuning, a novel sparse fine-tuning approach that mitigates disalignment by letting the gradient update to be partially constrained within the principal subspace of the pre-trained network, constructed based on the in-distribution (ID) data used for its pre-training. Specifically, we define the two disjoint subsets of trainable parameters for sparse channel unfreezing: i) a random subset and ii) a subset with higher gradient projections onto the principal subspace. The latter serves as a disalignment regularizer during fine-tuning, while the random subset ensures a minimal bias in parameter selection. By adjusting the ratio between the two subsets, we can control the strength of subspace regularization, thereby balancing the trade-off between generalization capacity and strong fitting to new downstream tasks. By employing DARS, we achieved SOTA performance on various benchmarks, including commonsense and arithmetic reasoning tasks, across LLaMA-7B and LLaMA2-7B.

1 INTRODUCTION

Since Large Language Models (LLMs) achieved great success with broad adaptability to diverse domains, fine-tuning methods to better adapt the pre-trained LLMs to target downstream tasks have been extensively explored, leading to advancements in domain-specific applications. The most naive approach that re-train the entire networks, called as full fine-tuning (FT) (Roziere et al., 2023; Azerbayev et al., 2023) not only takes enormous computational resources but also suffers from catastrophic forgetting, where the model loses its pre-trained knowledge (Luo et al., 2023; Biderman et al., 2024). To resolve these bottlenecks of full FT, Parameter-Efficient Fine-Tuning (PEFT) techniques have been proposed to minimize the number of trainable parameters during fine-tuning. By only updating a smaller subset of parameters, PEFT aims to retain critical pre-trained knowledge, while efficiently acquiring new information. Common PEFT approaches include LoRA-based techniques, such as AdaLoRA (Zhang et al., 2023), DoRA (Liu et al., 2024), and Galore (Zhao et al., 2024) that constrains the model updates into low-rank subspaces of original weight spaces.

Sparse Fine-Tuning However, these LoRA-variants are in general highly sensitive to the choice of initialization (Hayou et al., 2024; Huang & Balestriero, 2024) and shows much worse performance than full FT in some tasks (Ding et al., 2022). Recent researches actively investigates non-LoRA based approaches, searching for more robust and efficient fine-tuning strategies. Especially, sparse fine-tuning that selects only a subset of pre-trained parameters that are critical to downstream tasks while remaining the rest intact has achieved better performance and efficiency across various tasks (Xu et al., 2021; Ansell et al., 2023; Yang et al., 2024; Pan et al., 2024). Among them, S2FT (Yang et al., 2024) that randomly unfreezes small fraction of row and column vectors of pre-trained weight matrices shows significant performance jump across diverse domains with improved generalization capabilities.

Alignment There has been recent researches that investigate the role of alignment as a mechanism of feature learning. Alignment can be identified in various forms, such as subspace alignment (Fernando et al., 2014; Thopalli et al., 2022), layer-wise alignment between parameter space (Xu & Ziyin, 2024), or alignment between learned representations and the weight matrices (Everett et al., 2024). Studying the evolution of weight alignment between two consecutive layers during training, prior work has demonstrated that with small enough initialization, the model preferably learns features by alignment regardless of the choice of non-linearities (Xu & Ziyin, 2024). This preference on alignment over disalignment for feature learning can be understood as the alignment enables efficient layerwise information transfer.

Alignment learning during training is closely related to the magnitude of model norm, which plays a crucial role in determining the model's generalization capacity. While aligned model requires a smaller model norm to make a prediction, a model with disaligned layers requires an overly large model norm to make the same prediction. However, increasing the model alignment is not always desirable. While alignment ensures efficient layer utilization and prevents overfitting, overly high alignment can mask the feature saliency with limited norm size, leading to underfitting. Therefore, keeping an optimal balance for the trade-off between alignment and disalignment will be a critical point to consider when developing a new feature learning algorithm.

Although previous works have figured out the favorable conditions for alignment learning and how it affects the generalization ability of the model, the detailed mechanism of how this alignment occurs with respect to the training data has not yet been investigated. In this paper, we identify that the feature learning by alignment is actually deeply connected to fitting the network to the space spanned by training data. We also explain fine-tuning to new data with large distribution shift from pre-trained data, namely out-of-distribution (OOD) data, significantly alters this learned alignment, worsening the generalization capacity of the model.

Contribution and Outlines As the alignment has been recently discussed as a mechanism of feature learning, existing sparse fine-tuning methods do not consider the effect of altered structural alignment during fine-tuning. They have employed channel-wise (Yang et al., 2024) or layer-wise (Pan et al., 2024) sparsity with a primary focus on maximizing the parameter efficiency by reducing the number of parameters while maintaining the comparable performance. Therefore, we propose a novel mechanism to regularize the disalignment during fine-tuning by constraining the subset of selected sparse channels to the parameters whose gradient updates dominantly align to the principal subspace to which the pre-trained network is originally fitted. Our contributions and outline of this paper are summarized as follows:

- In Section 3, we discuss how alignment occurs at the basic level of network operations in accordance with the training data. Then we discuss how fine-tuning to OOD data perturbs this alignment, impairing the generalization capacity.
- As a remedy to this problem, we propose a novel **D**is**A**lignment-**R**egularized **S**parse fine-tuning algorithm, **DARS** in Section 4.
- In Section 5, we demonstrate that our method improves the model performance across various domains and models.

2 PRELIMINARIES

Model and Notation In this section, all the the necessary derivations rely on the very basic network dynamics universally applied to neural networks of any depth and complexity such as stochastic gradient descent (SGD) and matrix multiplication. Hence, we consider a simple two-layer network with power activations without loss of generality. Given the data distribution $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ where $x_i \in \mathbb{R}^{d_0}$ and $y_i \in \mathbb{R}^m$, the output of networks is given as

$$f(x) = U(Wx^T)^\beta, \tag{1}$$

where $U \in \mathbb{R}^{m \times d}$ and $W \in \mathbb{R}^{d \times d_0}$ are weight matrices with intermediate width d for the second and the first layer, respectively. We can denote the activations drawn from each of the two layers as

$$z_w = (Wx^T)^\beta, z_u = Uz_w. \tag{2}$$

Then the goal of training is to find f^* that satisfies $f^*(x) = \text{argmin}_{f \in \mathcal{F}} \mathcal{L}_\mathcal{D}(f, (x, y))$, where $\mathcal{F} : X \to \mathbb{R}^m$ is the hypothetical space for network function f and $\mathcal{L}$ is the loss function.

Layerwise Alignment and Generalization Following the definition of alignment from Xu & Ziyin (2024) and Everett et al. (2024), let us define the alignment between two consecutive layers as,

$$\zeta := \frac{\|UW\|}{\|U\|\|W\|}, \tag{3}$$

where the norm used is the L2 norm. As an analog of cosine similarity defined between two matrices instead of between vectors, this metric serves as a measure of geometric alignment between two layers, indicating how well the two parameter spaces align. If ζ is large, the subspaces spanned by U and W are well-aligned, meaning that transformations at one layer (represented by W) are effectively being used by the subsequent layer (represented by U). Low ζ, on the other hand, suggests inefficiency in the interaction between layers, potentially leading to redundancy or under-utilization of layer capacities. Using the alignment term ζ and with a trivial assum, the model can be re-written as,

$$f(x) = \|\mathbf{U}\|\|\mathbf{W}\|\,\zeta\, x^T. \tag{4}$$

The magnitude of model output is determined by two factors: (1) scale of $\|U\|\|W\|$ and (2) the alignment ζ. Assuming that there's an optimal scale of the model predictions, disaligned layers need to compensate for the loss of alignment by increasing the model norms to maintain the same outputs, which could in turn lead to strong overfitting.

$$\|\mathbf{U}\|\|\mathbf{W}\| \propto \frac{\|f^*(x)\|}{\zeta\|x\|} \tag{5}$$

Therefore, disalignment sacrifices the generalization power of the networks to increase the fitting in scale, converging to a disproportionally larger norm than what's actually required to fit the data. Experimental results from Xu & Ziyin (2024) shows that models trained by disalignment tend to have larger norms and worse generalization performance, despite achieving 100% training accuracy.

3 Alignment and Disalignment Learning

Alignment Fitting to Training Data Span During pre-training, the entire network is fitted to pre-training data via basic network dynamics. Firstly, since the backpropagated gradient update with SGD is confined within the input span, weight matrix W and U are the span of $x \in \mathbb{R}^{d_0}$ and $z_w \in \mathbb{R}^d$, respectively (See the proof in Appendix A). Similar to this alignment between parameter and its input, it is also possible to find alignments between (1) parameters and post-activations and (2) layerwise parameters. Let the power activation $\beta = 1$, the forward pass in the first-layer of the model can be given by,

$$z_w = Wx^T = \sum_j^{d_0} w_j x^j, \tag{6}$$

where $w_j \in \mathbb{R}^d$ is the column vector of weight matrix W. This indicates that the post-activations from the first layer, z_w, lie in the space spanned by the linear combinations of the column vectors of weight matrix W. Therefore, the weight matrix U confined within the span of z_w also aligns to the space spanned by W. Extending these alignments across multiple layers through iterated forward and backward passes introduces a strong correlation between the distribution of the pre-training data and the network's learned representations.

Disalignment Learning During Fine-Tuning This structured alignment across depths of the pre-trained network underlies the key bottleneck of fine-tuning, which is to efficiently handle the distribution shift from training data, known as OOD generalization (Kumar et al., 2022; Choi et al., 2024). Fine-tuning on data from OOD distributions not only results in underperformance but also causes significant distortion of the pre-trained knowledge. Given that pre-trained network has learned features by increasing the layer-wise alignment driven from ID data, fine-tuning to new data with large distribution shift from ID adds misalignment to the network. Especially when the fine-tuning data is embedded in an orthogonal space to the pre-trained data, gradient updates, which lie in the span of input data, significantly decreases the pre-trained alignment, further hindering the efficient transfer of learned representations. Detailed description about how fine-tuning on data from far OOD distributions results in reduced alignment is given in Appendix B. We also empirically show that

fine-tuning constantly decreases the layerwise alignment while compensating this with increased weight norm in Fig. 1 in Appendix B. This trade-off between efficient use of layer with small norm in well-aligned model and stronger fitting with larger norm due to disalignment well correlates to a common dilemma in neural networks, overfitting and generalization. Therefore, regularizing the layerwise alignment to find the optimal balance for this trade-off has great potential to resolve the issue.

4 Proposed Method

Here, we propose our new sparse fine-tuning approach, DARS, which adaptively regularizes the disalignment during fine-tuning. DARS achieves this by deploying two distinct subsets of trainable parameters. First subset is a random subset with no bias in parameter selection, which takes a dominant role in fitting the pre-trained network to new downstream tasks. The second one is a subset with higher gradient projections onto the principal subspace driven from ID data. While still serving task-specific parameter fitting, this subset primarily functions as a disalignment regularizer during the fine-tuning.

Identifying Parameters for Disalignment Regularization We use norm of sample gradients projected onto the principal subspaces $S := [S^1, S^2, ..., S^L]$ of the L-layered pre-trained network f_θ parameterized by $\theta = [\theta^1, \theta^2, ..., \theta^L]$. Principal subspace at l-th layer S^l is constructed as the lower-rank approximation of the SVD of final hidden representation from l-th layer of the pre-trained network given the ID data. See Appendix C for algorithmic details for computing the principal subspace. After computing S, we then get the gradient projection norms onto these subspaces. Given each data pair (x_i, y_i) from fine-tuning dataset $\mathbb{D} = \{(x_i, y_i)\}_{i=1}^N$, the gradient at layer l is as,

$$g_i^l = \nabla_{\theta^l} \mathcal{L}_\mathbb{D}(f_\theta(x_i), y_i),$$

where the $\mathcal{L}_\mathbb{D}$ is the cross-entropy loss by comparing the model's predicted softmax probability to a uniform vector used as the target, following Huang et al. (2021). Let the matrix of computed gradients concatenated acorss all N samples as $G^l = \{g_1^l, g_2^l, ..., g_N^l\}$, then the projection onto the l-th subspace S^l is given by

$$P_{S^l}(G^l) = \frac{1}{N} \sum_i^N \nabla_{\theta^l} \mathcal{L}_\mathbb{D}(f_\theta(x_i), y_i)\, S^l S^{l^T}. \tag{7}$$

Since the principal subspace is basically derived from the space spanned by the final activations from each layer, parameterized by θ^l, it lies within the span of parameter space given Equation (6). Accordingly, the parameters whose gradient update has higher projection onto these spaces indicate that the weight updates along these parameters dominantly lie in the space well-aligned to the consecutive layers of the pre-trained networks, preserving the original aligned structure. Therefore, selectively unfreezing these parameters for fine-tuning can regulate the disalignment, restricting the updates to the principal subspace.

Fine-Tuning Target Modules There are two modules considered for fine-tuning in our approach, Multi-Head Attention (MHA) and Multi-Layer Perceptron (MLP). For MHA module, there are four types of linear layers, Query, Key, Value, and Output; $Q, K, V, O \in \mathbb{R}^{d\times d}$. For MHA module with h attention heads, the corresponding linear projection for i-th head can be denoted as $Q_i, K_i, V_i \in \mathbb{R}^{d\times d_h}$ and $O_i \in \mathbb{R}^{d_h \times d}$, where $d_h = d/h$ is the head dimension. MLP module has three distinct linear projections, Up ($U \in \mathbb{R}^{d'\times d}$), Gate ($G \in \mathbb{R}^{d'\times d}$), and Down ($D \in \mathbb{R}^{d\times d'}$).

Parameter Selection Let $\mathcal{S}_{\text{MHA}}$ be the total selected attention heads for MHA module and we denote $|\cdot|$ as the cardinality of a set. Then, let $\mathcal{R}_{\text{MHA}}$ denotes the randomly selected heads with size $\alpha|\mathcal{S}_{\text{MHA}}|$ and $\mathcal{H}_{\text{MHA}}$ denotes the heads selected as top-$\{(1-\alpha)|\mathcal{S}_{\text{MHA}}|\}$ gradient projection norm summed over d_h, where $\mathcal{R}_{\text{MHA}}, \mathcal{H}_{\text{MHA}} \subseteq [h]$ and $\mathcal{S}_{\text{MHA}} = \mathcal{R}_{\text{MHA}} \cup \mathcal{H}_{\text{MHA}}$. Similarly, $\mathcal{S}_{\text{MLP}}$ indicates the set of total selected channels for MLP module. Then $\mathcal{R}_{\text{MLP}}$ is a set of randomly selected channels with set size $\beta|\mathcal{S}_{\text{MLP}}|$ and $\mathcal{H}_{\text{MLP}}$ is the set that consists of top-$\{(1-\beta)|\mathcal{S}_{\text{MLP}}|\}$ channels based on gradient projection norm, where $\mathcal{S}_{\text{MLP}} = \mathcal{R}_{\text{MLP}} \cup \mathcal{H}_{\text{MLP}}$. Here both $\alpha, \beta \in [0, 1]$ are the hyperparameters that control the balance between the two subsets for each module. By adjusting the magnitude of α, β, we can adaptively control the disalignment driven by random parameter selection.

5 EXPERIMENTS

Here we present the performance improvements achieved by DARS across commonsense and arithmetic reasoning tasks on two pre-trained models, LLaMA-7B and LLaMA2-7B.

Commonsense Reasoning The commonsense reasoning dataset comprises 8 sub-tasks: BoolQ , PIQA, SocialQA, HellaSwag, WinoGrande, ARC-challenge, ARC-easy, and OpenbookQA. We firstly extracted 1% subset of the training data for each sub-task and utilized it to fit the parameter subset $\mathcal{H}$ for that specific task. Then we combined the rest training data from all eight tasks into a single fine-tuninig dataset to train the subset $\mathcal{R}$ and evaluated performance on the individual test dataset for each task. Further details, including the hyperparameters, in Appendix D.

Table 1: Comparison among various fine-tuning methods for the LLaMA-7B, LLaMA2-7B models on eight commonsense reasoning tasks. ([1] from Liu et al. (2024), [2] from Yang et al. (2024))

Model	Method	# Param(%)	BoolQ	PIQA	SIQA	HellaSwag	Wino	ARC-e	ARC-c	OBQA	Avg. ↑
LLaMA-7B	LoRA[1]	0.83	69.2	81.7	78.4	83.4	80.8	79.0	62.4	78.4	76.7
	DoRA[1]	0.84	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	78.1
	S^2FT[2]	0.81	**72.7**	83.7	79.6	93.4	83.5	**86.1**	72.2	83.4	81.8
	DARS	0.81	71.2	**84.3**	**80.1**	**93.6**	**83.7**	85.9	**73.6**	**83.6**	**82.0**
LLaMA-2-7B	LoRA[1]	0.83	69.8	79.9	79.5	83.6	82.6	79.8	64.7	81.0	77.6
	DoRA[1]	0.84	71.8	83.7	76.0	89.1	82.6	83.7	68.2	82.4	79.7
	S^2FT[2]	0.81	72.9	**86.1**	80.2	**94.3**	85.5	87.2	74.6	83.4	83.0
	DARS	0.82	**73.3**	85.0	**80.6**	93.9	**86.0**	**89.1**	**75.4**	**85.8**	**83.6**

As shown in Table 1, DARS consistently achieves superior accuracy over other PEFT methods, including LoRA, DoRA, and S^2FT. In average across LLaMA-7B and LLaMA2-7B, the performance gain is 5.6% over LoRA, 3.9% over Dora, and 0.2% over S^2FT.

Arithmetic Reasoning We also evaluated DARS on seven math reasoning tasks, including MultiArith, GSM8K, AddSub, AQuA, SingleEq, SVAMP and MAWPS. We used Math-10K dataset for fine-tuning, which combines training sets from GSM8K, MAWPS, and AQuA, augmented with LM-generated chain-of-thought steps. Unlike commonsense reasoning tasks, we didn't divide the Math-10K into multiple sub-tasks for fitting $\mathcal{H}$ separately for each of them. For training both $\mathcal{H}$ and $\mathcal{R}$, we simply follow the experimental setup same as done in Hu et al. (2023).

Table 2: Comparison among various fine-tuning methods for different models on seven math reasoning tasks. ([1] from Yang et al. (2024))

Model	Method	# Param (%)	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	MAWPS	Avg. ↑
LLaMA-7B	LoRA[1]	0.83	98.0	40.0	91.2	21.7	93.1	56.7	85.3	69.7
	DoRA[1]	0.84	97.3	38.9	89.6	22.4	93.9	**58.4**	85.3	69.4
	S^2FT[1]	0.81	**98.8**	**41.3**	**91.4**	21.3	93.5	**58.4**	86.1	70.1
	DARS	0.81	**98.8**	37.8	90.9	**24.8**	**95.5**	56.3	**89.5**	**70.5**
LLaMA-2-7B	LoRA[1]	0.81	97.5	44.0	91.2	20.9	94.1	59.2	85.7	70.4
	DoRA[1]	0.84	98.2	44.8	90.1	24.4	94.5	59.1	89.1	71.3
	S^2FT[1]	0.81	98.5	44.3	91.1	**25.2**	94.7	**61.8**	88.2	72.0
	DARS	0.81	**99.0**	**45.9**	**91.6**	22.8	**95.5**	61.5	**89.3**	**72.2**

Table 2 demonstrates that DARS also exceeds other existing methods, 1.3% over LoRA, 1% over DoRA, and 0.3% over S^2FT.

6 CONCLUSION

This paper introduces a novel sparse fine-tuning approach that regularizes the disalignment to ensure OOD-robust and efficient performance on various tasks. We believe that our approach provides a promising start for considering structural alignment across networks into fine-tuning strategies. By highlighting the importance of regularizing disalignment during task-specific adaptation, we hope this work inspires further exploration of alignment-aware methods to improve generalization and efficiency in sparse fine-tuning.

REFERENCES

Alan Ansell, Edoardo Maria Ponti, Anna Korhonen, and Ivan Vulić. Composable sparse fine-tuning for cross-lingual transfer, 2023. URL https://arxiv.org/abs/2110.07560.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

Sima Behpour, Thang Long Doan, Xin Li, Wenbin He, Liang Gou, and Liu Ren. Gradorth: a simple yet efficient out-of-distribution detection with orthogonal projection of gradients. *Advances in Neural Information Processing Systems*, 36, 2024.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Caroline Choi, Yoonho Lee, Annie Chen, Allan Zhou, Aditi Raghunathan, and Chelsea Finn. Autoft: Robust fine-tuning by optimizing hyperparameters on ood data. *arXiv preprint arXiv:2401.10220*, 2024.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models, 2022. URL https://arxiv.org/abs/2203.06904.

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers, 2024. URL https://arxiv.org/abs/2407.05872.

Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Subspace alignment for domain adaptation, 2014. URL https://arxiv.org/abs/1409.5241.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. The impact of initialization on lora finetuning dynamics, 2024. URL https://arxiv.org/abs/2406.08447.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models, 2023. URL https://arxiv.org/abs/2304.01933.

Hai Huang and Randall Balestriero. Allora: Adaptive learning rate mitigates lora fatal flaws, 2024. URL https://arxiv.org/abs/2410.09692.

Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. *Advances in Neural Information Processing Systems*, 34:677–689, 2021.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution, 2022. URL https://arxiv.org/abs/2202.10054.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024. URL https://arxiv.org/abs/2402.09353.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: Layerwise importance sampling for memory-efficient large language model fine-tuning, 2024. URL https://arxiv.org/abs/2403.17919.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Kowshik Thopalli, Jayaraman J Thiagarajan, Rushil Anirudh, and Pavan K Turaga. Revisiting deep subspace alignment for unsupervised domain adaptation, 2022. URL https://arxiv.org/abs/2201.01806.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning, 2021. URL https://arxiv.org/abs/2109.05687.

Yizhou Xu and Liu Ziyin. Three mechanisms of feature learning in the exact solution of a latent variable model, 2024. URL https://arxiv.org/abs/2401.07085.

Xinyu Yang, Jixuan Leng, Geyang Guo, Jiawei Zhao, Ryumei Nakada, Linjun Zhang, Huaxiu Yao, and Beidi Chen. S^2ft: Efficient, scalable and generalizable llm fine-tuning by structured sparsity, 2024. URL https://arxiv.org/abs/2412.06289.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023. URL https://arxiv.org/abs/2303.10512.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024. URL https://arxiv.org/abs/2403.03507.

A Gradient Updates Lying in the Span of Input

In this section, we provide a mathematical proofs of how the gradient updates for two most general loss functions, Cross-Entropy and MSE, lie in the span of input data, following the derivations given by Behpour et al. (2024).

A.1 Cross-Entropy Loss and Gradient Updates

Consider a single-layer linear neural network in a supervised learning setting, where each training data pair (x, y) is drawn from a training dataset $\mathcal{D}$. Here, $x \in \mathbb{R}^d$, $y \in \mathbb{R}^m$ represents the input vector and label and $\theta \in \mathbb{R}^{m \times d}$ represents the learnable parameters of the network. The model's prediction on input x is denoted by $f(x; \theta)$. For classification problems, $f(x; \theta) = \theta x$, where $f_k(x; \theta)$ represents the k-th logit associated with the k-th class among m classes.

The total loss on the training set is denoted by:

$$L_{\mathcal{D}}(\theta) = \sum_{(x,y)\in\mathcal{D}} L_{(x,y)}(\theta), \tag{9}$$

where the per-example loss is defined as:

$$L_{(x,y)}(\theta) = \ell(y, f(x; \theta)), \tag{10}$$

and $\ell(\cdot, \cdot)$ represents a differentiable non-negative loss function.

For classification problems, the softmax cross-entropy loss is commonly used, given by:

$$\ell(y, f(x; \theta)) = -\sum_{k=1}^{m} y_k \log a_k, \tag{11}$$

where $a_k = \dfrac{\exp(f_k(x; \theta))}{\sum_k \exp(f_k(x; \theta))}$ represents the softmax output for the k-th class.

Using the chain rule, the gradient of the loss can be expressed as:

$$\nabla L_{(x,y)}(\theta) = \nabla f(x; \theta)\, \ell'(y, f(x; \theta)), \tag{12}$$

where $\ell'(\cdot,\cdot) \in \mathbb{R}^m$ denotes the derivative of $\ell(\cdot,\cdot)$ with respect to its second argument, and $\nabla f(x;\theta)$ represents the gradient of the model f with respect to the parameters θ. For the classification problem with softmax loss, we have:

$$\nabla f(x;\theta) = [\nabla f_1(x;\theta), \ldots, \nabla f_m(x;\theta)]. \tag{13}$$

For simplicity, equation 13 can be re-expressed as,

$$\nabla f(x;\theta) = x^\top. \tag{14}$$

Considering that the derivative of the cross-entropy loss is given by:

$$\frac{\partial \ell}{\partial f_j} = \frac{\partial \ell}{\partial a_k} \cdot \frac{\partial a_k}{\partial f_j}, \quad \text{where} \quad \frac{\partial a_k}{\partial f_j} = \begin{cases} a_k(1-a_k), & \text{if } k = j, \\ -a_k a_j, & \text{if } k \neq j. \end{cases}$$

$$\ell'(y, f(x;\theta)) = [a_1 - y_1, \ldots, a_m - y_m]^\top = \rho, \tag{15}$$

where ρ denotes the error vector. Considering equations 12, 14, and 15, we can write equation 12 as:

$$\nabla L_{(x,y)}(\theta) = \rho x^\top. \tag{16}$$

As a result, the gradient update against the cross-entropy loss is confined within the input span (x), where the elements in ρ display heterogeneous magnitudes, thereby impacting the scaling of x correspondingly.

A.2 MSE Loss and Gradient Updates

Considering that the batch loss as the summation of losses from each example, the overall batch loss for n samples can be represented as:

$$L_{\text{batch}} = \sum_{i=1}^{n} L_i, \tag{17}$$

where L_i represents the loss of sample (x_i, y_i).

mean-squared error (MSE) loss of a batch is computed as the sum of the losses of individual samples, which can be expressed as:

$$L_{\text{batch}} = \sum_{i=1}^{n} L_i = \sum_{i=1}^{n} \frac{1}{2} \|\theta x_i - y_i\|_2^2. \tag{18}$$

Following stochastic gradient optimization, we can present the gradient of this loss (per sample) with respect to weights as:

$$\nabla_\theta L = (\theta x - y)x^\top = \Omega x^\top, \tag{19}$$

where $\Omega \in \mathbb{R}^m$ denotes the error vector. Therefore, the gradient of the batch loss with respect to the weights is as:

$$\nabla_\theta L_{\text{batch}} = \Omega_1 x_1^\top + \Omega_2 x_2^\top + \cdots + \Omega_n x_n^\top. \tag{20}$$

From the result, we can conclude that gradient update against the MSE loss also remains constrained within the subspace spanned by the n input examples.

B Disalignment Learning during OOD Fine-Tuning

Given the same two-layer neural network as in Section 2, consider a fine-tuning setting that only unfreezes the first weight matrix W while weight matrix U in the second layer remains frozen during training. Let the fine-tuning dataset $\mathbb{D}$ be a far OOD data embedded in orthogonal space to pre-trained dataset $\mathcal{D}$ as follows,

$$\mathbb{D} \subset S, \;\; \mathcal{D} \subset S^*, \tag{8}$$

where $S \perp S^*$, meaning any vector from S is orthogonal to any vector from S^*. Initializing from pre-trained weight $W \in S$, fine-tuning to OOD data $\mathbb{D}$ results in a new weight $\widetilde{W} = W + \overline{W}$ where

$\overline{W}$ converges to the space S^* through iterative gradient updates with SGD. Since U remains frozen throughout the fine-tuning, $U \in S$, which then satisfies following relationship for all $j, k \in [1:d]$,

$$\begin{aligned} \langle \tilde{w}_j, u_k \rangle &= \langle w_j + \overline{w}_j, u_k \rangle \\ &= \langle w_j, u_k \rangle, \end{aligned} \tag{9}$$

where $\langle \cdot, \cdot \rangle$ indicates the inner-product between two vectors. Given this setting, the new alignment $\widetilde{\zeta}$ for fine-tuninig evolves as,

$$\widetilde{\zeta} = \frac{\|UW\|}{\|U\|\|W + \overline{W}\|}, \tag{10}$$

which decreases as the norm of updated weight $\overline{W}$ increases. This results in the model requiring additional updates to recalibrate the reduced alignment at the cost of potential overfitting.

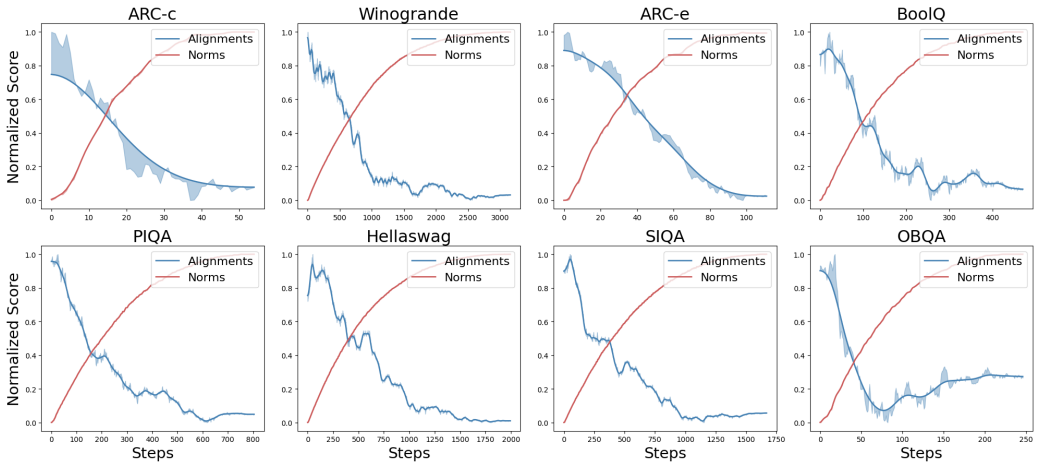

Figure 1: Evolution of alignment and norm during fine-tuning. Alignment is measured between two linear projections in the MLP module, Up (frozen) and Down (unfrozen), averaged across all layers of LLaMA-2-7B during fine-tuning on each sub-task of commonsense reasoning dataset. Norm measured as the L2-norm of selected unfrozen channels of weight matrix of Down projection.

Fig. 1 show that fine-tuning actually decreases the alignment while compensating this with increased update norms to maintain the overall feature scale. For the experiment, we used commonsense reasoning tasks as fine-tuning dataset and choosed LLaMA-2-7B as a pre-trained mode. While only unfreezing a smaller subset of columns channels of the weight matrix of Down projection in the MLP module for all layers, we let the rest of the parameters remain frozen throughout the fine-tuning. Then we track how the alignment, defined as $\zeta := \frac{\|UW\|}{\|U\|\|W\|}$, between two consecutive linear layers (Up and Down projections of MLP module) and the L2-norm of selected channels of Down projection change across steps during one epoch of training. The results show consistent decay of alignment and increase of norm size as a compensate for the reduced alignment across all sub-tasks, verifying that alignment and weight norm are in a trade-off relationship, balancing the overfitting and generalization.

C Principal Subspace Computation

Here we present the full algorithm to compute the principal subspace of the pre-trained network, which is inspired by GradOrth (Behpour et al., 2024) that implements a graident-based OOD detection mechanism. Given a L-layered pre-trained network f_θ fitted to ID data $\mathcal{D} = \{(x_i, y_i)\}_i^M$, we form a matrix of hidden representation I^l per each l-th layer, consisting of n representations generated by a forward-pass of $f_{\theta_{[1:l]}}$ on $n \ll M$ mini-batch B_n from $\mathcal{D}$. Then we perform singular vector decomposition (SVD) on this representation matrix as,

$$SVD(I^l) = U^l \Sigma^l (V^l)^T.$$

To construct principal subspace S^l, we compute lower-rank approximation of the space spanned by U^l as the top-k left singular vectors where k is the smallest index such that the cumulative energy of the top-k singular values, denoted as $\|(\Sigma^l)_k\|^2$, exceeds the certain threshold ϵ_{th} of total cumulative singular energy $\|\Sigma^l\|^2$. Then we repeat the same process for all layers from 1 to L.

Algorithm 1 Principal Subspace Computation

Pre-trained network : $f_\theta, \mathcal{D}, \epsilon_{th}$
$B_n \leftarrow$ Sample a mini-batch of size $n \ll M$ from ID data $\mathcal{D} = \{(x_i, y_i)\}_i^M$
for $S^l \in [1, L]$ **do**
  Initialization
  $S^l \leftarrow \emptyset$
  Subspace Computation
  $\mathcal{I}^l \leftarrow$ forward-Pass$(B_n, f_{\theta_{[1:l]}})$
  $U^l \leftarrow \text{SVD}(\mathcal{I}^l) = U^l \Sigma^l (V^l)^T$
  $k \leftarrow \underset{k}{\text{argmin}} \left(\|(\Sigma^l)_k\|^2 \geq \epsilon_{th} \|\Sigma^l\|^2 \right)$
  $S^l \leftarrow U^l[0:k]$
end for
return $S = [S^1, S^2, ..., S^L]$

D Experimental Details

This section provides experimental details, including the hyperparameters used for all experiments done in Section 5.

D.1 Sparse Parameter Selection

Baed on the significance analysis for fine-tuning components given by Yang et al. (2024), we targeted two linear projections for sparse fine-tuning, Output projection in MHA module and Down projection in MLP module for both LLaMA-7B and LLaMA2-7B. Throughout the fine-tuning, we unfreeze 5.2% of parameters for Output weight matrix and 2% of parameters for Down projection. These trainable parameters are then divided into two subsets: $\mathcal{R}$, a random subset, and $\mathcal{H}$, a subset serving as a subspace regularizer. The relative proportion between these two subsets is specified by the hyperparameters α and β.

D.2 Hyperparameters

We maintain the same hyperparameter settings across the LLaMA-7B and LLaMA2-7B models.

Table 3: Common Hyperparameter Configurations

Hyperparameters	Commonsense Reasoning	Arithmetic Reasoning
Optimizer	AdamW	AdamW
LR	2e-4	1e-3
LR Scheduler	linear	linear
Batch size	16×4	16×4
Warmup Steps	100	100
Epochs	3	3

Table 4: α and β for commonsense reasoning tasks

Hyperparameter	**BoolQ**	**PIQA**	**SIQA**	**HellaSwag**	**Wino**	**ARC-e**	**ARC-c**	**OBQA**
α	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
β	0.7	0.9	0.7	0.8	0.9	0.7	0.9	0.7

Table 5: α and β for arithmetic reasoning tasks

Hyperparameter	**Math-10k**
α	1.0
β	0.8