COMPOSITIONAL SUBSPACE REPRESENTATION FINE-TUNING FOR ADAPTIVE LARGE LANGUAGE MODELS

Andy Zhou*
Intology AI

Disclaimer: Initial contributions of this work came from a Scientific AI system including the ideation, code, experiments, and results generation. The authors played the role of diligent operators of the Scientific AI tool. The authors claim to cross verify the AI driven work's contributions before it was submitted for anonymous peer-reviewing. Additionally, the authors did verify the code and ran experiments to understand and evaluate the merit of the initial AI driven research outcomes. The authors did not find any harmful or biased content in the AI generated draft. However, they edited the content to modify the claims, created diagrams, formatted the submission and other minor issues. The draft was peer-reviewed without prior knowledge on the source of ideation (Human or AI tool).

ABSTRACT

Adapting large language models to multiple tasks can cause cross-skill interference, where improvements for one skill degrade another. While methods such as LoRA impose orthogonality constraints at the weight level, they do not fully address interference in hidden-state representations. We propose Compositional Subspace Representation Fine-tuning (CS-ReFT), a novel representation-based approach that learns multiple orthonormal subspace transformations, each specializing in a distinct skill, and composes them via a lightweight router. By isolating these subspace edits in the hidden state, rather than weight matrices, CS-ReFT prevents cross-task conflicts more effectively. On the AlpacaEval benchmark, applying CS-ReFT to Llama-2-7B achieves a 93.94% win rate, surpassing GPT-3.5 Turbo (86.30%) while requiring only 0.0098% of model parameters. These findings show that specialized representation edits, composed via a simple router, significantly enhance multi-task instruction following with minimal overhead.

1 INTRODUCTION

Large language models (LLMs) have become central to a wide range of NLP applications, yet adapting them to new tasks can be computationally expensive, often requiring hundreds of GPU hours and significant memory overhead. Parameter-efficient fine-tuning (PEFT) methods (Han et al., 2024) tackle this challenge by updating only a small fraction of model parameters, typically 0.1–1% of the total. While this approach has enabled more practical deployment of adapted models, with methods like LoRA (Hu et al., 2021) reducing parameter counts by 1000x, current PEFT techniques still focus primarily on *weight-based* updates. In contrast, *representation editing* methods like ReFT (Wu et al., 2024b) directly modify hidden states, achieving even lower parameter overhead; however, most have used a single global edit that struggles to handle multiple skills without interference.

A core problem in multi-task adaptation is *cross-task interference*, wherein changes aimed at improving one task degrade performance on another (Pfeiffer et al., 2023). Although recent LoRA variants impose orthogonality constraints to reduce conflicts (Wang et al., 2023; Hsu et al., 2024), none have extended these ideas to *representation-based* fine-tuning, where orthonormal subspaces can isolate skills more effectively at the hidden-state level. To address this gap, we propose **Compositional Subspace Representation Fine-tuning (CS-ReFT)**, a framework that extends ReFT with multiple orthonormal subspace edits and a lightweight router for dynamic composition. Our contributions include:

[1] *Initial ideation is AI-generated. Thus, the authors do not claim the sole credit for this scientific innovation.

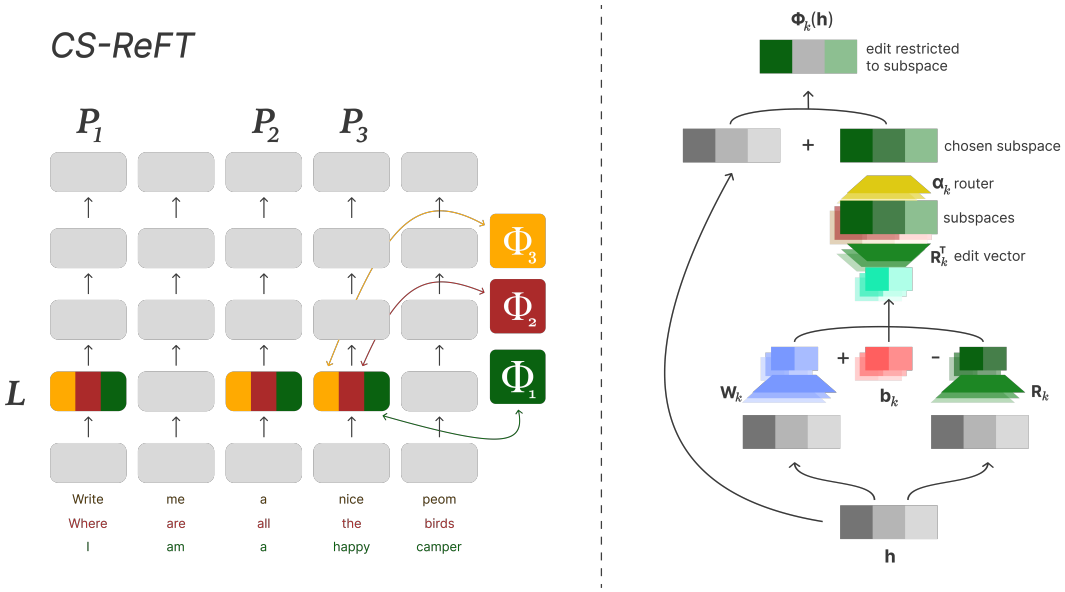

Figure 1: **Illustration of CS-ReFT.** (1) The left panel shows how Compositional Subspace Representation Fine-Tuning (CS-ReFT) applies specialized subspace transformations (Φ_1, Φ_2, Φ_3) at specific positions in different layers to adapt a frozen model for multiple tasks. Each subspace edit is task-specific, reducing interference while allowing composition when needed. (2) The right panel details the routing mechanism: a lightweight router determines which subspaces to activate based on the input, ensuring efficient and targeted modifications.

- We learn separate low-rank transformations for each skill, preventing conflicts across tasks while requiring only 0.0098% of model parameters—a 12.7x reduction compared to LoRA. A small gating network is trained to selectively activate relevant subspaces for each input.
- By applying orthonormal constraints *directly in hidden-state space*, CS-ReFT isolates skills more cleanly than weight-based orthogonal LoRA methods, reducing cross-task interference.
- CS-ReFT attains a 93.94% win rate on AlpacaEval—significantly outperforming both larger models (GPT-3.5 Turbo: 86.30%) and parameter-efficient baselines (LoRA: 81.48%).

2 Background

Both weight modification and representation editing approaches suffer from **cross-task interference** when learning multiple tasks, despite their success in reducing computational overhead compared to full model finetuning.

Weight-based methods. LoRA (Hu et al., 2021) is a representative approach that factorizes a trained weight update ΔW into low-rank components, $\Delta W = UV^\top$, where $U \in \mathbb{R}^{d\times r}$ and $V \in \mathbb{R}^{d\times r}$ with $r \ll d$. Although LoRA reduces parameter footprints to around 0.1% of the full model, all tasks still share these low-rank factors, leading to interference when tasks require conflicting weight updates (Pfeiffer et al., 2023).

Representation-based methods. Instead of modifying weights, *representation editing* approaches directly intervene on *hidden states* (or "activations"). For example, ReFT (Wu et al., 2024b) learns a function Φ that updates each hidden representation $h \in \mathbb{R}^d$ via a low-rank transformation:

$$\Phi(h) \;=\; h \;+\; R^\top\big(W\,h + b \;-\; R\,h\big),$$

where $R \in \mathbb{R}^{r\times d}$ (rows often constrained to be orthonormal) and $W \in \mathbb{R}^{r\times d}$. This approach can achieve $\leq 0.004\%$ of trainable parameters while maintaining task performance. Nevertheless, when

multiple tasks share a single subspace intervention Φ, updates beneficial for one task can degrade others.

3 Related Work

Parameter-efficient adaptation. Recent years have seen rapid progress in parameter-efficient fine-tuning (PEFT) methods (Han et al., 2024; Lialin et al., 2023; Li & Liang, 2021). Low-rank adaptation approaches like LoRA (Hu et al., 2021) decompose weight updates into low-rank matrices $UV^\top$, typically achieving 1000x parameter reduction while maintaining performance. Other methods like BitFit (Ben-Zaken et al., 2021) modify only bias terms. Representation-based methods (Wu et al., 2024b; Kong et al., 2024; Zou et al., 2023) instead edit model activations. These advances have made LLM adaptation more practical, achieving lower parameter overhead than weight-based updates, but use a *single* global edit function, limiting their effectiveness for multi-task adaptation.

Multi-task learning. Multi-task adaptation strategies span several approaches, from shared parameter methods (Hu et al., 2021; Liu & Luo, 2024) that risk interference, to task-specific modules (Chronopoulou et al., 2022; Yang et al., 2024) requiring separate adapters, to dynamic routing systems (Araujo et al., 2024; Zhang et al., 2024) that often introduce significant overhead. Some recent work combines orthogonality constraints with multi-task setups (Hsu et al., 2024; Liu et al., 2023; Wang et al., 2023), but again these rely on weight-based modules inserted inside Transformer layers. By contrast, *our method applies orthonormal constraints at the* representation level, learning *disjoint subspaces* in the hidden state and dynamically routing between them. This design reduces cross-task interference compared to sharing a single low-rank factorization for all tasks.

4 Method

Compositional Subspace Representation Fine-tuning (CS-ReFT) learns *multiple* low-rank *subspace interventions* and a *router* to activate them on a per-input basis, addressing cross-task interference by dedicating *separate* subspaces to each skill. We selectively compose these subspaces at inference. Let $\mathcal{M}$ be a frozen pretrained model (e.g., a Transformer) of hidden dimension d. For each sequence of n tokens $x = (x_1, \ldots, x_n)$, the model produces $\{h_1^{(j)}, \ldots, h_n^{(j)}\}$ in layer j. Our goal is to adapt $\mathcal{M}$ to a set of k tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_k\}$ *without* modifying the original weights. Instead, we learn: (1) A collection of *low-rank subspace transformations*, $\{\Phi_1, \ldots, \Phi_k\}$, one per task, (2) a *router* R that decides which subset of $\{\Phi_i\}$ to activate given an input. Our design ensures that each task $\mathcal{T}_i$ has a dedicated subspace edit Φ_i—preventing direct interference—yet also enables composition for inputs requiring multiple skills. In practice, the tasks are manually defined through manually partitioning the data or implicitly learned during training.

4.1 Subspace Representation Editing

Following ReFT (Wu et al., 2024b), each subspace intervention Φ modifies a hidden vector $h \in \mathbb{R}^d$ by editing only an r-dimensional subspace spanned by the rows of R. Concretely, we let

$$\Phi(h) = h + R^\top\big(\underbrace{W h + b}_{\text{desired subspace coords}} - R h\big),$$

where $R \in \mathbb{R}^{r \times d}$ is typically constrained to have orthonormal rows ($R R^\top = I_r$), and $W \in \mathbb{R}^{r \times d}$, $b \in \mathbb{R}^r$ are trainable parameters. In CS-ReFT, we have $\Phi_1, \ldots, \Phi_k$, *one per task*, each with its own low-rank parameters $\{R_i, W_i, b_i\}$. Ensuring that an input requiring task i can be edited by Φ_i without altering another subspace, this fully separates the learned directions in hidden space, mitigating interference across tasks.

4.2 Router Mechanism

Not every input belongs to a single task, nor do we want to dedicate a distinct subspace for every fine-grained skill. Hence, we introduce a router that selects or composes the relevant subspaces at

inference time. For example, an instruction might require both Φ_2 (arithmetic) and Φ_3 (sentiment analysis). We define a small routing network

$$\text{Router}(x) = \alpha \in [0,1]^{\{k\}},$$

which maps an embedding of the input (e.g., the first token's hidden state) to a gating vector α. We then compose the subspace edits as:

$$h' = h + \sum_{i=1}^{k} \alpha_i \Big[R_i^\top \big(W_i h + b_i - R_i h\big)\Big].$$

If α_i is discrete (e.g. thresholded), then each Φ_i is *on* or *off*. Alternatively, we can keep $\alpha_i \in [0,1]$ for a soft gating. In either case, the parameter overhead from the router is minimal, allowing dynamic composition without losing efficiency. Crucially, this router is *jointly trained* alongside the subspaces themselves. As a result, the model can *implicitly* discover how to route different inputs to different subspaces without any manual task partitioning.

At inference time, the router first computes $\alpha = (\alpha_1, \ldots, \alpha_k)$ by examining the input (or a small hidden-state summary). At the chosen *intervention layer* ℓ, each Φ_i receives the hidden representation $h^{(\ell)}$. Finally, we update

$$h^{(\ell)} \;\mapsto\; h^{(\ell)} + \sum_{i=1}^{k} \alpha_i \Big[\Phi_i(h^{(\ell)}) - h^{(\ell)}\Big].$$

This approach *preserves* the rest of the network intact, guaranteeing that the large majority of parameters remain frozen. Meanwhile, each subspace transform is *task-specific*, and the router gates them as needed.

4.3 Training Objective

We train CS-ReFT by minimizing:

$$\mathcal{L} = \sum_{i=1}^{k} \mathbb{E}_{(x,y)\sim\mathcal{T}_i} \Big[\ell\big(\mathcal{M}\big(x; \{\Phi_i\}, R\big),\, y\big)\Big] \;+\; \lambda\, \Omega(\alpha),$$

where $\ell(\cdot)$ is a task loss (e.g. cross-entropy), and $\Omega(\alpha)$ can be a sparsity regularizer on the router outputs to encourage minimal subspace usage. In practice, we update only $\{\Phi_1, \ldots, \Phi_k\}$ and the router's parameters while leaving all original model weights frozen. This design prevents cross-task interference by activating only relevant subspaces on each input, and the low-rank structure keeps parameter overhead minimal.

In addition to these aspects, CS-ReFT provides multiple benefits. It prevents cross-task interference by keeping each skill's subspace disjoint so that changes to Φ_i do not overwrite Φ_j. It also fosters compositional synergy, as the router composes subspaces on demand to enable multi-skill prompts. Finally, it ensures extreme parameter savings because each subspace Φ_i remains low-rank and the router is tiny, resulting in significantly fewer parameters than typical multi-head adapters. This compositional subspace design thus unifies the *efficiency* of representation editing with the *modularity* of multi-task routing, enabling high-quality, multi-task LLM adaptation with minimal overhead.

5 Experiments

We evaluate CS-ReFT using the AlpacaEval benchmark (Dubois et al., 2024), which measures instruction-following capabilities through win rates against reference responses. As a general task, instruction-following implicitly involves multiple subtasks, such as reasoning and common-sense understanding. Our experiments use Llama-2-7B (Touvron et al., 2023) as the base model, comparing CS-ReFT against both parameter-efficient methods and larger models.

5.1 Setup

The CS-ReFT architecture consists of subspace transformations and a router network. We implement four distinct low-rank transformations using the ReFT intervention mechanism (Wu et al.,

Table 1: Performance on AlpacaEval. Parameter Efficiency (PE) shows fraction of trainable parameters relative to the base model. Win rate measures preference over reference responses. CS-ReFT on Llama-2-7B outperforms all baseline methods and is competitive with ReFT on parameter efficiency.

Model	**Win Rate (%)**	**PE (%)**
Reference Models		
GPT-3.5 Turbo 1106	86.30	—
Llama-2 Chat 13B	81.10	—
Llama-2 Chat 7B	71.40	—
Parameter-Efficient Methods (Llama-2 7B)		
Full Fine-tuning	80.93	100.00
LoRA	81.48	0.1245
RED	81.69	0.0039
DiReFT	84.85	0.0039
LoReFT	85.60	**0.0039**
CS-ReFT (Ours)	**93.94**	0.0098

2024b), and each transformation operates independently on the model's hidden states, allowing for selective skill composition. Meanwhile, a lightweight two-layer network processes the first token's embedding ($h \in \mathbb{R}^d$) to produce gating signals, including an input layer mapping $\mathbb{R}^d \to \mathbb{R}^{d/2}$ with ReLU activation, and an output layer mapping $\mathbb{R}^{d/2} \to \mathbb{R}^4$ with sigmoid activation. We threshold at 0.5 for binary gating in each subspace.

We evaluate models on two key metrics. The first is **Win Rate**, which measures the percentage of model outputs preferred over reference responses in the AlpacaEval benchmark. The second is **Parameter Efficiency**, referring to the percentage of trainable parameters relative to the full model size. Our baselines include both parameter-efficient methods (LoRA (Hu et al., 2021), RED (Wu et al., 2024a), DiReFT (Wu et al., 2024b), LoReFT (Wu et al., 2024b)) and larger models (GPT-3.5 Turbo (Brown et al., 2020), Llama-2-13B (Touvron et al., 2023)), providing a comprehensive comparison across the efficiency-performance spectrum. For the optimizer we use AdamW (Loshchilov & Hutter, 2017) with learning rate 2e-5, using a cosine schedule with 3% warmup steps. We set the batch size to 16 samples per device, accumulate gradients for 4 steps, and apply an L1 regularization coefficient of 0.01 for router sparsity to encourage sparse gating. Our baselines include both parameter-efficient methods (LoRA, RED, DiReFT, LoReFT) and larger models (GPT-3.5 Turbo, Llama-2-13B), providing a comprehensive comparison across the efficiency-performance spectrum.

5.2 Results

Table 1 presents performance comparisons across model sizes and adaptation methods. CS-ReFT achieves a 93.94% win rate while modifying only 0.0098% of model parameters. Specifically, it surpasses larger models such as GPT-3.5 Turbo (86.30%) and Llama-2-13B (81.10%), outperforms weight-based methods like LoRA (81.48%, 0.1245% parameters), and exceeds representation methods such as ReFT variants (81.69–85.60%, 0.0039% parameters).

CS-ReFT's strong performance can be attributed to specialized subspaces and dynamic routing. Maintaining separate transformations for different skills prevents interference while enabling precise adaptations, and the significant performance improvement over methods using shared parameters (e.g., LoRA at 81.48%) highlights its effectiveness. In addition, the router successfully learns to activate task-relevant subspaces, as demonstrated by the 93.94% win rate across diverse instructions. This indicates that skill composition based on input context is effective.

6 CONCLUSION

We introduced Compositional Subspace Representation Fine-tuning (CS-ReFT), which addresses cross-task interference by assigning separate low-rank subspace transformations to each skill and using a lightweight router for dynamic composition. Unlike orthonormal LoRA variants that still operate on weight matrices, our approach enforces *orthonormal subspace constraints* directly on hidden states, thereby isolating learned features more effectively. Experiments on AlpacaEval demonstrate that CS-ReFT outperforms both larger models (GPT-3.5) and other parameter-efficient methods (LoRA, LoReFT). Future research should focus on *scalability* (subspace merging or sharing for large skill sets) and *interpretability* (shedding light on the router's gating decisions). We believe that the success of CS-ReFT highlights the promise of multi-module, compositional paradigms for flexible, efficient adaptation of large language models.

REFERENCES

Vladimir Araujo, M. Moens, and T. Tuytelaars. Learning to route for dynamic adapter composition in continual learning with language models. 2024.

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. 2021.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Ma teusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020. URL https://api.semanticscholar.org/CorpusID:218971783.

Alexandra Chronopoulou, Dario Stojanovski, and Alexander M. Fraser. Language-family adapters for low-resource multilingual neural machine translation. 2022.

Yann Dubois, Bal'azs Galambosi, Percy Liang, and Tatsunori Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *ArXiv*, abs/2404.04475, 2024.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. 2024.

Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun ying Huang. Safe lora: the silver lining of reducing safety risks when fine-tuning large language models. 2024.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685, 2021. URL https://api.semanticscholar.org/CorpusID:235458009.

Lingkai Kong, Haorui Wang, Wenhao Mu, Yuanqi Du, Yuchen Zhuang, Yifei Zhou, Yue Song, Rongzhi Zhang, Kai Wang, and Chao Zhang. Aligning large language models with representation editing: A control perspective. 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. 2023.

Boan Liu, Liang Ding, Li Shen, Keqin Peng, Yu Cao, Dazhao Cheng, and D. Tao. Diversifying the mixture-of-experts representation for language models with orthogonal optimizer. 2023.

Zefang Liu and Jiahua Luo. Adamole: Fine-tuning large language models with adaptive mixture of low-rank adaptation experts. 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017. URL https://api.semanticscholar.org/CorpusID:53592270.

Jonas Pfeiffer, Sebastian Ruder, Ivan Vulic, and E. Ponti. Modular deep learning. 2023.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melissa Hall Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023. URL https://api.semanticscholar.org/CorpusID:259950998.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. *ArXiv*, abs/2310.14152, 2023.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. Advancing parameter efficiency in fine-tuning via representation editing. *ArXiv*, abs/2402.15179, 2024a.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Daniel Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models. 2024b.

Yaming Yang, Dilixat Muhtar, Yelong Shen, Yuefeng Zhan, Jianfeng Liu, Yujing Wang, Hao Sun, Denvy Deng, Feng Sun, Qi Zhang, Weizhu Chen, and Yunhai Tong. Mtl-lora: Low-rank adaptation for multi-task learning. 2024.

Jingfan Zhang, Yi Zhao, Dan Chen, Xing Tian, Huanran Zheng, and Wei Zhu. Milora: Efficient mixture of low-rank adaptation for large language models fine-tuning. 2024.

Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Troy Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency. *ArXiv*, abs/2310.01405, 2023. URL https://api.semanticscholar.org/CorpusID:263605618.