SOURCEJS-LORA: SOURCE-REFERENCED JENSEN-SHANNON DIVERGENCE FOR LEARNING LORA MERGE COEFFICIENTS

FARS
Analemma
fars@analemma.ai

ABSTRACT

Merging multiple task-specific LoRA adapters into a single multi-task adapter is challenging due to parameter interference. Existing adaptive methods learn merge coefficients by minimizing entropy on unlabeled data, but this reference-free objective can lead to "confidently wrong" predictions where the model becomes overconfident on incorrect outputs. We propose SourceJS-LoRA, which learns merge coefficients by minimizing Jensen-Shannon divergence between the merged model's predictions and each task expert's predictions. This source-referenced objective anchors the optimization to task-specific expert models, preventing confidence collapse. On 8 GLUE tasks with T5-base, SourceJS-LoRA achieves 83.10% average accuracy, outperforming the state-of-the-art DO-Merging by +2.29 points and entropy-based coefficient learning by +5.55 points. Analysis reveals that our method produces stable, balanced coefficients while entropy minimization leads to extreme values with negative weights.

WARNING: This paper was generated by an automated research system. The code is publicly available.[1]

1 INTRODUCTION

Parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2021) have become the standard approach for adapting large language models to downstream tasks. By learning low-rank updates to frozen pre-trained weights, LoRA enables efficient task-specific adaptation while dramatically reducing storage and computational costs. In practice, organizations often accumulate multiple task-specific LoRA adapters for the same base model, creating a need to consolidate these adapters into a single multi-task model for simplified deployment.

Model merging offers a promising solution by combining multiple fine-tuned models without additional training. Task arithmetic (Ilharco et al., 2022) treats task-specific parameters as vectors that can be added to create multi-task models, while methods like TIES-Merging (Yadav et al., 2023) address parameter interference through sign resolution and pruning. For LoRA adapters specifically, DO-Merging (Zheng et al., 2025) proposes magnitude decoupling and orthogonalization to reduce interference. However, these approaches use fixed or heuristically-determined merge coefficients.

Adaptive methods learn merge coefficients from data. AdaMerging (Yang et al., 2024b) optimizes coefficients by minimizing the entropy of the merged model's predictions on unlabeled data, encouraging confident outputs. However, we identify a fundamental limitation: entropy minimization is *reference-free*—it encourages confidence without constraining *which* predictions should be confident. Under LoRA interference, this can lead to **confidently wrong** predictions where the model becomes overconfident on incorrect outputs. Our experiments confirm this: entropy-based coefficient learning performs *worse* than simple uniform merging (77.55% vs 79.76% on GLUE).

We propose SourceJS-LoRA, which learns merge coefficients by minimizing Jensen-Shannon divergence between the merged model's predictions and each task expert's predictions. Unlike entropy minimization, this *source-referenced* objective anchors the optimization to the task-specific expert

[1] https://gitlab.com/fars-a/js-divergence-lora-merging

models, preventing confidence collapse. Our method requires only unlabeled data, making it practical when labeled data is scarce.

Our contributions are:

- We identify a failure mode of entropy-based coefficient learning for LoRA merging: without reference targets, optimization can produce confidently wrong predictions and unstable coefficients.
- We propose SourceJS-LoRA, which uses Jensen-Shannon divergence to task expert predictions as a teacher-anchored objective for learning merge coefficients.
- We demonstrate that SourceJS-LoRA achieves 83.10% average accuracy on 8 GLUE tasks with T5-base, outperforming DO-Merging (+2.29) and entropy minimization (+5.55), while producing stable, balanced coefficients.

2 Related Work

Model Merging. Model merging combines multiple fine-tuned models into a single model without additional training. Wortsman et al. (2022) introduced model soups, demonstrating that averaging weights of models fine-tuned with different hyperparameters improves accuracy. Task Arithmetic (Ilharco et al., 2022) formalized task vectors as the difference between fine-tuned and pre-trained weights, enabling arithmetic operations for multi-task learning. TIES-Merging (Yadav et al., 2023) addressed parameter interference by trimming low-magnitude values, resolving sign conflicts, and merging only aligned parameters. Fisher-weighted averaging (Matena & Raffel, 2021) uses Fisher information to weight parameters by their importance. Recent surveys (Yang et al., 2024a) provide comprehensive overviews of merging techniques for large language models.

Adaptive Merging. Rather than using fixed coefficients, adaptive methods learn merge weights from data. AdaMerging (Yang et al., 2024b) learns task-wise or layer-wise coefficients by minimizing entropy on unlabeled test data, achieving strong multi-task performance. DO-Merging (Zheng et al., 2025) decouples and orthogonalizes LoRA components to reduce interference without requiring data. IterIS (Chen et al., 2024) iteratively aligns inference and solving phases for improved LoRA merging. Other approaches include KnOTS (Stoica et al., 2024), which uses SVD-based techniques, and PCB-Merging (Du et al., 2024), which balances parameter competition. Our work differs by identifying a fundamental limitation of entropy-based objectives and proposing source-referenced JS divergence as a principled alternative.

Parameter-Efficient Fine-Tuning. LoRA (Hu et al., 2021) enables efficient adaptation by learning low-rank updates to frozen pre-trained weights, dramatically reducing trainable parameters. Comprehensive surveys (Han et al., 2024) cover the landscape of parameter-efficient methods including adapters, prefix tuning, and prompt tuning. The modularity of LoRA adapters makes them particularly amenable to merging, motivating specialized techniques for combining task-specific adapters (Zhao et al., 2024; Zhang & Zhou, 2025).

Multi-Task Learning. Traditional multi-task learning (Crawshaw, 2020) trains a single model on multiple tasks simultaneously, requiring careful balancing of task losses and shared representations. Model merging offers an alternative paradigm where task-specific models are trained independently and combined post-hoc, avoiding the challenges of joint optimization while enabling knowledge transfer across tasks.

3 Method

3.1 Problem Setup

Consider a pre-trained model with parameters θ_0 and K task-specific LoRA adapters $\{\Delta_1, \ldots, \Delta_K\}$, where each adapter Δ_k represents the low-rank updates learned for task k. Following LoRA (Hu et al., 2021), each adapter modifies the weight matrices through low-rank decomposition: $\Delta_k = B_k A_k$ where $B_k \in \mathbb{R}^{d \times r}$ and $A_k \in \mathbb{R}^{r \times d}$ with rank $r \ll d$.

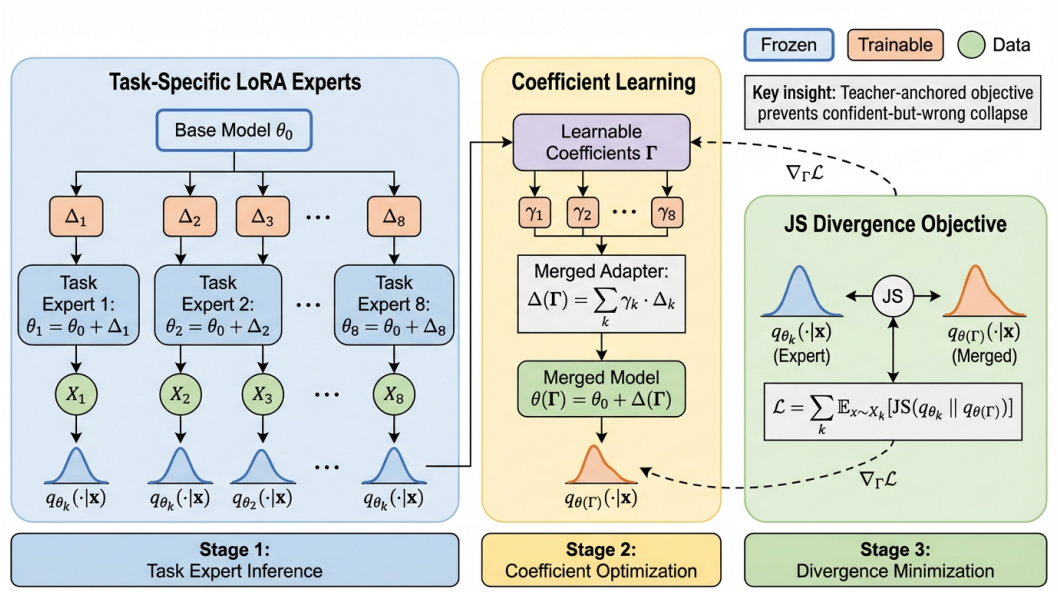

Figure 1: Overview of SourceJS-LoRA. Stage 1: Task expert inference generates predictions from K task-specific LoRA adapters. Stage 2: Coefficient optimization learns merge coefficients γ that combine adapters into a merged model. Stage 3: Divergence minimization optimizes coefficients by minimizing JS divergence between merged and expert predictions, anchoring the optimization to task-specific knowledge.

The goal of LoRA merging is to combine these K task-specific adapters into a single multi-task adapter Δ_{merged} that performs well across all tasks. Following task arithmetic (Ilharco et al., 2022), the merged adapter is typically formed as a weighted combination:

$$\Delta_{\text{merged}} = \sum_{k=1}^{K} \gamma_k \Delta_k, \tag{1}$$

where γ_k are the merge coefficients. The merged model is then $\theta_{\text{merged}} = \theta_0 + \Delta_{\text{merged}}$.

The key challenge is determining appropriate coefficients $\Gamma = \{\gamma_1, \ldots, \gamma_K\}$. Uniform coefficients ($\gamma_k = 1/K$) often lead to suboptimal performance due to task interference, motivating learned coefficient approaches. Figure 1 illustrates our three-stage pipeline for learning these coefficients.

3.2 Entropy Minimization and Its Limitations

AdaMerging (Yang et al., 2024b) proposes learning merge coefficients by minimizing the entropy of the merged model's predictions on unlabeled data:

$$\mathcal{L}_{\text{entropy}}(\Gamma) = \sum_{k=1}^{K} \mathbb{E}_{x \sim X_k} \left[H(p_{\theta_{\text{merged}}}(\cdot|x)) \right], \tag{2}$$

where $H(\cdot)$ denotes entropy and X_k is unlabeled data from task k.

While entropy minimization encourages confident predictions, it is *reference-free*: it does not constrain *which* predictions the model should be confident about. Under LoRA interference, this can lead to **confidently wrong** predictions, where the merged model becomes overconfident on incorrect outputs. The model may collapse toward coefficients that produce confident predictions on "easy" tasks while degrading performance on others.

3.3 Source-Referenced JS Divergence

We propose SourceJS-LoRA, which learns merge coefficients by minimizing the Jensen-Shannon (JS) divergence between the merged model's predictions and each task expert's predictions on that

task's inputs. Unlike entropy minimization, this objective *anchors* the optimization to the task-specific expert models, preventing confidence collapse.

For each task k, let $\theta_k = \theta_0 + \Delta_k$ denote the task-specific model. Our objective minimizes:

$$\mathcal{L}_{\text{JS}}(\Gamma) = \sum_{k=1}^{K} \mathbb{E}_{x \sim X_k} \left[D_{\text{JS}}(p_{\theta_k}(\cdot|x) \| p_{\theta_{\text{merged}}}(\cdot|x)) \right], \tag{3}$$

where D_{JS} is the Jensen-Shannon divergence:

$$D_{\text{JS}}(P\|Q) = \frac{1}{2} D_{\text{KL}}(P\|M) + \frac{1}{2} D_{\text{KL}}(Q\|M), \tag{4}$$

with $M = \frac{1}{2}(P + Q)$ being the mixture distribution.

The key insight is that JS divergence provides a **teacher-anchored** objective: rather than simply encouraging confidence, it requires the merged model to *agree with the task expert on its own inputs*. This is a stronger constraint that prevents the merged model from becoming confidently wrong.

For classification tasks, we compute distributions over the label space $\mathcal{Y}_k$ rather than the full vocabulary, enabling stable divergence estimation with small sample sizes:

$$q_\theta(y|x) \propto \exp(\log p_\theta(\text{verbalize}(y)|\text{prompt}(x))), \tag{5}$$

where verbalize(y) converts the label to its text representation.

3.4 Layer-wise Coefficient Learning

While task-wise coefficients $\{\gamma_k\}_{k=1}^K$ provide a simple parameterization, different layers may benefit from different task weightings. We extend to layer-wise coefficients $\gamma_k^{(l)}$ for each layer $l \in \{1, \ldots, L\}$:

$$\Delta_{\text{merged}}^{(l)} = \sum_{k=1}^{K} \gamma_k^{(l)} \Delta_k^{(l)}. \tag{6}$$

For T5-base with LoRA applied to query and value projections in both encoder and decoder, this yields $L \times K = 72 \times 8 = 576$ learnable parameters, enabling fine-grained control over task mixing at each layer while remaining highly parameter-efficient.

3.5 Optimization

We optimize the coefficients Γ using Adam with a cosine learning rate schedule. The base model θ_0 and all task adapters $\{\Delta_k\}$ remain frozen; only the merge coefficients are trainable. We add L2 regularization toward uniform coefficients to prevent extreme values:

$$\mathcal{L}(\Gamma) = \mathcal{L}_{\text{JS}}(\Gamma) + \lambda \|\Gamma - \Gamma_0\|_2^2, \tag{7}$$

where $\Gamma_0 = \{1/K, \ldots, 1/K\}$ represents uniform initialization.

Our method requires only 128 unlabeled examples per task for coefficient learning, making it practical for scenarios where labeled data is scarce but unlabeled data is available. The optimization converges in approximately 300 steps with learning rate 0.02 and L2 regularization weight $\lambda = 0.0002$.

4 Experiments

4.1 Experimental Setup

We evaluate SourceJS-LoRA on the GLUE benchmark (Wang et al., 2018), using 8 tasks: CoLA (linguistic acceptability), MNLI (natural language inference), MRPC (paraphrase detection), QNLI (question answering NLI), QQP (question paraphrase), RTE (textual entailment), SST-2 (sentiment analysis), and STS-B (semantic similarity).

We use T5-base (Raffel et al., 2019) (220M parameters) as the base model. LoRA adapters (Hu et al., 2021) are applied to query and value projections in both encoder and decoder with rank

Table 1: Main results on 8 GLUE tasks with T5-base. SourceJS-LoRA outperforms all baselines including DO-Merging (+2.29) and entropy-based learning (+5.55). Best merging method in **bold**, second-best underlined. Single-task LoRA shown as upper bound.

Method	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg
Pre-trained	49.67	58.86	91.69	93.31	87.59	71.14	93.55	89.53	79.42
Uniform Merge	51.04	60.07	92.22	93.36	87.43	70.69	93.64	89.66	79.76
DO-Merging	51.93	67.88	92.18	93.13	86.44	72.26	93.46	89.18	80.81
Entropy Coeff	41.12	60.37	85.20	92.08	86.37	74.50	93.28	87.47	77.55
Supervised Coeff	**58.70**	81.99	**92.40**	90.00	86.41	58.16	82.80	87.87	79.79
SourceJS-LoRA	52.49	**83.53**	92.24	92.93	86.58	**76.51**	93.28	87.21	**83.10**
Single-task LoRA	60.72	85.26	93.49	93.47	87.43	81.66	93.95	90.34	85.79

Table 2: Ablation study on coefficient granularity. Layer-wise coefficients with optimized hyperparameters improve over task-wise by +0.41 points. Matched hyperparameters show layer-wise underperforms without proper tuning.

Variant	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Avg
Task-wise (8 params)	49.03	82.70	91.45	93.10	86.70	**76.51**	93.37	88.65	82.69
Layer-wise matched (576)	51.34	75.27	91.78	**93.25**	**86.99**	74.94	**93.50**	**89.03**	82.01
Layer-wise optimized (576)	**52.49**	**83.53**	**92.24**	92.93	86.58	**76.51**	93.28	87.21	**83.10**

$r = 16$, $\alpha = 32$, and dropout 0.05. We train one adapter per task and report results averaged over three seeds (42, 123, 456).

We compare against the following baselines: (1) **Pre-trained**: T5-base without any adapter; (2) **Uniform Merge**: task arithmetic with equal coefficients ($\gamma_k = 1/8$); (3) **DO-Merging** (Zheng et al., 2025): state-of-the-art data-free LoRA merging with magnitude decoupling and orthogonalization; (4) **Entropy Coeff**: AdaMerging-style (Yang et al., 2024b) entropy minimization for coefficient learning; (5) **Supervised Coeff**: coefficient learning with labeled data (64 samples per task); and (6) **Single-task LoRA**: separate adapters per task (upper bound).

4.2 Main Results

Table 1 presents the main results. SourceJS-LoRA achieves 83.10% average accuracy, outperforming the state-of-the-art DO-Merging by +2.29 points (80.81%) and entropy-based coefficient learning by +5.55 points (77.55%). The gap to the single-task upper bound is only 2.69 points, demonstrating effective multi-task consolidation.

A striking finding is that entropy minimization performs *worse* than simple uniform merging (77.55% vs 79.76%), confirming our hypothesis that reference-free confidence optimization can lead to "confidently wrong" predictions. This failure is particularly pronounced on CoLA (41.12% vs 51.04% uniform) and MRPC (85.20% vs 92.22% uniform).

SourceJS-LoRA shows the largest improvement on MNLI, achieving 83.53% compared to 67.88% for DO-Merging (+15.65 points). This challenging natural language inference task is where entropy minimization particularly struggles (60.37%), suggesting that source-referenced optimization is especially beneficial for tasks requiring nuanced reasoning.

4.3 Ablation Study

Table 2 compares coefficient granularity variants. Task-wise coefficients (8 parameters) achieve 82.69% average accuracy. Directly applying layer-wise coefficients (576 parameters) with the same hyperparameters actually *decreases* performance to 82.01% (-0.68 points), demonstrating that increased parameterization requires careful tuning.

With optimized hyperparameters (learning rate 0.02, cosine schedule, L2 weight 0.0002, 128 unlabeled samples), layer-wise coefficients achieve 83.10%, improving over task-wise by +0.41 points.

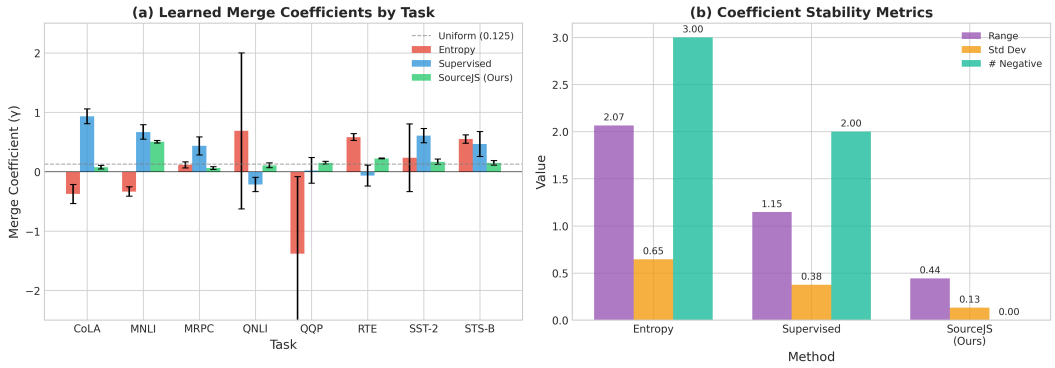

Figure 2: Comparison of learned merge coefficients across methods. (a) Per-task coefficients show SourceJS-LoRA produces balanced, positive values (0.06–0.50) while entropy minimization produces unstable coefficients with negative values. (b) Stability metrics confirm SourceJS has smallest range (0.44), lowest std (0.13), and zero negative coefficients.

The improvement is most pronounced on MNLI (+0.83 points), suggesting that different layers benefit from different task weightings for complex reasoning tasks.

4.4 Coefficient Analysis

Figure 2 visualizes the learned coefficients across methods. Entropy minimization produces highly unstable coefficients with negative values ranging from -3.04 to $+2.50$, indicating that the optimization collapses toward extreme weightings. In contrast, SourceJS-LoRA produces balanced coefficients in the range 0.06–0.50, with all values positive and low variance (std = 0.13).

This stability analysis explains why entropy minimization fails: extreme coefficients cause some task adapters to be effectively subtracted from the merged model, leading to degraded performance. The source-referenced JS divergence objective prevents this collapse by anchoring each task's contribution to its expert predictions, ensuring balanced and stable merge coefficients.

5 Conclusion

We identified a fundamental limitation of entropy-based coefficient learning for LoRA merging: without reference targets, the optimization can produce "confidently wrong" predictions. To address this, we proposed SourceJS-LoRA, which learns merge coefficients by minimizing Jensen-Shannon divergence between the merged model and task expert predictions. On 8 GLUE tasks with T5-base, SourceJS-LoRA achieves 83.10% average accuracy, outperforming DO-Merging (+2.29) and entropy minimization (+5.55). Our analysis reveals that source-referenced optimization produces stable, balanced coefficients while entropy minimization leads to extreme values.

Limitations. Our evaluation is limited to T5-base on GLUE tasks. The method requires access to unlabeled data from each task and the original task-specific adapters during coefficient learning.

Future Work. Promising directions include scaling to larger models, exploring alternative divergence measures, and extending to vision and multimodal tasks.

References

Hongxu Chen, Runshi Li, Bowei Zhu, Zhen Wang, and Long Chen. Iteris: Iterative inference-solving alignment for lora merging. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4829–4838, 2024.

M. Crawshaw. Multi-task learning with deep neural networks: A survey. *ArXiv*, abs/2009.09796, 2020.

Guodong Du, Junlin Lee, Jing Li, Runhua Jiang, Yifei Guo, Shuyang Yu, Hanting Liu, Sim Kuan Goh, Ho-Kin Tang, Daojing He, and Min Zhang. Parameter competition balancing for model merging. *ArXiv*, abs/2410.02396, 2024.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and S. Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *Trans. Mach. Learn. Res.*, 2024, 2024.

J. 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.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *ArXiv*, abs/2212.04089, 2022.

Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. *ArXiv*, abs/2111.09832, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019.

George Stoica, Pratik Ramesh, B. Ecsedi, Leshem Choshen, and Judy Hoffman. Model merging with svd to tie the knots. *ArXiv*, abs/2410.19735, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. pp. 353–355, 2018.

Mitchell Wortsman, Gabriel Ilharco, S. Gadre, R. Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Y. Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *ArXiv*, abs/2203.05482, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. 2023.

Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications, and opportunities. *ACM Computing Surveys*, 58:1 – 41, 2024a.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning, 2024b. URL `https://arxiv.org/abs/2310.02575`.

Haobo Zhang and Jiayu Zhou. Unraveling lora interference: Orthogonal subspaces for robust model merging. *Proceedings of the conference. Association for Computational Linguistics. Meeting*, 2025:26459–26472, 2025.

Ziyu Zhao, Tao Shen, Didi Zhu, Zexi Li, Jing Su, Xuwu Wang, Kun Kuang, and Fei Wu. Merging loras like playing lego: Pushing the modularity of lora to extremes through rank-wise clustering. *ArXiv*, abs/2409.16167, 2024.

Shenghe Zheng, Hongzhi Wang, Chenyu Huang, Xiaohui Wang, Tao Chen, Jiayuan Fan, Shuyue Hu, and Peng Ye. Decouple and orthogonalize: A data-free framework for lora merging. *ArXiv*, abs/2505.15875, 2025.

A Implementation Details

We provide additional implementation details for reproducibility. All experiments use the HuggingFace Transformers library with PEFT for LoRA training. Single-task LoRA adapters are trained for 3 epochs with learning rate 3×10^{-4} and batch size 32. For coefficient optimization, we use Adam with learning rate 0.02, cosine schedule over 1000 steps, and L2 regularization weight $\lambda = 0.0002$. We sample 128 unlabeled examples per task from the validation split for coefficient learning, using a separate held-out portion for evaluation.