LEWIS (LAYER WISE SPARSITY) - A TRAINING FREE GUIDED MODEL MERGING APPROACH

Hetarth Chopra, Vidhi Rambhia & Vikram Adve
Siebel School of Computing and Data Science
University of Illinois at Urbana Champaign
Urbana, IL 61820, USA
{hetarth2, vidhisr2, vadve}@illinois.edu

ABSTRACT

As specialized large language models (LLMs) become increasingly prevalent, model merging methods are being used to combine them to create a single multi-task model without requiring any additional data or training. However, these approaches fall short when the objective of merging is to increase the downstream model's performance on a particular task-specific benchmark. In this work, we propose LEWIS (**L**ay**E**r **WI**se **S**parsity), a guided model-merging framework that uses activation-based layer importance to dynamically adjust layer-wise task-vector sparsity required for the merge process. LEWIS uses a calibration dataset to prioritize critical layers during the task-vector pruning process required for model merging. This approach guides existing merging methods by preserving essential layer-wise task-specific knowledge while ensuring the merged model performs the best at benchmarks resembling the calibration dataset. Our experiments demonstrate the effectiveness of LEWIS with performance improvements of code instruction-following and math-solving models created through model merging up to 4% and 11.3%, respectively, outperforming unguided data-less model merging approaches that use uniform-sparsity.

1 INTRODUCTION

As specialized large language models (LLMs) fine-tuned for tasks such as math solving or instruction following become more prevalent, efficient model-merging methods have gained critical importance. State-of-the-art techniques like TIES (Yadav et al., 2024), DARE (Yu et al., 2024), and DeLLA (Deep et al., 2024) rely on task vectors (Ilharco et al., 2022)—parameter deltas between a pre-trained model and its fine-tuned variant—to merge models. Although these data-less strategies prune task vectors and fuse them into multi-task models, they often yield only moderate performance across tasks. To address this, recent works such as Model Breadcrumbs (Davari & Belilovsky, 2025), AdaMerging++ (Yang et al., 2023), and Localize and Stitch (He et al., 2024) have explored optimizing layer- or parameter-level importance to reduce task interference, but at a higher computational cost. Earlier model-merging methods, including simple averaging (Choshen et al., 2022), Fisher-weighted approaches (Matena & Raffel, 2022), and geometric-based solutions (Ainsworth et al., 2022; Stoica et al., 2023), often suffer from task interference or disregard crucial details like outlier activations. These outlier activations emerge in large-scale transformers (Dettmers et al., 2022) and can be 100 times larger than typical hidden states, making naive pruning detrimental to LLM performance (Sun et al., 2023; Wei et al., 2024). Empirical findings also indicate that different fine-tunes exhibit varying layer norms, magnitudes, and angles (Jang et al., 2025), hinting that layer-wise treatment can be beneficial. In this work, we propose a guided model-merging strategy that augments state-of-the-art methods (e.g., TIES (Yadav et al., 2024) and DARE (Yu et al., 2024)) by leveraging insights from Wanda pruning (Sun et al., 2023) and a calibration dataset to fine-tune layer-level task-vector sparsity. Our approach selectively preserves critical task-vector components in the most influential layers during merging, boosting performance on benchmarks that resemble the calibration data. Current merging methods create multi-task models that try to balance individual task performance - achieving an all-rounded performance - we aim to enhance this by adding an additional layer on top. We have released our code at https://github.com/VidhiRambhia/LEWIS

2 METHODOLOGY

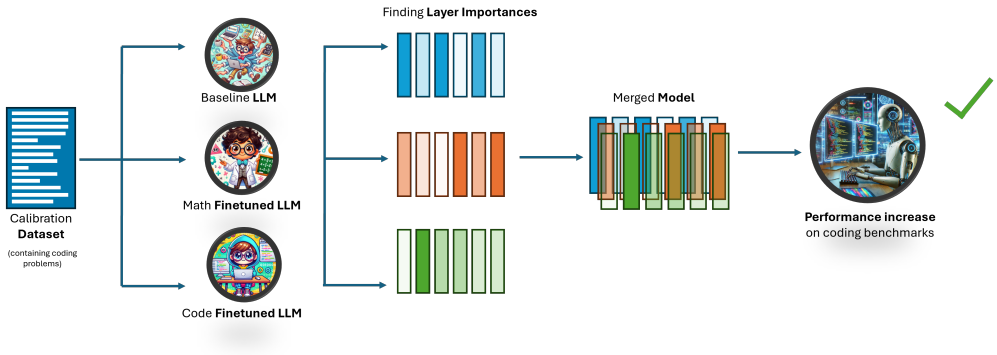

Figure 1: Process flow of the LEWIS framework: We show an example of how a calibration dataset (containing coding problems) can be used to compute layer-wise importance for a baseline LLM and it's finetunes, enabling selective ask-vector pruning and merging to perform best on benchmarks containing coding problems.

2.1 PRELIMINARY NOTATIONS

Let an LLM f be parameterized by θ such that $y = f(x;\theta)$, where x is an input prompt. The pre-trained model has parameters θ_0. Fine-tuning for task T yields $\theta_T = \theta_0 + \Delta\theta_T$, where $\Delta\theta_T$ captures the changes from θ_0. Now consider a series of models $\{\mathcal{M}_p\}_{p=1}^{P}$, each fine-tuned on a distinct task p. Each model $\mathcal{M}_p$ is:

$$\mathcal{M}_p = f(x;\theta_p), \quad \theta_p = \theta_0 + \Delta\theta_p.$$

Here, $\Delta\theta_p$ is the task vector (parameter changes for task p). We can combine fine-tuned models by summing their task vectors. For tasks $T_1, \ldots, T_n$:

$$\theta_{\text{merged}} = \theta_0 + \sum_{i=1}^{n} \alpha_i \, g(\Delta\theta_{T_i}),$$

where α_i are scaling coefficients (responsible for controlling per-model influence in the final merge) and $g(\cdot)$ is a pruning function (responsible for random/magnitude task-vector pruning function in DARE(Yu et al., 2024) and the trimming functionality in TIES (Yadav et al., 2024)). We introduce a calibration set $\mathcal{D} = \{x_i\}_{i=1}^{N}$ responsible for guiding the merging process by offering representative data on which we want θ_{merged} to perform effectively on.

2.2 LEWIS: MODEL MERGING USING LAYER IMPORTANCE

Inspired by how Wanda (Sun et al., 2023) works, LEWIS provides a layer importance of an LLM by comparing the activation norms of each layer in the fine-tuned and pre-trained models on a calibration dataset. Layers whose activations deviate more from the pre-trained model are deemed more critical and are pruned less aggressively during the merging process. This selective task-vector pruning ensures that the most critical layers retain higher densities across all fine-tuned models during the model-merging process. The entire methodology can be seen in Algorithm 1. We first pass the calibration dataset through each fine-tuned model to gather per-layer activations and compute average activation norms, which are compared against the pre-trained baseline (lines 2–12). The resulting differences are normalized and clipped, between empirically tested task-vector sparsity bounds $[\gamma, \epsilon]$, emphasizing layers with significant deviations (lines 14–24). These deviations guide a pruning function that retains parameters crucial for each task (lines 26–30). Finally, weighted task-specific parameter changes are combined to form a single merged model, which is more suited at doing well on a benchmark from which the calibration set $\mathcal{D}$ is sampled from.

Algorithm 1 Lewis: Model Merging

Require: θ_0: Pre-trained parameters, $\{M_p\}_{p=1}^{P}$: Fine-tuned models with $\{\theta_p\}_{p=1}^{P}$, $\mathcal{D} = \{x_i\}_{i=1}^{N}$: Calibration dataset, $[\gamma, \epsilon]$: Sparsity bounds
Ensure: θ_{merged}: Merged parameters

for $p = 1 \dots P$ **do**
  for $x_i \in \mathcal{D}$ **do**
    Get activations $A_{(p,l)}(x_i)$ for all layers l
  end for
end for
for $p = 1 \dots P, l$ **do**
  $\text{Norm}_{(p,l)} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \|A_{(p,l)}(x_i)\|$
end for
for l **do**
  $\text{Norm}_{\text{pre-trained},l} \leftarrow$ Compute similarly
end for
for $p = 1 \dots P, l$ **do**
  $\Delta A_{(p,l)} \leftarrow |\text{Norm}_{(p,l)} - \text{Norm}_{\text{pre-trained},l}|$
end for
$S \leftarrow \sum_l \Delta A_{(p,l)}$
for $p = 1 \dots P, l$ **do**
  $\Delta A_{(p,l)} \leftarrow \text{clip}(\Delta A_{(p,l)}/S, \gamma, \epsilon)$
end for
for $p = 1 \dots P, l$ **do**
  $g_{(l,p)} \propto \Delta A_{(p,l)}$
end for ▷ Higher $\Delta A_{(p,l)}$ = lower pruning rate
$\theta_{\text{merged}} \leftarrow \theta_0 + \sum_p \alpha_p \cdot g(\Delta\theta_p)$
return θ_{merged}

3 EXPERIMENTS

In this section, we evaluate how LEWIS can bootstrap and guide the TIES model merging process in two key scenarios - code instruction-following and math-solving tasks. To achieve this, we use a calibration dataset comprising 15 samples from the training splits of two popular benchmarks for both of these scenarios respectively - MBPP (Most Basic Python Programming) (Austin et al., 2021) and GSM8K (Grade School Math 8K). These samples help provide layer-wise importance scores, which inform the design of the pruning function $g(.)$ used to control task-vector sparsity prior to merging. During the merging process, layers with higher importance scores retain a greater fraction of their task-vectors. Sparsity is constrained within empirically determined task-vector sparsity bounds, $[\gamma, \epsilon]$, to maintain model performance. We compare TIES merging with uniform task-vector sparsity (0.5) for all the layers l having Q, K, V, O and MLP typical of the transformer architecture, as a model merging baseline, with LEWIS. Validation splits from the same benchmarks evaluate the code instruction-following and math-solving capabilities of the merged model. We leverage mergekit (Goddard et al., 2024) for the implementation of our methodology. Both Table 1 and 2 highlight our best-performing results in bold and the second-best results underlined.

3.1 SCENARIO 1: USING TIES TO CREATE BETTER CODE INSTRUCTION FOLLOWING MODELS

This experiment evaluates the effectiveness of merging Gemma-2b and Gemma-9b (Team et al., 2024) with their instruction fine-tuned counterparts using LEWIS as a guiding principle for TIES. The performance of models was assessed using Pass@1 and Pass@10 scores (Chen et al., 2021). Table 1 summarizes the results across various task-vector sparsity bounds selected empirically. For Gemma-2b LEWIS improves baseline TIES merging using a task-vector sparsity pruning bound of $[\gamma = 0.5, \epsilon = 0.8]$, by 1.3% Pass@1 and 4% Pass@10 scores. Similarly, for Gemma-9b, using LEWIS with a sparsity bound of $[\gamma = 0.3, \epsilon = 0.8]$ Pass@1 and Pass@10 scores are increased by 1.6% and 1.8% respectively.

Table 1: Performance comparison of Gemma and merged model across metrics.

Model	Merge Style	Sparsity bounds	Pass@1	Pass@10
Gemma-2b	N/A	N/A	0.2746	0.3603
Gemma-2b-Instruction	N/A	N/A	0.3446	0.3829
Gemma-2b + Gemma-2b-Instruction	Unguided TIES	Uniform 0.5 for all layers	0.3508	0.3850
	LEWIS guided TIES	$[\gamma = 0.5, \epsilon = 1]$	<u>0.3536</u>	<u>0.3962</u>
	LEWIS guided TIES	$[\gamma = 0.3, \epsilon = 0.8]$	0.3456	0.3840
	LEWIS guided TIES	$[\gamma = 0.5, \epsilon = 0.8]$	**0.3554**	**0.4004**
Gemma-9b	N/A	N/A	0.4881	0.5718
Gemma-9b-Instruction	N/A	N/A	0.5490	0.5762
Gemma-9b + Gemma-9b-Instruction	Unguided TIES	Uniform 0.5 for all layers	0.5324	0.5590
	LEWIS guided TIES	$[\gamma = 0.5, \epsilon = 1]$	<u>0.5389</u>	<u>0.5706</u>
	LEWIS guided TIES	$[\gamma = 0.3, \epsilon = 0.8]$	**0.5408**	**0.5769**
	LEWIS guided TIES	$[\gamma = 0.5, \epsilon = 0.8]$	0.5346	0.5617

3.2 Scenario 2: Using TIES to create beter math solving models

We evaluate the effectiveness of merging LLaMA 3.1 8b (Dubey et al., 2024) with Mathcoder (Wang et al., 2023) leveraging LEWIS to guide TIES merging. We chose Mathcoder, as it has the same architecture as LLaMA 3.1 model, and performs great across different math-solving tasks. The performance of the models was evaluated using Flexible-Extract (FE) [1] and Strict-Match (SM) [2] metrics. Table 2 summarizes results for baseline models and merged configurations of Llama-3.1-8b and Mathcoder on the GSM8k benchmark. The best performance was achieved with LEWIS-guided sparsity in the range $[\gamma = 0.5, \epsilon = 0.8]$, outperforming the baseline uniform sparsity TIES method by 11.3% in FE and 11.2% in SM.

Table 2: Comparison of baseline performance for LLaMA 3.1 8b and Mathcoder with results from merged models under different sparsity configurations.

Model	Merge Style	Sparsity bounds	FE	SM
LLaMA 3.1 8b	N/A	N/A	0.4943	0.4928
Mathcoder	N/A	N/A	0.6300	0.6262
Llama+ Mathcoder	Unguided TIES	Uniform 0.5 for all layers	0.5625	0.5595
	LEWIS guided TIES	$[\gamma = 0.5, \epsilon = 1]$	<u>0.6240</u>	<u>0.6217</u>
	LEWIS guided TIES	$[\gamma = 0.3, \epsilon = 0.8]$	0.5390	0.5390
	LEWIS guided TIES	$[\gamma = 0.5, \epsilon = 0.8]$	**0.6262**	**0.6224**

3.3 Conclusion

In this work, we introduced a novel, guided model-merging strategy that builds on top of state-of-the-art merging methods by incorporating layer-wise importance scores. We then leverage calibration data to compute layer-wise activation norms in fine-tuned models, identifying critical parameters that should be preserved during task-vector pruning. This approach mitigates limitations of uniform or purely magnitude-based pruning, thus retaining essential task vectors for improved performance on target benchmarks. Our experiments show its efficiency on both code instruction-following and math-solving tasks: Pass@10 improves by 4% and 1.8% for Gemma-2b and 9b, respectively, and FE scores increase by 11.3% when merging LLaMA 3.1 8b with Mathcoder. These results show LEWIS's capacity to enhance merged models while preserving critical layer-wise knowledge. Future work will explore automated methods for determining task-vector sparsity bounds and extending this approach to broader task domains, model architectures, and merging strategies.

[1] **Flexible-Extract:** This metric captures numeric answers from text in a broad and adaptable way by using a regex pattern that identifies numbers in diverse formats, such as those with dollar signs, commas, or decimals (e.g., 1,234.56)

[2] **Strict-Match:** This metric enforces stricter criteria, requiring an exact match to a predefined answer format (e.g., "The answer is -123.45").

REFERENCES

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*, 2022.

MohammadReza Davari and Eugene Belilovsky. Model breadcrumbs: Scaling multi-task model merging with sparse masks. In *European Conference on Computer Vision*, pp. 270–287. Springer, 2025.

Pala Tej Deep, Rishabh Bhardwaj, and Soujanya Poria. Della-merging: Reducing interference in model merging through magnitude-based sampling. *arXiv preprint arXiv:2406.11617*, 2024.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35: 30318–30332, 2022.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's MergeKit: A toolkit for merging large language models. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 477–485, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.36. URL https://aclanthology.org/2024.emnlp-industry.36.

Yifei He, Yuzheng Hu, Yong Lin, Tong Zhang, and Han Zhao. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *arXiv preprint arXiv:2408.13656*, 2024.

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

Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. In *European Conference on Computer Vision*, pp. 207–223. Springer, 2025.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

George Stoica, Daniel Bolya, Jakob Bjorner, Pratik Ramesh, Taylor Hearn, and Judy Hoffman. Zipit! merging models from different tasks without training. *arXiv preprint arXiv:2305.03053*, 2023.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*, 2023.

Boyi Wei, Kaixuan Huang, Yangsibo Huang, Tinghao Xie, Xiangyu Qi, Mengzhou Xia, Prateek Mittal, Mengdi Wang, and Peter Henderson. Assessing the brittleness of safety alignment via pruning and low-rank modifications. *arXiv preprint arXiv:2402.05162*, 2024.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*, 2023.

Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. Learning to mine aligned code and natural language pairs from stack overflow. In *Proceedings of the 15th international conference on mining software repositories*, pp. 476–486, 2018.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*, 2024.

A Appendix

This section includes additional ablations that build on these experiments and provide a more comprehensive view of the merging process. The experiments used NVIDIA GPUs - 4xA100s for the LLM Evaluation; and 2xRTX5000s for LEWIS and Model Merging process.

A.1 Strategy of Selecting Task-Vector Sparsity Bounds $[\gamma, \epsilon]$

The performance of the merged model is dependent on the hyper-parameters $[\gamma, \epsilon]$, which are crucial in balancing knowledge retention and sparsity. Excessively pruning the task vectors of a fine-tuned model can reduce the knowledge it imparts to the final merge. Conversely, as shown by TIES, pruning the task vectors less can lead to interference. Therefore, a strategy is to prune task vectors by magnitude within a particular range, $[\gamma, \epsilon]$, and an empirical grid search strategy is recommended to find values that work for your use case. Below is an intuition of why the different sparsity configurations were selected for experimentation in our case:

- $[\gamma = 0.3, \epsilon = 0.8]$: Retains important parameters within task-vectors while pruning redundant ones.
- $[\gamma = 0.5, \epsilon = 1.0]$: A more aggressive pruning setting to minimize interference in task merging.
- $[\gamma = 0.5, \epsilon = 0.8]$: An overlap of the above ranges.

A.2 Selectively Merging the Top Most Important Layers

To assess the impact of layer-wise sparsity during the model merging process, we explored a partial merging strategy. This technique involves merging the top $k\%$ most important layers with a density of 1.0, while the remaining layers are merged with a negligible density of 0.1. The results for this experiment, conducted on the `gemma-2b-it` model, are presented in Table 3

The results demonstrate that merging based on layer importance, particularly when top-$k\%$ layers are fully preserved, can yield performance improvements. Specifically, for $k = 0.5$, the `Pass@10` score increased by 5.2% over the TIES baseline at a uniform density of 0.5. These findings highlight the importance of effectively understanding layer-wise contributions and leveraging them to optimize model merging strategies.

Table 3: Performance comparison of TIES baseline (uniform density) and partial merging strategy with top-$k\%$ important layers as determined by LEWIS.

Merge Strategy	Configuration	Pass@1	Pass@10
Unguided TIES	Uniform Sparsity 0.5 for all layers	0.3508	0.3850
Merge Top-$k\%$ Layers+ as determined by LEWIS	$k = 40\%$	0.3378	**0.3943**
	$k = 50\%$	0.3419	**0.4052**
	$k = 60\%$	0.3429	**0.3973**
	$k = 70\%$	**0.3556**	**0.4013**
	$k = 80\%$	**0.3536**	**0.4022**

Table 4: Performance of Selected Layer Merge strategy with 100% density for specific layers and 1% for others, compared to Unguided TIES.

Merge Strategy	Configuration	Pass@1	Pass@10
Unguided TIES	Uniform Sparsity 0.5 for all layers	0.3508	0.3850
Selected Layer Merge	Only MLP	0.3410	**0.3927**
	Only Q	0.2689	0.3335
	Only K	0.2709	0.3323
	Only V	0.2480	0.3147
	Only O	0.2608	0.3247

A.3 Selectively Merging Layers of A Specific Type

In this experiment, 100% of the task vectors from a specific layer type (`Q`, `K`, `V`, `MLP`) were retained during merging, while the remaining layers were pruned to 1% sparsity. The results for this experiment are summarized in Table 4. This experiment was performed using the `Gemma-2b + Gemma-2b-Instruction` models, calibrated with 15 samples from the MBPP dataset.

The results reveal that merging layers selectively can significantly influence performance. Notably, retaining the MLP layers at full density while pruning others yields the highest scores for `Pass@10`. This suggests that MLP layers play a critical role in preserving task-specific knowledge and merit further exploration. Conversely, merging only the attention-related layers (`Q`, `K`, `V`, and `O`) results in comparatively lower performance, underscoring the importance of MLP layers in the model's ability to generalize during the merge process. These insights emphasize the need to account for layer-specific contributions when designing model merging strategies.

A.4 Scenarios 1 with DARE

To further validate the performance of layer-wise guided model merging, we repeated the experiments from Scenario 1 (focused on MBPP tasks) using another state-of-the-art merging method, DARE. The performance of Gemma models, including their code instruction-tuned variants and merged configurations, was evaluated using the DARE method with different sparsity configurations. Results, calibrated with 15 MBPP samples, are summarized in Table 1.

When merging Gemma-2b and Gemma-2b-Instruction, the Unguided DARE method performed worse than the fine-tuned model, yielding 0.3321 (`Pass@1`) and 0.3710 (`Pass@10`). However, using LEWIS-guided sparsity, we observed best performance with $[\gamma = 0.5, \epsilon = 0.8]$, achieving 0.3459 (`Pass@1`) and 0.3888 (`Pass@10`), which represents a 4.2% increase in `Pass@1` and 4.8% increase in `Pass@10` compared to the unguided method. For Gemma-9b, fine-tuning improved its scores to 0.5490 (`Pass@1`) and 0.5762 (`Pass@10`). Merging Gemma-9b with its instruction-tuned variant using DARE followed a similar trend with the best performance observed with $[\gamma = 0.5, \epsilon = 1]$, achieving 0.5377 (`Pass@1`) and 0.5724 (`Pass@10`), improving over the unguided DARE method (0.5316, 0.5555) by 1.1% in `Pass@1` and 3.0% in `Pass@10`.

Table 5: Performance comparison of Gemma models, instruction-tuned variants, and models merged with DARE across metrics. Bold results indicate the best performance for each metric, while underlined results represent the second-best performance.

Model	Merge Style	Sparsity bounds	Pass@1	Pass@10
Gemma-2b	N/A	N/A	0.2746	0.3603
Gemma-2b-Instruction	Finetuned	N/A	0.3446	0.3829
Gemma-2b + Gemma-2b-Instruction	Unguided DARE	Uniform	0.3321	0.3710
	LEWIS guided DARE	$[\gamma = 0.5, \epsilon = 1]$	<u>0.3424</u>	<u>0.3915</u>
	LEWIS guided DARE	$[\gamma = 0.3, \epsilon = 0.8]$	0.3335	0.3839
	LEWIS guided DARE	$[\gamma = 0.5, \epsilon = 0.8]$	**0.3459**	**0.3888**
Gemma-9b	N/A	N/A	0.4881	0.5718
Gemma-9b-Instruction	Finetuned	N/A	0.5490	0.5762
Gemma-9b + Gemma-9b-Instruction	Unguided DARE	Uniform	<u>0.5316</u>	0.5555
	LEWIS guided DARE	$[\gamma = 0.5, \epsilon = 1]$	**0.5377**	**0.5724**
	LEWIS guided DARE	$[\gamma = 0.3, \epsilon = 0.8]$	0.4843	0.5178
	LEWIS guided DARE	$[\gamma = 0.5, \epsilon = 0.8]$	0.5290	<u>0.5564</u>

A.5 Scenario 2 With DARE

We also repeated the experiments from Scenario 2 (focused on math-solving tasks) using DARE. Similar to TIES, this experiment was conducted with models calibrated using 15 samples from the GSM8k dataset, and the results are presented in Table 6.

Table 6: Comparison of baseline performance for LLaMA 3.1 8b and Mathcoder with results from merged models with DARE under different sparsity configurations. The table highlights the best-performing results shown in bold and the second-best results underlined.

Model	Merge Style	Sparsity bounds	FE	SM
LLaMA 3.1 8b	N/A	N/A	0.4943	0.4928
Mathcoder	N/A	N/A	0.6300	0.6262
Llama+ Mathcoder	Unguided DARE	Uniform 0.5 for all layers	0.0622	0.0531
	LEWIS guided DARE	$[\gamma = 0.5, \epsilon = 1]$	**0.6240**	**0.6217**
	LEWIS guided DARE	$[\gamma = 0.3, \epsilon = 0.8]$	<u>0.5390</u>	<u>0.5390</u>
	LEWIS guided DARE	$[\gamma = 0.5, \epsilon = 0.8]$	0.3230	0.3184

However, incorporating LEWIS-guided sparsity restored performance dramatically. The best performance was observed with $[\gamma = 0.5, \epsilon = 1]$, reaching 0.6240 (`FE`) and 0.6217 (`SM`), a near full recovery to Mathcoder's fine-tuned performance. The second-best configuration, $[\gamma = 0.3, \epsilon = 0.8]$, also showed strong results with 0.539 (`FE`) and 0.539 (`SM`), significantly outperforming the unguided merging approach. Uniform Pruning with DARE underperformed guided pruning across all configurations, highlighting the importance of structured pruning strategies in preserving task-specific knowledge. These findings reinforce the generalization of LEWIS across different merging methods like TIES and DARE.

A.6 Scenario 3: Improving performance of CoNaLa Dataset

To test robustness of our methdology we also tested our idea on another-code instruction following dataset - CoNaLa (Yin et al., 2018). We merged Gemma-2b and Gemma-9b with Gemma-2b-Instruction and Gemma-9b-Instruction, which have already been fine-tuned with code-instruction following data sample, using the same hyperparameters for sparsity and number of samples used for determining parameter importance. The results are presented in Table 7.

incorporating LEWIS-guided sparsity yielded notable gains in CoNaLa evaluation metrics. For the Gemma-2b merged models, the LEWIS-TIES configuration with sparsity bounds $[\gamma = 0.5, \epsilon = 1]$ achieved the highest BLEU score of 0.2849. In the larger Gemma-9b setting, the LEWIS-TIES configuration with $[\gamma = 0.3, \epsilon = 0.8]$ delivered the best overall performance, achieving a BLEU score

Table 7: Comparison of baseline performance and merged models on the CoNaLa evaluation task under different sparsity configurations. The table highlights the best-performing BLEU and 4-gram precision scores in bold and the second-best scores underlined.

Model	**Merge Style**	**Sparsity Bounds**	**BLEU**	**Precision**
Gemma-2b	No Merge	N/A	0.2441	0.2312
Gemma-2b-Instruction	No Merge	N/A	0.2814	0.2921
Gemma-2b + Gemma-2b-Instruction	Unguided TIES	N/A	0.2768	**0.2989**
	LEWIS-TIES	$[\gamma = 0.5, \epsilon = 1]$	**0.2849**	<u>0.2929</u>
	LEWIS-TIES	$[\gamma = 0.3, \epsilon = 0.8]$	0.2804	0.2841
	LEWIS-TIES	$[\gamma = 0.5, \epsilon = 0.8]$	<u>0.2822</u>	0.2901
Gemma-9b	No Merge	N/A	0.3643	0.3729
Gemma-9b-Instruction	No Merge	N/A	0.3768	0.3275
Gemma-9b + Gemma-9b-Instruction	Unguided TIES	N/A	<u>0.3802</u>	<u>0.3449</u>
	LEWIS-TIES	$[\gamma = 0.5, \epsilon = 1]$	0.3738	0.33953
	LEWIS-TIES	$[\gamma = 0.3, \epsilon = 0.8]$	**0.3867**	**0.3460**
	LEWIS-TIES	$[\gamma = 0.5, \epsilon = 0.8]$	0.3737	0.3389

of 0.3867 along with a peak precision score of 0.3460; both surpassing the results from unguided merging and other guided configurations.

ACKNOWLEDGMENTS

This work was supported in part by the IBM-IL Discovery Accelerator Institute (IIDAI) Undergraduate Research Experience (URE) Grant, under the project "Sound Code Creation Using Program Synthesis and Language Models" (Award No. 114160), and by the National Science Foundation under grant CNS 23-05883. We are also grateful to the National Center for Supercomputing Applications (NCSA) for providing access to their Delta Cluster and A100 NVIDIA GPUs, which were essential for the experiments conducted in this study.