UNMERGE: Verifiable Model Capability Attribution via Sparse Coding

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Model merging has emerged as a powerful technique for combining specialized capabilities from multiple fine-tuned models. However, the inverse problem (decomposing merged models back into their constituent capabilities) remains largely unexplored, limiting our ability to verify and understand model compositions. We introduce UNMERGE, a framework for model capability attribution that treats fine-tuned capabilities as sparse combinations of known micro-task vectors from a pre-built dictionary. Through comprehensive experiments across 15 tasks, 72 merged models were created with 4 different merging methods. Out of 6 decomposition algorithms, Non-negative Least Squares (NNLS) and Orthogonal Matching Pursuit (OMP) achieve exceptional performance with perfect precision and recall for models composed entirely of known tasks. While we focus on parameter-space reconstruction as a necessary first step, we discuss the important relationship between parameter fidelity and functional performance, acknowledging behavioral validation as crucial future work. Our framework enables controlled verification of model compositions and provides a foundation for future work in neural network interpretability and capability attribution.

1 Introduction

The rapid advancement of large language models has led to increasingly sophisticated techniques for combining specialized capabilities from multiple models. Model merging methods such as Task Arithmetic [Ilharco et al., 2022], TIES [Yadav et al., 2023], and DARE [Yu et al., 2023] enable practitioners to create unified models that retain diverse skills without additional training. However, these techniques operate as black boxes, making it difficult to verify which capabilities are present in a merged model or to attribute specific behaviors to their constituent components.

This opacity poses significant challenges for model interpretability, safety verification, and intellectual property protection. When a merged model exhibits unexpected behavior, practitioners lack tools to determine which original components contributed to the outcome. Similarly, in scenarios where model provenance matters (such as academic research or commercial applications) there is no systematic way to verify that a model contains only the intended capabilities.

We address this gap by introducing UNMERGE, a verifiable framework for model capability attribution that enables the decomposition of merged models back into their constituent task-specific components. Our approach treats a model's fine-tuned capabilities, represented as task vectors, as sparse combinations of known "micro-task" vectors from a pre-built dictionary.

Key Contributions:

- We formalize the model decomposition problem as sparse coding over a dictionary of known task vectors, enabling verifiable capability attribution.

Submitted to 1st Open Conference on AI Agents for Science (agents4science 2025). Do not distribute.

- We develop a comprehensive experimental framework with 72 merged models across three categories (known, mixed, unknown compositions) to evaluate decomposition performance.
- We demonstrate that NNLS achieves perfect precision/recall for known compositions.
- We provide extensive analysis of decomposition algorithm performance across different merging methods, identifying Task Arithmetic as optimal for decomposable merging.
- We establish a foundation for future work in neural network interpretability through parameter-space decomposition while discussing the relationship to behavioral validation.

Our work opens new directions for understanding and verifying model compositions, with applications ranging from model auditing to intellectual property protection and scientific reproducibility.

2 Related Work

Our work intersects several key research areas in neural network analysis and interpretability.

2.1 Model Merging and Task Arithmetic

Task arithmetic [Ilharco et al., 2022] introduced the foundational concept that model capabilities can be manipulated through parameter-space operations. This work demonstrated that task vectors (computed as the difference between fine-tuned and base model parameters) can be added, subtracted, and scaled to transfer or remove capabilities. TIES-Merging [Yadav et al., 2023] addressed interference problems in naive parameter averaging by resolving conflicts through magnitude-based selection. DARE [Yu et al., 2023] introduced drop-and-rescale techniques to reduce redundancy in merged models.

These methods operate under the assumption that model capabilities compose linearly in parameter space, a hypothesis that our decomposition framework both leverages and validates. However, no prior work has systematically studied the inverse problem of decomposing merged models back into constituent components.

2.2 Sparse Coding and Dictionary Learning

Sparse coding has a rich history in signal processing [Olshausen and Field, 1996] and has been extensively studied in machine learning contexts [Elad, 2010]. Recent work has applied sparse coding principles to neural network analysis, particularly in mechanistic interpretability.

Anthropic's work on monosemanticity [Bricken et al., 2023, Templeton et al., 2024] demonstrated that sparse autoencoders can decompose neural activations into interpretable features. These approaches operate in activation space and focus on understanding individual neuron behaviors rather than parameter-level decomposition.

Kim et al. [Kim et al., 2020] showed that neural networks trained with sparse coding constraints yield more interpretable representations. Most recently, Braun et al. [Braun et al., 2025] introduced Attribution-based Parameter Decomposition (APD), which directly decomposes neural network parameters into mechanistic components. Our work extends this direction by focusing specifically on task vector decomposition with verifiable ground truth obtained by existing merging methods.

2.3 Neural Network Interpretability

Network dissection [Bau et al., 2017, 2020] established frameworks for quantifying interpretability by evaluating alignment between network components and semantic concepts. Mechanistic interpretability [Elhage et al., 2021, Wang et al., 2022] aims to reverse-engineer neural networks to understand their computational mechanisms.

Recent advances in automated circuit discovery [Conmy et al., 2023] and attribution patching [Syed et al., 2023] provide tools for identifying functional components within neural networks. Our parameter-space decomposition approach complements these activation-space methods by operating directly on model weights.

2.4 Parameter-Space vs. Function-Space Analysis

A fundamental distinction in neural network analysis lies between parameter-space and function-space approaches. Parameter-space methods [Braun et al., 2025] analyze model weights directly, while function-space methods evaluate behavioral outputs. While both approaches are valuable, they address different aspects of model understanding.

Parameter-space analysis offers computational efficiency and mathematical tractability, making it feasible to analyze large models without extensive inference. However, the relationship between parameter similarity and functional equivalence remains an open research question. Studies on model editing [Mitchell et al., 2022, Meng et al., 2022] suggest that localized parameter changes can have far-reaching functional impacts, while other work indicates that parameter-space structure often reflects functional organization [Zhang et al., 2024].

Our work focuses on parameter-space reconstruction as a necessary first step toward understanding model compositions. We acknowledge that behavioral validation represents crucial future work and discuss this relationship in detail.

3 Method

Our approach decomposes merged models into constituent task-specific components through sparse coding over a pre-built dictionary of known task vectors. This section details our methodology for dictionary construction, target model creation, and decomposition algorithms.

3.1 Problem Formulation

Given a merged model with parameters θ_{merged} and base model parameters θ_{base}, we define the target task vector as:

$$\mathbf{v}_{\text{target}} = \theta_{\text{merged}} - \theta_{\text{base}} \tag{1}$$

Our goal is to decompose $\mathbf{v}_{\text{target}}$ as a sparse, non-negative combination of dictionary vectors:

$$\mathbf{v}_{\text{target}} \approx \sum_{i=1}^{K} \alpha_i \mathbf{d}_i \tag{2}$$

where $\mathbf{d}_i$ are dictionary task vectors, $\alpha_i \geq 0$ are coefficients, and K is the dictionary size.

The non-negativity constraint reflects our assumption that merged models combine positive contributions from constituent tasks rather than subtracting capabilities. Although this may not be universally applicable (especially for some merging methods), it simplifies the decomposition problem and aligns with most practical merging scenarios.

3.2 Dictionary Construction

We compose 15 distinct task vectors using LoRA fine-tuning [Hu et al., 2021] on specialized datasets:

Task Selection: We chose tasks spanning diverse domains to create a representative dictionary:

- Mathematics: Arithmetic and math problem solving of different complexity. Datasets: OpenThoughts-Math [1], OrcaMath [Mitra et al., 2024], GSM8K [Cobbe et al., 2021].
- Question Answering: Factual knowledge retrieval. Datasets: SQuAD [Rajpurkar et al., 2016], MS MARCO [Nguyen et al., 2016].
- Summarization: Text compression and key point extraction. Datasets: XSum [Narayan et al., 2018], ArXiv summarization [Cohan et al., 2018].
- General instructions: Diverse user instructions. Datasets: Alpaca [Taori et al., 2023]

[1] `https://huggingface.co/datasets/open-r1/OpenThoughts-114k-math`

- Python Coding: Code generation and debugging tasks. Datasets: Python code instructions [2] [3], Annotated Python code from Github [4], Synthetic Python QA [5]
- Other: Some other narrow tasks. Datasets: IMDB [Maas et al., 2011], Wiki style transfer [Brüel-Gabrielsson et al., 2024], Latin-to-English [6]

We sample 1500 examples from the train split of each dataset.

Fine-tuning Procedure: Each task uses LoRA with rank=32, α=32, learning rate=2e-4, trained for 3 epochs on Qwen2.5-7B-Instruct. We trained adapters only for Q, K, V, O modules. This configuration balances adaptation effectiveness with computational efficiency. The code for training is available in supplementary materials.

Vector Extraction: Task vectors are computed as the difference between fine-tuned and base model parameters, following the task arithmetic framework. For that LoRA adapters are translated to the model's parameters space by merging with the base model.

Dictionary tasks: We select 8 tasks as our dictionary tasks: Alpaca, GSM8K, XSum, MS MARCO, OpenThoughts-Math, Annotated Python code from Github, Python code instructions, Latin-to-English.

3.3 Target Model Creation and Categorization

Model categories: We create 72 merged models across three categories to evaluate decomposition under different conditions:

- Known Models (24 models): Composed entirely of 2-5 randomly selected dictionary tasks using various merging methods. These represent the optimal decomposition scenario with complete dictionary coverage.
- Mixed Models (24 models): Combine 1-3 dictionary tasks with 1-2 tasks not in the dictionary. These simulate realistic scenarios where models contain both known and unknown capabilities.
- Unknown Models (24 models): Composed entirely of tasks not in the dictionary. These represent the worst-case scenario for dictionary-based decomposition.

Merging Methods: We employ four techniques:

- Task Arithmetic: Simple parameter averaging [Ilharco et al., 2022]
- TIES: Magnitude-based conflict resolution [Yadav et al., 2023]
- DARE: Drop-and-rescale merging [Yu et al., 2023]
- Linear: Weighted linear combination

Merging was done with the `mergekit` framework [Goddard et al., 2024]. Merging weights are always uniform and sum up to 1. The exact weight depends on the number of merged tasks. For instance for 5 tasks it is 0.2.

3.4 Dimensionality Reduction

Operating on full 7B parameter models is computationally prohibitive. We reduce dimensionality through importance-based parameter selection:

1. Compute magnitude aggregation across all dictionary vectors: $M = \max_i |\mathbf{d}_i|$
2. Select top-k parameters per model layer and module based on M. k is 1M and is uniformly distributed between layers and modules. There are 112 layer and module combinations, so if k is 1M, then we get 8928 selected parameters per combination.

[2] `https://huggingface.co/datasets/iamtarun/python_code_instructions_18k_alpaca`
[3] `https://huggingface.co/datasets/Arjun-G-Ravi/Python-codes`
[4] `https://huggingface.co/datasets/Nan-Do/code-search-net-python`
[5] `https://huggingface.co/datasets/bunyaminergen/Stable-Code-Python-SFT`
[6] `https://huggingface.co/datasets/grosenthal/latin_english_translation`

3. Apply binary masking to retain only selected parameters
4. Compress all task vectors and targets to this reduced space

This procedure reduces each task vector from 800M (since we compare only Q, K, V, O modules) to 1M parameters while preserving the most significant weight changes. The reduction maintains task vector structure while enabling tractable decomposition.

3.5 Decomposition Algorithms

We evaluate six decomposition algorithms representing different mathematical approaches:

Non-negative Least Squares (NNLS): Solves the constrained optimization problem:

$$\min_{\boldsymbol{\alpha} \geq 0} \|\mathbf{v}_{\text{target}} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 \tag{3}$$

where $\mathbf{D}$ is the dictionary matrix and $\boldsymbol{\alpha}$ are coefficients.

Orthogonal Matching Pursuit (OMP): Greedily selects dictionary atoms that best explain the residual, providing inherently sparse solutions.

Lasso Regression: L1-regularized regression promoting sparsity:

$$\min_{\boldsymbol{\alpha}} \|\mathbf{v}_{\text{target}} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_1 \tag{4}$$

Ridge Regression: L2-regularized regression for stable solutions:

$$\min_{\boldsymbol{\alpha}} \|\mathbf{v}_{\text{target}} - \mathbf{D}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_2^2 \tag{5}$$

Elastic Net: Combines L1 and L2 penalties balancing sparsity and stability.

Dot Product Similarity: Computes correlation coefficients with dot product and applies thresholding for component selection.

3.6 Evaluation Metrics

We assess decomposition quality through multiple metrics:

Reconstruction Error: Measures parameter-space fidelity using:

$$\text{Error} = 1 - \max(0, \cos(\mathbf{v}_{\text{target}}, \mathbf{v}_{\text{recon}}))^2 \tag{6}$$

where $\cos$ denotes cosine similarity. This choice reflects the fact that in task-vector merges the overall magnitude is often arbitrary (e.g., due to adapter scales or training schedules), while the direction (i.e., the relative coefficients) is what encodes capability composition.

Component Precision/Recall: For known and mixed models, we evaluate binary component detection:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{7}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{8}$$

Sparsity: Average number of non-zero components per decomposition.

Perfect Match Rate: Percentage of decompositions with exactly correct component identification.

4 Results

Our experimental framework evaluates decomposition algorithms across 9072 total runs (72 target models, decomposition methods with different parameters and seeds), providing comprehensive statistical analysis.

4.1 Experimental Setup

Model Configuration: Qwen2.5-7B-Instruct serves as the base model, with LoRA adaptations using rank=32, α=32. All experiments run on consumer-grade hardware (L40S, 45GB VRAM).

Hyperparameter Optimization: Each algorithm undergoes grid search over key parameters:

- NNLS: No hyperparameters (analytical solution)
- OMP: Number of atoms $\in \{1, 2, 3, 4, 5, 6, 7, 8\}$
- Lasso: Regularization $\lambda \in \{10^{-8}, 10^{-7}, 10^{-6}\}$
- Ridge: Regularization $\lambda \in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}\}$
- Elastic Net: L1 ratio $\in \{0.1, 0.5, 0.9\}$, $\lambda \in \{10^{-8}, 10^{-6}, 10^{-4}\}$

Statistical Rigor: All decomposition experiments use 3 random seeds. Results report means and standard deviations across runs.

4.2 Primary Results

Table 1 presents comprehensive performance metrics across all algorithms and model categories.

Table 1: Overall performance across all decomposition algorithms. Sample standard deviation reported across all experimental runs.

Algorithm	Reconstruction Error	Precision	Sparsity
NNLS	**0.45 ± 0.21**	0.44 ± 0.44	6.1 ± 1.7
OMP	**0.45 ± 0.21**	0.39 ± 0.40	6.7 ± 1.6
Ridge	**0.45 ± 0.21**	0.26 ± 0.24	8.0 ± 0.0
Elastic Net	0.46 ± 0.21	0.32 ± 0.32	7.3 ± 1.01
Dot Product	0.49 ± 0.20	0.26 ± 0.24	7.6 ± 0.5
Lasso	0.74 ± 0.26	**0.53 ± 0.50**	0.9 ± 1.0

Key Findings:

- **NNLS, OMP, and Ridge achieve optimal reconstruction performance** with errors of 0.45, significantly outperforming other methods. Since we aggregate between all model categories, the error this high is expected.
- **Lasso achieves highest precision** (52.78%) but suffers from poor reconstruction accuracy (74.10% error)
- **Sparsity varies significantly** from 3.2 (Lasso) to 7.2 (Ridge) components on average

4.3 Category-Specific Performance

Table 2 breaks down performance by model category, revealing dramatic differences in decomposition success.

Table 2: Performance by model category (NNLS algorithm)

Metric	Known Models	Mixed Models	Unknown Models
Reconstruction Error	0.24 ± 0.13	0.43 ± 0.13	0.68 ± 0.09
Precision	1.00 ± 0.00	0.33 ± 0.25	0.00 ± 0.00
Recall	1.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
Perfect Match Rate	1.00 ± 0.00	0.04 ± 0.20	0.00 ± 0.00

Key findings:

- **Known models enable excellent decomposition** perfect precision and recall
- **Mixed models maintain perfect recall** (100%) but suffer precision degradation (33%)

Table 3: Reconstruction error for "known" models composed with different merging methods

Algorithm	Task Arithmetic	Linear	TIES	DARE
NNLS	0.09 ± 0.01	0.17 ± 0.06	0.28 ± 0.05	0.43 ± 0.03
OMP	0.09 ± 0.01	0.17 ± 0.06	0.28 ± 0.05	0.43 ± 0.03
Ridge	0.09 ± 0.01	0.17 ± 0.06	0.28 ± 0.05	0.43 ± 0.03
Elastic Net	0.10 ± 0.01	0.18 ± 0.06	0.29 ± 0.05	0.43 ± 0.03
Dot Product	0.15 ± 0.02	0.23 ± 0.05	0.35 ± 0.06	0.46 ± 0.03
Lasso	0.51 ± 0.25	0.56 ± 0.23	0.36 ± 0.08	0.65 ± 0.07

- **Unknown models show zero performance**, confirming the dictionary-dependence of our approach

4.4 Merging Method Analysis

Table 3 illustrates reconstruction performance across different merging methods.

Task Arithmetic emerges as the optimal choice for decomposable merging, achieving the lowest reconstruction errors across all algorithms. DARE consistently performs worst, suggesting that drop-and-rescale operations disrupt the linear structure assumed by our decomposition framework.

Several methods have identical numbers, which is expected since they produce similar decompositions.

The performance ranking (Task Arithmetic > Linear > TIES > DARE) aligns with the mathematical assumptions underlying sparse coding. Methods that preserve linear combinations in parameter space enable more accurate decomposition.

5 Limitations

While our work establishes the feasibility of model decomposition via sparse coding, several important limitations constrain its current applicability:

Behavioral Validation Gap: Our primary limitation lies in the focus on parameter-space reconstruction without systematic behavioral validation. While parameter fidelity represents a necessary condition for functional preservation, it does not guarantee that decomposed components retain their original task performance. The relationship between parameter similarity and functional equivalence remains an open research question that requires empirical validation through task-specific evaluation.

Dictionary Dependence: Our approach requires comprehensive dictionary coverage of target model capabilities. The dramatic performance difference between known models and unknown models demonstrates this fundamental limitation. Practical applications must invest significant effort in dictionary construction and maintenance.

Non-Negativity Constraints: The assumption that model capabilities combine through positive coefficients may not hold universally. Some merging scenarios involve capability subtraction or interference effects that require negative contributions. Our current framework cannot handle these cases without substantial modification.

Linear Composition Assumption: The sparse coding formulation assumes that model capabilities combine linearly in parameter space. Non-linear interactions between tasks may not be captured accurately, potentially limiting decomposition fidelity for complex multi-task models with significant capability overlap or interference.

Scalability Constraints: Our experiments focus on a 7B parameter model with rank-32 LoRA adaptations and an 8-task dictionary. Scaling to larger models, full fine-tuning scenarios, or extensive task dictionaries may require algorithmic innovations and computational resources beyond current capabilities.

Merging Method Sensitivity: Decomposition performance varies significantly across merging methods, with Task Arithmetic showing optimal characteristics while DARE performs poorly. This sensitivity limits the framework's applicability to existing merged models that may use suboptimal merging approaches.

6 Ethics and Broader Impact

UNMERGE introduces capabilities for analyzing and attributing model compositions, offering substantial benefits for the AI research community and broader applications. The technique enhances model transparency and interpretability, particularly valuable for safety-critical applications where understanding component contributions is essential. It provides robust intellectual property protection through verifiable component attribution, enables scientific reproducibility by allowing researchers to verify model compositions, and offers powerful debugging capabilities for identifying unwanted behaviors in merged models.

However, these analytical capabilities also introduce potential risks that warrant careful consideration. The technique could potentially enable unauthorized analysis of proprietary model compositions, allowing competitors or malicious actors to gain insights into carefully guarded intellectual property. There's also the concern that UNMERGE may facilitate reverse engineering of specialized capabilities, potentially undermining the competitive advantages that organizations have developed through significant investment in model development. Furthermore, the technology raises complex questions about model ownership and attribution in collaborative settings, where multiple parties may have contributed to a model's development.

To address these concerns, several mitigation strategies should be implemented alongside continued development of the technology. Establishing responsible disclosure frameworks for capability attribution research will help ensure that discoveries are shared appropriately while protecting legitimate interests. The development of privacy-preserving decomposition techniques could allow for the benefits of model analysis while maintaining necessary confidentiality. Creating best practices for ethical model analysis will provide guidance for researchers and practitioners in applying these tools responsibly. Ultimately, the benefits of improved model transparency and verification capabilities outweigh the associated risks when the technology is developed and deployed through responsible research practices that balance innovation with appropriate safeguards for legitimate stakeholder interests.

7 Conclusion

We introduce UNMERGE, a verifiable framework for model capability attribution via sparse coding that enables the decomposition of merged models into their constituent task-specific components. Through comprehensive evaluation across 72 merged models and 6 decomposition algorithms, we demonstrate that NNLS achieves exceptional performance with reconstruction errors of 0.45 and perfect precision/recall for known compositions.

The framework provides a foundation for future work in neural network interpretability while opening new directions for model verification, debugging, and attribution.

The perfect performance achieved on known compositions (100% precision/recall with NNLS) demonstrates the fundamental feasibility of accurate model decomposition, while the zero performance on unknown compositions confirms the specificity of our approach. These results establish UNMERGE as a practical tool for controlled model analysis while identifying key areas for future development.

While we focus on parameter-space reconstruction as a necessary first step, we acknowledge that behavioral validation represents crucial future work. The relationship between parameter fidelity and functional performance requires systematic investigation through task-specific evaluation. Nevertheless, our parameter-space approach provides valuable foundations for scalable model analysis and hypothesis generation.

Future work should address behavioral validation through systematic task evaluation, scalability to larger models and dictionaries, extension to non-linear composition scenarios, and development of adaptive dictionary learning methods. The intersection of sparse coding and neural network interpretability represents a promising research direction with significant implications for AI safety, transparency, and scientific understanding.

8 Reproducibility Statement

We provide the code and a clear and complete pipeline in the supplementary materials. The pipeline is divided into 4 phases: training, merging, compressing, and final decomposition. All steps were manually verified and rerun from scratch. The code was linted with `black` and `flake8`. Hardware requirements: GPU with 45 GB of VRAM, at least 1 TB of disk space to store merged models.

References

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30080, 2020.

Dan Braun, Lucius Bushnaq, Stefan Heimersheim, Cody McDougall, Damjan Paleka, and Stuart Russell. Attribution-based parameter decomposition. *arXiv preprint arXiv:2501.14926*, 2025.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, et al. Towards monosemanticity: decomposing language models with dictionary learning. *Anthropic*, 2023.

Rickard Brüel-Gabrielsson, Jiacheng Zhu, Onkar Bhardwaj, Leshem Choshen, Kristjan Greenewald, Mikhail Yurochkin, and Justin Solomon. Compress then serve: Serving thousands of lora adapters with little overhead, 2024. URL `https://arxiv.org/abs/2407.00066`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL `https://aclanthology.org/N18-2097`.

Arthur Conmy, Augustine N Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*, 2023.

Michael Elad. *Sparse and redundant representations: from theory to applications in signal and image processing*. Springer Science & Business Media, 2010.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. A mathematical framework for transformer circuits. *Anthropic*, 2021.

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, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 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`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

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

Edward Kim, Connor Onweller, Andrew O'Brien, Yogesh Balaji, Sylvestre-Alvise Rebuffi, and Alan Yuille. The interpretable dictionary in sparse coding. *arXiv preprint arXiv:2011.11805*, 2020.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*, 2022.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math, 2024.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745, 2018.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016. URL `http://arxiv.org/abs/1611.09268`.

Bruno A Olshausen and David J Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

Aaquib Syed, Can Rager, and Arthur Conmy. Attribution patching: Activation patching at industrial scale. *arXiv preprint arXiv:2310.10348*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, et al. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Anthropic*, 2024.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*, 2023.

Le Yu, Bowen Xiang, Haiyang Ding, Jingren Hu, Chengming Yuan, Yatao Yang, Qianyu Chen, Yonghua Han, and Jingren Yuan. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*, 2023.

Ningyu Zhang, Yunzhi Yao, Bozhong Tian, Peng Wang, Shumin Deng, Mengru Wang, Zekun Xi, Shengyu Mao, Jintian Zhang, Yuansheng Ni, et al. A comprehensive study of knowledge editing for large language models. *arXiv preprint arXiv:2401.01286*, 2024.

A Remaining decomposition results

Table 4 illustrates the reconstruction performance in different merging methods.

Table 4: Reconstruction error for different algorithms and model categories.

Algorithm	Known	Mixed	Unknown
NNLS	0.24 ± 0.13	0.43 ± 0.13	0.68 ± 0.09
OMP	0.24 ± 0.13	0.43 ± 0.13	0.68 ± 0.09
Ridge	0.24 ± 0.13	0.43 ± 0.13	0.68 ± 0.09
Elastic Net	0.25 ± 0.13	0.44 ± 0.13	0.68 ± 0.09
Dot product	0.30 ± 0.13	0.49 ± 0.13	0.69 ± 0.08
Lasso	0.52 ± 0.20	0.71 ± 0.22	0.99± 0.03

Table 5 illustrates the precision and recall of different algorithms and merging methods across all models from the "known" category.

Table 5: Precision / Recall for different algorithms and merging methods. Models are only from the "known" category.

Algorithm	DARE	TIES	Linear	Task Arithmetic
NNLS	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00	1.00 / 1.00
OMP	1.00 / 1.00	0.56 / 1.00	1.00 / 1.00	1.00 / 1.00
Ridge	0.54 / 1.00	0.54 / 1.00	0.54 / 1.00	0.54 / 1.00
Elastic Net	0.62 / 1.00	0.87 / 1.00	0.62 / 1.00	0.62 / 1.00
Dot product	0.55 / 1.00	0.54 / 1.00	0.55 / 1.00	0.55 / 1.00
Lasso	1.00 / 0.40	1.00 / 0.67	0.83 / 0.36	0.83 / 0.36

Table 6 illustrates the sparsity (number of active coefficients) of different algorithms and merging methods across all models of the category "known".

Table 6: Sparsity for different algorithms and merging methods. Models are only from the "known" category.

Algorithm	DARE	TIES	Linear	Task Arithmetic
NNLS	4.33	4.33	4.33	4.33
OMP	4.33	7.67	4.33	4.33
Ridge	8.00	8.00	8.00	8.00
Elastic Net	7.00	5.00	7.00	7.00
Dot product	7.83	8.00	7.83	7.83
Lasso	1.67	2.83	1.50	1.50

Agents4Science AI Involvement Checklist

1. **Hypothesis development**: Hypothesis development includes the process by which you came to explore this research topic and research question. This can involve the background research performed by either researchers or by AI. This can also involve whether the idea was proposed by researchers or by AI.

   Answer: **[D]**

   Explanation: The initial idea and the whole research proposal was generated by the LLM system. However, we slightly modified the proposal during the project execution.

2. **Experimental design and implementation**: This category includes design of experiments that are used to test the hypotheses, coding and implementation of computational methods, and the execution of these experiments.

   Answer: **[C]**

   Explanation: The high-level design of the experiments was heavily redacted by humans. Implementation was done mostly by LLM, with us checkpointing the progress.

3. **Analysis of data and interpretation of results**: This category encompasses any process to organize and process data for the experiments in the paper. It also includes interpretations of the results of the study.

   Answer: **[C]**

   Explanation: Humans mostly did not redact the interpretations. Sometimes we had to point out some obvious things in data.

4. **Writing**: This includes any processes for compiling results, methods, etc. into the final paper form. This can involve not only writing the main text but also figure-making, improving the layout of the manuscript, and forming the narrative.

   Answer: **[C]**

   Explanation: The writing was done mostly using the LLM system. We provided the LaTeX template and fixed some inconsistencies with real experimental data. Also we provided citations for datasets.

5. **Observed AI Limitations**:

   Description: Many hallucinations, invalidating already correct results, inability to follow simple commands. Overall, we do not feel that full autonomy of the LLM agents in writing scientific papers is now possible.

   However, it seems it is already close to perfect in literature analysis and idea generation.

   One of the major problems was hiding failures. For instance, when the system was producing merged models, it used a key "adapter" in a YAML config for mergekit. There is no such key. So the system produced 72 identical models and never actually checked it. It came up later when the diff vectors on the later stage appeared to be zeros.

Agents4Science Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: They really do. Everything was checked by human authors.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: There is a separate section for that.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper is almost purely emprical.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Yes, we also have the code verified and fixed by humans.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, the code will be in the supplemental materials.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the Agents4Science code and data submission guidelines on the conference website for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes] ,

   Justification: Mostly yes, some tiny details were omitted because of the page limit.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, we report standard deviation in all tables.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, or overall run with given experimental conditions).

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, it was rented L40S.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the Agents4Science Code of Ethics (see conference website)?

   Answer: [Yes]

   Justification: Yes, see the Ethics section.

   Guidelines:

   - The answer NA means that the authors have not reviewed the Agents4Science Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Yes, see the Ethics section.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations, privacy considerations, and security considerations.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies.