Arcee's MergeKit: A Toolkit for Merging Large Language Models

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers,
Vlad Karpukhin, Brian Benedict, Mark McQuade, Jacob Solawetz
Arcee, Florida, USA
{charles, shamane, malikeh, luke, vlad, benedict, mark, jacob}@arcee.ai

Abstract

The rapid growth of open-source language models provides the opportunity to merge model checkpoints, combining their parameters to improve performance and versatility. Advances in transfer learning have led to numerous task-specific models, which model merging can integrate into powerful multitask models without additional training. MergeKit is an open-source library designed to support this process with an efficient and extensible framework suitable for any hardware. It has facilitated the merging of thousands of models, contributing to some of the world's most powerful open-source model checkpoints. The library is accessible at: https://github.com/arcee-ai/mergekit.

1 Introduction

Over the past year, open-source Large Language Models (LLMs) have rapidly developed and are accessible via the Hugging Face model hub (Wolf et al., 2019). These models, trained on up to trillions of tokens, typically range from 1-70+ billion parameters (Minaee et al., 2024; Zhang et al., 2024). Open-source checkpoints include pretrained and instruction-tuned models across domains like coding (Roziere et al., 2023) and medical applications (Wu et al., 2023). Fine-tuning separate models for each task presents challenges: storing and deploying each model separately and the inability of independently trained models to leverage insights from related tasks (Sanh et al., 2021; Ramé et al., 2023; Yadav et al., 2024; Yu et al., 2023).

Training these models from scratch requires substantial investment. Further fine-tuning can lead to catastrophic forgetting (De Lange et al., 2021), degrading their general capabilities and performances across tasks (Cheng et al., 2023; Wu et al., 2024). Aligning models to respond favorably requires extensive human preference data, often unattainable for most teams (Wang et al., 2023; Rafailov et al., 2024). This raises the question of leveraging existing pretrained checkpoints. Model merging has emerged as a transformative strategy, combining parameters from multiple models into a single one, enabling multitask and continual learning while reducing catastrophic forgetting (Siriwardhana et al., 2024).

In this paper, we introduce MergeKit[1], a centralized library for executing community-formulated merging strategies, compatible with memory-constrained CPUs and accelerated GPUs. Our main contributions are: (1) an overview of current model merging research to date and (2) a presentation of MergeKit's key objectives, architectural decisions, and development principles to establish an extensible foundation for the future efforts of the model merging community.

2 Background & Related Work

2.1 The Concept of Model Merging

Model merging (Ainsworth et al., 2022), a recent focus in research, integrates two or more pretrained models into a unified model that retains their strengths. This concept builds on weight averaging (Utans, 1996) and mode connectivity (Garipov et al., 2018). Techniques often leverage Linear Mode Connectivity (LMC) (Entezari et al., 2021) for models fine-tuned from a common pretrained model (Nagarajan and Kolter, 2019; Neyshabur et al., 2021). Other works employ permutation equivariance and apply transformations to model weights, aligning them in the loss landscape (Ainsworth et al., 2022; Stoica et al., 2023; Verma and Elbayad, 2024).

2.2 Different Types of Model Merging

In developing our toolkit, as shown in Figure 1, we categorize existing and anticipated model merging techniques. This classification enhances understanding by focusing on two critical aspects:

[1] https://github.com/arcee-ai/mergekit

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 477–485
November 12-16, 2024 ©2024 Association for Computational Linguistics

weight initializations and the architectural configurations of various checkpoints.

2.2.1 Merging Models with Both Identical Architectures and Initializations

This section explores model merging techniques using LMC (Nagarajan and Kolter, 2019) to derive a final merged model through linear interpolation. A key requirement is that the models must have identical architectures and initializations.

The simplest method, built upon the results of weight averaging literature (Utans, 1996; Smith and Gashler, 2017; Garipov et al., 2018; Izmailov et al., 2018) and the Model Soups (Wortsman et al., 2022) approach, is linear averaging of weights. This technique relies on linear mode connectivity and is the foundation of most others.

Task Arithmetic (Ilharco et al., 2022) expands upon this approach by introducing the concept of task vectors, showing that performing arithmetic on the differences between fine-tuned models and a common base model is both useful and semantically meaningful.

Trim, Elect Sign & Merge (TIES merging) (Yadav et al., 2023), Model Breadcrumbs (Davari and Belilovsky, 2023), and Drop And REscale (DARE) (Yu et al., 2023) further introduce methods for sparsifying and combining these task vectors that enable larger numbers of models to be combined into one without degrading capabilities.

The use of the Spherical Linear intERPolation (SLERP) technique (Shoemake, 1985) to interpolate between model checkpoints is an extension of simple weight averaging. Its success shows that there is often a spherical path with a lower loss barrier than a direct linear interpolation. SLERP[2] leverages the geometric and rotational properties within the models' vector space, ensuring a blend that more accurately embodies the characteristics of both parent models.

Other approaches introduce weighting factors defined in terms of model activations that must be computed with training data. Matena and Raffel (2022) explore the use of the Fisher information matrix. Jin et al. (2022) introduce the Regression Mean (RegMean) method, which allows merges to produce optimal weights with respect to $L2$ distance to model predictions while keeping training data private.

MergeKit introduces two novel methods for building larger models without performing any parameter-space combination. Referred to online as 'FrankenMerging', the passthrough method in MergeKit allows the piecewise combination of layers from multiple models into a new model of unusual size. This technique is behind the popular model Goliath-120b[3], and is the first step of the Depth Up-Scaling technique of (Kim et al., 2023) used for SOLAR-10.7B[4] and Yi-9B[5]. Similarly referred to as Franken Mixture of Experts ('FrankenMoE'), the `mergekit-moe` script allows building a Mixture of Experts (MoE) model from multiple dense models using either a prompt based hidden state heuristic for semantic routing or randomly initialized gates for sparse up-cycling as in (Komatsuzaki et al., 2023).

Evolutionary Model Merging (Akiba et al., 2024) is a novel method that automates the creation of foundation models by leveraging diverse open-source models without extensive additional training data. This approach optimizes combining models from different domains in both parameter space (PS) and data flow space (DFS). PS optimization integrates the weights of multiple models, while DFS preserves original weights and optimizes the inference path. Models created using evolutionary model merging, such as EvoLLM-JP (Akiba et al., 2024), demonstrate state-of-the-art performance, highlighting the efficiency and generalizability of this technique.

2.2.2 Merging Models with Identical Architectures and Different Initializations

This section explores advanced merging methods beyond combining checkpoints with identical initializations. Previous research shows that simple linear model combination is insufficient for different initializations (Ainsworth et al., 2022). Methods leveraging permutation symmetry of checkpoints include Git-Rebasin (Ainsworth et al., 2022) and Optimizing Mode Connectivity via Neuron Alignment (Tatro et al., 2020), which permute weights of independently trained models to reduce interpolation barriers. Optimal Transport Fusion (OTFusion) (Singh and Jaggi, 2020) operates similarly but computes a soft mapping between neurons using Optimal Transport. These methods assign correspondences between model neurons

[2] https://github.com/Digitous/LLM-SLERP-Merge

[3] alpindale/goliath-120b

[4] upstage/SOLAR-10.7B-v1.0

[5] 01-ai/Yi-1.5-9B

and perform simple interpolation in transformed weight space. Recent work (Imfeld et al., 2023; Verma and Elbayad, 2024) extends these methods to Transformer-based models. (Jordan et al., 2022) addresses variance collapse in interpolated networks with a rescaling step, reducing loss barriers between permuted models. ZipIt (Stoica et al., 2023) expands the scope by merging models with similar architectures trained on distinct tasks. This method correlates features within and across models, and can also allow partial merging to create a multi-head model. ZipIt preserves and integrates knowledge from different domains into a unified model without additional training.

These techniques do not yet share the wide adoption and success of merging models trained from a common initialization, but present a promising future research direction for the field of merging.

2.2.3 Fusing Models with Different Architectures

While not strictly model merging, Composition to Augment Language Models (CALM) (Bansal et al., 2024) and knowledge fusion approaches like FUSELLM (Wan et al., 2024) advance the fusion of models with diverse architectures. CALM uses cross-attention mechanisms to blend representations from different models, leveraging their combined strengths across varied neural network structures. FUSELLM focuses on aligning and fusing the probabilistic distributions of source LLMs to amplify their collective knowledge and advantages. Unlike previous methods, these approaches require additional training of the models.

2.3 Practical Use Cases of Model Merging

Model merging significantly impacts machine learning models on platforms like Hugging Face (Wolf et al., 2019). Merged models, such as BioMistral (Labrak et al., 2024), Aloe (Gururajan et al., 2024), Llama-3-SEC (Siriwardhana et al., 2024), Prometheus 2 (Kim et al., 2024), and OpenPipe's Mistral 7B Fine-Tune Optimized (Corbitt, 2023), demonstrate competitive performance in specialized domains and fine-tuning applications. Wei et al. (2024) highlight merging's success in enhancing hallucination detection performance. Tao et al. (2024) show effectiveness of model merging to develop task-solving LLMs for low-resource languages. The success of merged models underscores their value in continuous and multitask learning, enabling the creation of versatile models that excel at multiple tasks or adapt to new domains without retraining from scratch. This approach maximizes existing resources and fosters innovative solutions for complex problems.

3 Library Design: Key Design Principles

MergeKit has been thoughtfully engineered to facilitate the straightforward application of both current and forthcoming model merging techniques. Our repository includes detailed tutorials and IPython[6] notebooks to guide users through the process of utilizing MergeKit effectively. This section is dedicated to outlining the fundamental design principles underpinning the library, with the aim of assisting the open-source community in adopting our toolkit and incorporating new techniques.

3.1 User-Centric Design: Intuitive Interface and YAML Configuration Control

The primary interface for MergeKit is through YAML configuration files that allow users of all skill levels to define complex merge operations without the need for coding experience. This approach both democratizes the use of MergeKit and fosters community engagement by making merge recipes easily repeatable, shareable, and remixable.

A YAML[7] configuration file defines the merge method, input models, and any parameters necessary for the merging algorithm selected. Parameters can be set globally or targeted to specific model components, and can be specified as constant scalar values or as layer-varying interpolated gradients. These different levels of granularity offer an easy introduction for simple merges while allowing power users to define truly complex operations.

3.2 Modularity: Plug-and-Play Components

MergeKit is designed with composability and reusability as guiding principles. Merge methods are designed to be interchangeable and easy-to-add. Components are structured such that they can be added, removed, or interchanged to allow customization and experimentation. Wherever possible, components are designed to be useful standalone for external use. For instance, MergeKit's lazy tensor loading functionality is a core component of the toolkit, but is also simple and convenient

[6]https://github.com/arcee-ai/mergekit/blob/main/notebook.ipynb

[7]https://github.com/arcee-ai/mergekit/tree/main/examples

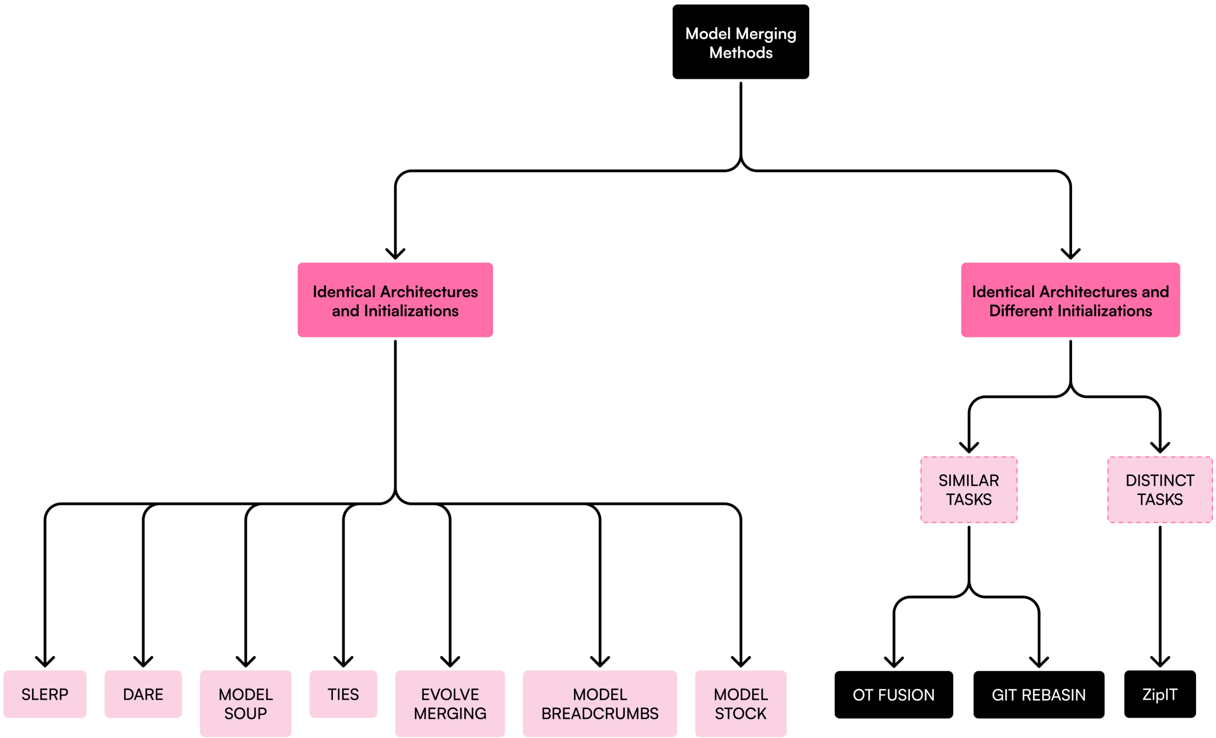

Figure 1: Classification of model merging methods. We currently support the model merging methods outlined on the left, and we are actively working to incorporate additional merging techniques such as ZipIt, OT Fusion, and Git Rebasin.

to pull into one-off scripts. Figure 2 highlights some important points of extensibility and reusable components. MergeKit is tightly integrated with the Hugging Face Transformers library (Wolf et al., 2019) and its model hub.

3.3 Scalability: Efficiency and Performance Optimization

MergeKit is designed specifically to address the challenge of merging large pretrained language models with a diverse range of available computational resources. At the heart of its efficiency is an out-of-core approach to model merging. By loading only the tensors necessary for each individual operation into working memory, MergeKit can scale from a high-end research cluster all the way down to a personal laptop with no GPU and limited Random-Access Memory (RAM). We use Directed Acyclic Graph (DAG) approach to optimize the merging process for large models. The DAG structure allows for efficient computation by organizing operations in a way that minimizes redundancy and resource usage. This method is particularly advantageous in handling model merging on resource-constrained environments.

3.3.1 Computational Graph Scheduling

MergeKit internally represents a merge as a directed acyclic graph of operations, or Task instances. This representation is used to schedule the execution of tasks such that the working set needed at any given time is minimized. Execution of the graph also implicitly handles eviction of intermediate values that are no longer needed. This infrastructure allows developers to build new merge methods that benefit from MergeKit's memory efficiency and hardware scalability with little to no extra effort.

3.4 Mergekit Graphical User Interface (GUI)

We developed MergeKit-GUI[8], a user-friendly interface hosted on Hugging Face running on A100 GPU, designed to simplify the model merging process. With this GUI, users can easily upload configuration files, select from an array of different merging techniques, and execute merges with a few clicks. A demonstration of MergeKit-GUI is shown in Figure 3.

The workflow is straightforward: users start by uploading a YAML configuration file—either by providing their own or by choosing from a variety of pre-configured examples available on the interface. After the configuration file is set, users input their Hugging Face token for authentication and specify the repository name where the final merged model will be stored.

[8]https://huggingface.co/spaces/arcee-ai/mergekit-gui

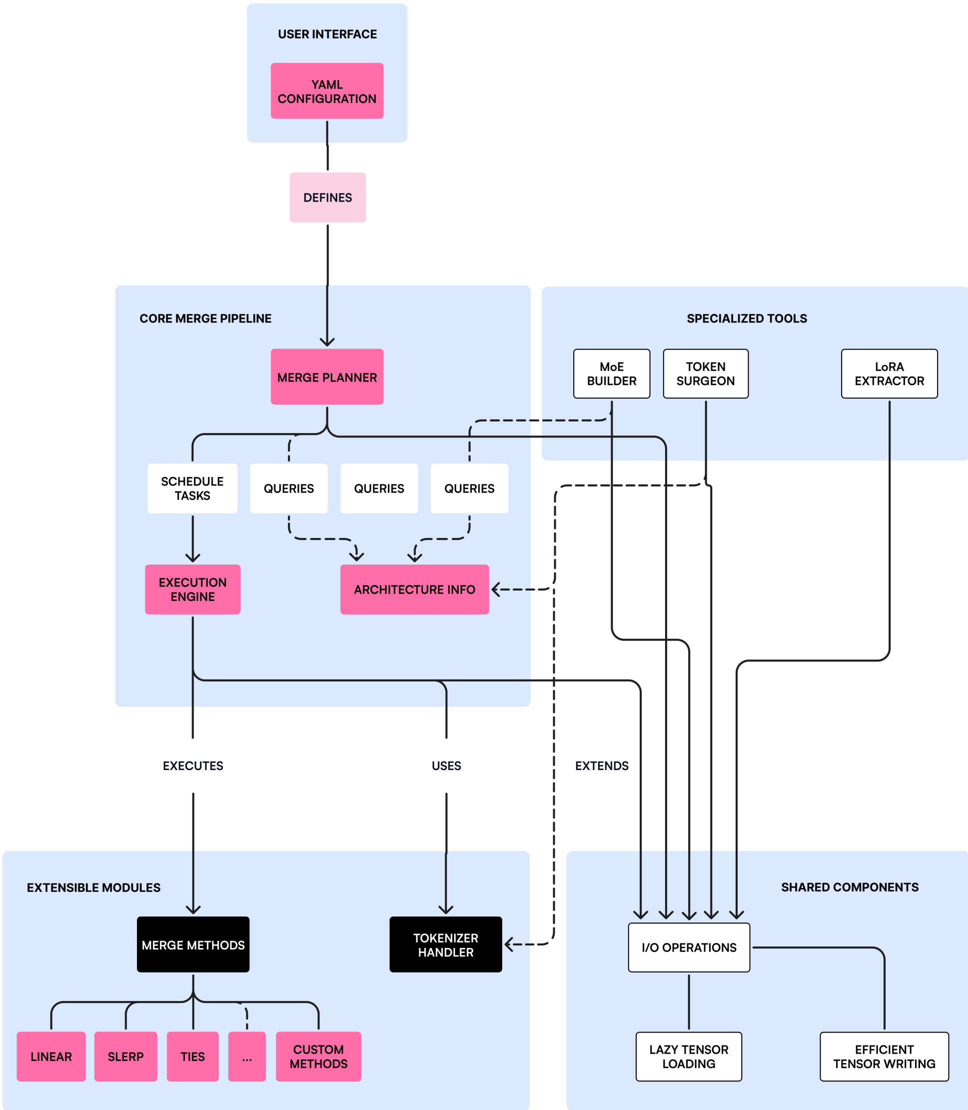

Figure 2: MergeKit Architecture. The diagram depicts the software architecture of MergeKit and highlights the points meant to be extended and components that are easily reusable in other scripts.

Once all parameters are configured, users can click on the 'Merge' button to initiate the process. The terminal output displays real-time logs, allowing users to monitor the merging process step-by-step. Upon successful completion, the following confirmation message appears: `Process completed successfully. Model successfully uploaded to HF: <REPOSITORY_NAME>`.

4 Extensibility of MergeKit

Given the rapid success of model merging techniques and the anticipated development of innovative methods, we invite the community to develop novel merging strategies and enhancements, thereby contributing to the growth and refinement of MergeKit. This section aims to provide a streamlined guide on integrating new merging methods into MergeKit, utilizing existing functionalities where applicable to facilitate the process.

To incorporate a new merging method into MergeKit, contributors should familiarize themselves with several key Python modules within the repository:

- `merge_methods/base.py`: Defines the interface that new merge methods must implement.
- `graph.py`: Handles scheduling, execution, and data management throughout the merge process. This is the heart of MergeKit's performance and resource efficiency. Understanding this module is important to ensure that intermediate results and data movement across devices is handled efficiently.

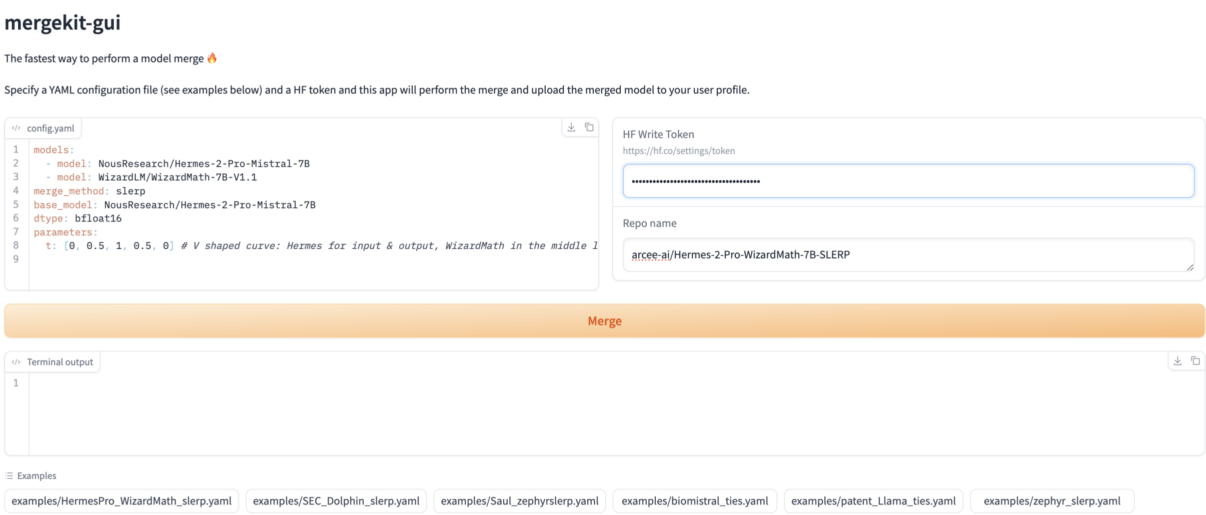

Figure 3: Demo of MergeKit-GUI.

- `plan.py`: Responsible for creating the computational graph for a merge. If a new merging strategy has different steps involved or inputs required in combining multiple models, they should be accommodated here.

- `architecture.py`: This module deals with the structures of different checkpoints. Most model architectures are defined using simple JSON files. To add support for odd or unique architectures you may need to modify this file.

4.1 Practical Example: Applying Model Merging in Medical Domain

As illustrated in Table 1, we experimented with a range of merging techniques available in MergeKit, including Linear intERPolation (LERP), SLERP, TIES, and DARE-TIES, to merge Meditron-7B[9] (Chen et al., 2023) with the Llama2-7B chat model (Touvron et al., 2023). Both models are based on the Llama2-7B base model. The evaluation results are depicted in Table 1. According to the findings, all the merged models outperform the Meditron-7B model across various medical benchmarks, including the US Medical License Exam (USMLE) (Jin et al., 2021), Medical Multiple-Choice Question Answering (MedMCQA) (Pal et al., 2022), and PubMed[10] Question Answering (PubMedQA) (Jin et al., 2019). Furthermore, models merged using LERP and SLERP techniques exhibit superior performance over the Llama2-7B chat model in general benchmarks. Our empirical experiments highlight the varying capabilities of merged models and provide comparative performance insights. Within the medical domain, the SLERP method appears to outperform others. However, more importantly, these experiments reveal how model merging can lead to the development of more generalized models with enhanced capabilities across diverse applications.

Recent studies emphasize the importance of merging fine-tuned models into their base models to address challenges like catastrophic forgetting and skill transfer (Alexandrov et al., 2024; Siriwardhana et al., 2024). This technique helps maintain prior knowledge while integrating new capabilities. We employed several merging techniques, each with its own hyper-parameters, such as the contribution of each pre-trained model and parameter masking in task vectors.

5 Conclusion and Future Work

In this paper, we introduce MergeKit, an innovative open-source tool for seamlessly integrating LLMs. We detail its functionalities and provide an overview of recent model merging literature from an engineering perspective. Additionally, we offer insights on incorporating new merging techniques, encouraging community contributions. MergeKit is a dynamic project, committed to continuously integrating new methodologies through collaborative efforts with the open-source community.

Ethical Considerations

As stewards of the open-source community dedicated to the advancement of LLMs, our work with MergeKit underscores a commitment to democratizing access to cutting-edge AI technologies while fostering an environment of ethical integrity and

[9]Meditron-7B checkpoint is based on Llama2-7B base model, which is extensively pretrained on a comprehensively curated medical corpus.

[10]https://pubmed.ncbi.nlm.nih.gov/

Model	Medical Benchmarks			General Benchmarks		
	USMLE	MedMCQA	PubMedQA	Arc Challenge	HellaSwag	MMLU
Llama2-7B-Chat (Touvron et al., 2023)	35.90	35.45	73.40	44.20	55.40	46.37
Meditron-7B (Chen et al., 2023)	38.40	24.07	71.40	40.20	54.50	33.06
MeditronLlama-7B-Lerp	39.10	36.65	**75.60**	46.76	58.66	**48.44**
MeditronLlama-7B-Slerp	**39.20**	**36.91**	**75.60**	**46.84**	**58.67**	47.97
MeditronLlama-7B-Dare-Ties	36.37	27.56	72.20	42.92	54.79	41.17
MeditronLlama-7B-Ties	38.73	32.27	**75.60**	45.05	58.23	45.03

Table 1: Comparison of the Llama2-7B Chat and Meditron-7B (Chen et al., 2023) models, plus their merged variants, using MergeKit techniques across medical and general benchmarks. It highlights the best-performing models in bold for each metric.

continuous improvement. By providing an open-source toolkit that enables the merging of model checkpoints, we aim to enhance the collaborative capabilities of researchers, developers, and practitioners across the globe, encouraging innovation and the sharing of knowledge. In doing so, we are acutely aware of the necessity to uphold principles of fairness, accountability, and transparency within this community. This includes the proactive identification and mitigation of biases within merged models, ensuring the ethical use of data, and maintaining the privacy and security of information. Our commitment extends beyond technological advancements, encompassing the responsibility to engage with diverse stakeholders, gather feedback, and adapt our approaches to address ethical concerns effectively. We recognize the imperative to continually evolve our practices, striving for solutions that not only push the boundaries of AI but also do so with an unwavering commitment to the improvement of society.

References

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

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. 2024. Evolutionary optimization of model merging recipes. *arXiv preprint arXiv:2403.13187*.

Anton Alexandrov, Veselin Raychev, Mark Niklas Müller, Ce Zhang, Martin Vechev, and Kristina Toutanova. 2024. Mitigating catastrophic forgetting in language transfer via model merging. *arXiv preprint arXiv:2407.08699*.

Rachit Bansal, Bidisha Samanta, Siddharth Dalmia, Nitish Gupta, Shikhar Vashishth, Sriram Ganapathy, Abhishek Bapna, Prateek Jain, and Partha Talukdar. 2024. Llm augmented llms: Expanding capabilities through composition. *arXiv preprint arXiv:2401.02412*.

Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, et al. 2023. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.16079*.

Daixuan Cheng, Shaohan Huang, and Furu Wei. 2023. Adapting large language models via reading comprehension. *arXiv preprint arXiv:2309.09530*.

Kyle Corbitt. 2023. How we built "mistral 7b fine-tune optimized," the best 7b model for fine-tuning.

MohammadReza Davari and Eugene Belilovsky. 2023. Model breadcrumbs: Scaling multi-task model merging with sparse masks. *arXiv preprint arXiv:2312.06795*.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385.

Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. 2021. The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. 2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31.

Ashwin Kumar Gururajan, Enrique Lopez-Cuena, Jordi Bayarri-Planas, Adrian Tormos, Daniel Hinjos, Pablo Bernabeu-Perez, Anna Arias-Duart, Pablo Agustin Martin-Torres, Lucia Urcelay-Ganzabal, Marta Gonzalez-Mallo, et al. 2024. Aloe: A family of fine-tuned open healthcare llms. *arXiv preprint arXiv:2405.01886*.

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

Moritz Imfeld, Jacopo Graldi, Marco Giordano, Thomas Hofmann, Sotiris Anagnostidis, and Sidak Pal Singh. 2023. Transformer fusion with optimal transport. *arXiv preprint arXiv:2310.05719*.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. 2022. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. 2022. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403*.

Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, et al. 2023. Solar 10.7 b: Scaling large language models with simple yet effective depth up-scaling. *arXiv preprint arXiv:2312.15166*.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2024. Prometheus 2: An open source language model specialized in evaluating other language models. *arXiv preprint arXiv:2405.01535*.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. 2023. Sparse upcycling: Training mixture-of-experts from dense checkpoints.

Yanis Labrak, Adrien Bazoge, Emmanuel Morin, Pierre-Antoine Gourraud, Mickael Rouvier, and Richard Dufour. 2024. Biomistral: A collection of open-source pretrained large language models for medical domains. *arXiv preprint arXiv:2402.10373*.

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

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Vaishnavh Nagarajan and J Zico Kolter. 2019. Uniform convergence may be unable to explain generalization in deep learning. *Advances in Neural Information Processing Systems*, 32.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2021. What is being transferred in transfer learning? *arXiv preprint arXiv:2008.11687*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Alexandre Ramé, Kartik Ahuja, Jianyu Zhang, Matthieu Cord, Léon Bottou, and David Lopez-Paz. 2023. Model ratatouille: Recycling diverse models for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 28656–28679. PMLR.

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

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Ken Shoemake. 1985. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254.

Sidak Pal Singh and Martin Jaggi. 2020. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055.

Shamane Siriwardhana, Mark McQuade, Thomas Gauthier, Lucas Atkins, Fernando Fernandes Neto, Luke Meyers, Anneketh Vij, Tyler Odenthal, Charles Goddard, Mary MacCarthy, et al. 2024. Domain adaptation of llama3-70b-instruct through continual pre-training and model merging: A comprehensive evaluation. *arXiv preprint arXiv:2406.14971*.

Joshua Smith and Michael Gashler. 2017. An investigation of how neural networks learn from the experiences of peers through periodic weight averaging. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 731–736. IEEE.

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

Mingxu Tao, Chen Zhang, Quzhe Huang, Tianyao Ma, Songfang Huang, Dongyan Zhao, and Yansong Feng. 2024. Unlocking the potential of model merging for low-resource languages. *arXiv preprint arXiv:2407.03994*.

Norman Tatro, Pin-Yu Chen, Payel Das, Igor Melnyk, Prasanna Sattigeri, and Rongjie Lai. 2020. Optimizing mode connectivity via neuron alignment. *Advances in Neural Information Processing Systems*, 33:15300–15311.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Joachim Utans. 1996. Weight averaging for neural networks and local resampling schemes. In *Proc. AAAI-96 Workshop on Integrating Multiple Learned Models. AAAI Press*, pages 133–138. Citeseer.

Neha Verma and Maha Elbayad. 2024. Merging text transformer models from different initializations. *arXiv preprint arXiv:2403.00986*.

Fanqi Wan, Xinting Huang, Deng Cai, Xiaojun Quan, Wei Bi, and Shuming Shi. 2024. Knowledge fusion of large language models. *arXiv preprint arXiv:2401.10491*.

Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*.

Chengcheng Wei, Ze Chen, Songtan Fang, Jiarong He, and Max Gao. 2024. Opdai at semeval-2024 task 6: Small llms can accelerate hallucination detection with weakly supervised data. *arXiv preprint arXiv:2402.12913*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR.

Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023. Pmc-llama: Further finetuning llama on medical papers. *arXiv preprint arXiv:2304.14454*.

Chengyue Wu, Yukang Gan, Yixiao Ge, Zeyu Lu, Jiahao Wang, Ye Feng, Ping Luo, and Ying Shan. 2024. Llama pro: Progressive llama with block expansion. *arXiv preprint arXiv:2401.02415*.

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

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

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2023. Language models are super mario: Absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2311.03099*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.