Collective Model Intelligence Requires Compatible Specialization

Abstract

In this work, we explore the limitations of combining models by averaging intermediate features, referred to as *model merging*, and propose a new direction for achieving collective model intelligence through what we call *compatible specialization*. Current methods for model merging, such as parameter and feature averaging, struggle to effectively combine specialized models due to representational divergence during fine-tuning. As models specialize to their individual domains, their internal feature representations become increasingly incompatible, leading to poor performance when attempting to merge them for new tasks. We analyze this phenomenon using centered kernel alignment (CKA) and show that as models specialize, the similarity in their feature space structure diminishes, hindering their capacity for collective use. To address these challenges, we investigate routing-based merging strategies, which offer more flexible methods for combining specialized models by dynamically routing across different layers. This allows us to improve on existing methods by combining features from multiple layers rather than relying on fixed, layer-wise combinations. However, we find that these approaches still face limitations when layers within models are representationally incompatible. Our findings highlight the importance of designing new approaches for model merging that operate on well-defined input and output spaces, similar to how humans communicate through language rather than intermediate neural activations.

Code:
Website:

1 Introduction

Machine learning has recently seen an explosion of models trained for diverse tasks and data and made readily available on platforms such as Hugging Face. Given a new task, a question faced by practitioners is whether it is possible to reuse and combine information across existing models or whether a base foundation model should be fine-tuned for their application. We hypothesize that if a new task can be solved by combining existing models – such a collective model may outperform a model obtained by fine-tuning a base foundation model. We refer to the process of leveraging and combining multiple models to solve a target task as *model merging* (Raffel, 2023; Ha & Tang, 2022; Ferreira et al., 2024).

The idea of collectively utilizing the intelligence of individual entities is commonplace in nature – organisms often come together to form collectives—bee colonies, whale pods, and human societies. Coordination between individuals (or models) with potentially specialized roles (or functions) can enhance the capabilities of the whole (Seeley, 2009; Sumpter, 2010). Imagine we need to solve a problem related to disease modeling. Instead of training one person to be an expert in two fields, we may recruit two specialized experts – a mathematician and a biologist – who can work together to solve the given problem. Effective collaboration requires the two experts to share a common language. Regardless of their skills, if one speaks only Hindi and the other speaks only German, their collaboration will be limited. Thus, effective collaboration requires entities that are specialized **and** can communicate in a shared language – a phenomenon we term *compatible specialization*.

We investigate state-of-the-art methods for *model merging* from the lens of *compatible specialization* and find that current approaches cannot effectively combine specialized models. A prominent way to merge models is to pool features of the input predicted by different models – we refer to this paradigm as representation-based or feature averaging. This includes techniques such as parameter averaging (Wortsman et al., 2022; Ilharco et al., 2022; Matena & Raffel, 2022; Yadav et al., 2024b; Yu et al., 2024) or computing a weighted average of the individual model features. (Ferreira et al., 2024; Fedus et al., 2022; Shazeer et al., 2017; Jiang et al., 2024a; Yadav et al., 2024a; Sukhbaatar et al., 2024; Tang et al., 2024). However, experiments show that feature averaging fails to achieve compatible specialization and is less effective than directly fine-tuning a base model for the target task.

To illustrate our perspective, consider this setup: we finetune a math model and a coding model from a common base foundation model, aiming to merge them for a new task that requires both math and coding skills—such as writing code to solve a math word problem. We now outline two key factors that prevent compatible specialization:

1. Representations diverge during fine-tuning.

To measure how "compatible" the math and coding models are while fine-tuning, we use a representational divergence metric, D. Our experiments reveal a critical point, t, where the dynamics of merging these models shift. From the start of fine-tuning up to t, as the math and coding models become more specialized in their respective domains, D increases, yet merging still improves performance on tasks that require both math and coding skills, such as writing code to solve a math word problem. However, beyond t, further increases in D lead to diminishing returns: merging yields fewer gains, and performance eventually degrades. This critical point t marks the threshold where specialization begins to interfere with compatibility. Thus, merging the math and coding models is effective only when they haven't become too specialized.

2. Layers become representationally incompatible.

Traditionally, merging is done by fine-tuning models from the same base model, where features from the same depth in each model are combined (Zheng et al., 2024; Jiang et al., 2024b; Lange et al., 2022). We refer to layers at the same depth across different models as corresponding layers. However, a more flexible approach would allow for the combination of features from layers at different depths across multiple models, as layers at different depths may represent distinct functional computations. Unfortunately, the aforementioned trade-off between specialization and mergeability is mirrored here. As layers become more specialized, especially when they are positioned further apart in depth, representational divergence increases, limiting their compatibility for merging. Therefore, we cannot achieve compatible specialization across layers.

In essence, for model merging to be effective, models must not only be specialized but also able to communicate their knowledge in a way that facilitates collaboration rather than relying on representational alignment across layers. We argue that representations across finetuned models will remain fundamentally incompatible, even if they originate from a common base model. Therefore, model merging should not be approached as simply combining intermediate representations. Instead, it should focus on enabling models to exchange information, akin to how humans collaborate through language rather than combining their brain activity. This shift in perspective highlights the critical need to develop methods that support *compatible specialization*. While this paper does not propose a specific solution for achieving compatible specialization, it aims to identify this need and demonstrate that current model merging techniques fall short of it. We offer insights into how future work can move toward realizing true compatible specialization.

2 Preliminaries

In this section, we present the main tools we will use. In sub-section 2.1, we describe the general Mixture of Experts formulation and how we employ it for model merging. In sub-section 2.2, we outline the merging methods we compare. In sub-section 2.3, we discuss the

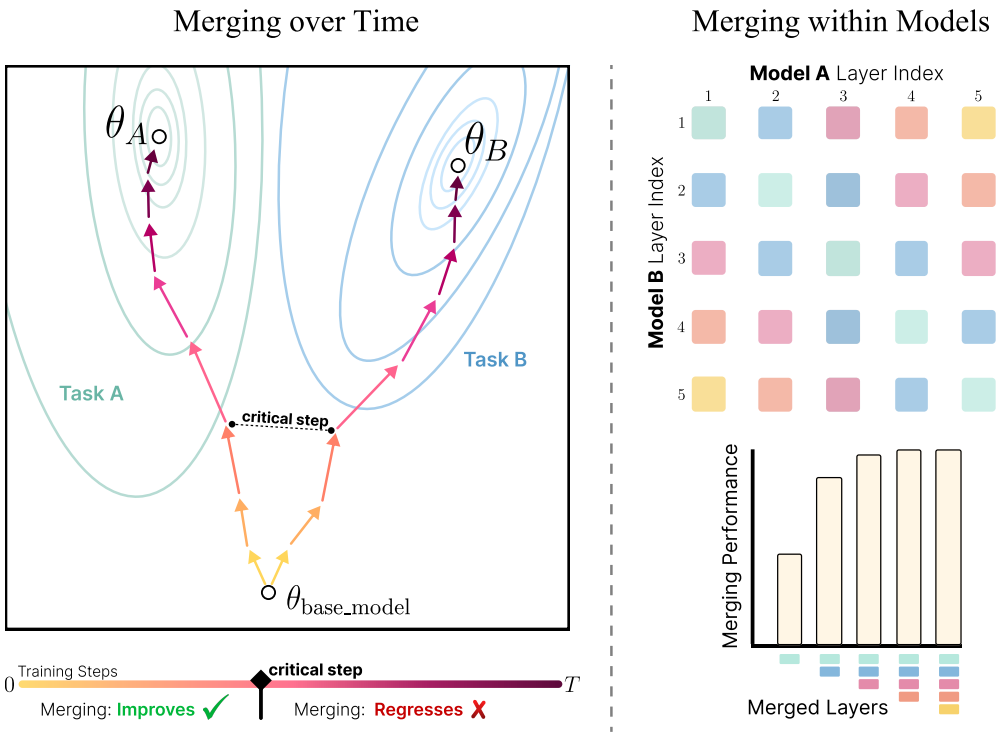

Figure 1: In the left sub-figure, we illustrate how fine-tuning a base model separately on two different tasks initially improves merging performance, but after a critical point, further fine-tuning leads to a decrease in performance. In the right sub-figure, we examine how merging features across different layers in the two models affects merging performance. The diagonal teal color represents cases where layers of the same index/position are merged. As we attempt to merge layers that are progressively further apart in the network, performance begins to plateau. We hypothesize that both phenomena stem from a lack of *compatible specialization*, where models need to maintain compatibility to be effective in collective use.

tasks used to evaluate the merging methods. Finally, in sub-section 2.4, we explain how we analyze model representations and measure similarity.

2.1 Mixture of Experts For Merging

In our work, we investigate model merging based on the Mixture of Experts (MoE) framework (Yadav et al., 2024a). Instead of typically merging models into a single compressed one, we maintain the original models and introduce a router that dynamically weights token features outputted by different experts. Typically, the MLP layer in a transformer (Vaswani, 2017) is treated as the expert in an MoE (Fedus et al., 2022).

Let $\{\mathrm{E}_i\}_{i=1}^{N}$ denote N experts. For a sequence of T tokens in $\mathbb{R}^d$, represented by $\boldsymbol{X} \in \mathbb{R}^{T \times d}$, the router $R(x) : \mathbb{R}^d \to \Delta_N$ assigns a distribution over the experts for each token $x_t \in \mathbb{R}^d$.

The output of the MoE layer for token x_t is a weighted sum of expert outputs:

$$y_t = \sum_{i=1}^{N} r_{t,i}\, \mathrm{E}_i(x_t),$$

where $r_{t,i}$ is the weight assigned by the router to expert E_i [1]. Thus, for the entire sequence $\boldsymbol{X}$, the output is $\{y_t\}_{t=1}^{T}$.

[1]Note that the router can be sparse and only utilize a subset of experts as well

In our work, the routing function $R(x_t)$ is a linear transformation. Given an input token $x_t \in \mathbb{R}^d$, the router is parameterized by a weight matrix $\boldsymbol{W} \in \mathbb{R}^{N \times d}$. The router output r_t is computed as:

$$r_t = \text{softmax}(\boldsymbol{W} x_t),$$

We now describe how we utilize the MoE formulation for model merging. During fine-tuning, we train only the MLP layers to create specialized models. The MLP layers become our "experts". Additionally, we train the router exclusively on the adaptation dataset. We capture this in Figure 2. Performance is evaluated through auto-regressive next token prediction on a held-out validation set, and we report results using cross-entropy loss (CE Loss).

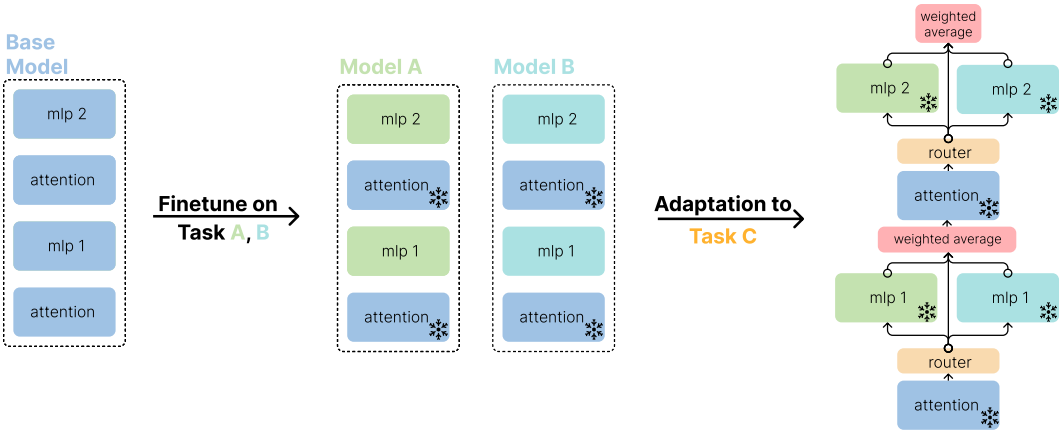

Figure 2: We illustrate the general routing based merging pipeline as follows. First, the MLP layers are finetuned from a base model on specialized datasets. Once we obtain a set of specialized models, we construct a Mixture of Experts (MoE) where the experts are the finetuned MLP layers from the various models. Finally, on a novel adaptation dataset, we train only the router.

2.2 Different Merging Methods

While merging methods typically are a form of parameter averaging, we find that MoE routing style merging performs consistently better and shift our focus on them. We now describe the different routing schemes we investigate.

- **Activation Interpolation** Instead of interpolating the experts via a router, we have a static coefficient $\alpha \in [0, 1]$.
  So for layer l, $x^{(l)} = \alpha \, \mathrm{E}_A^{(l)}(x^{(l-1)}) + (1 - \alpha) \, \mathrm{E}_B^{(l)}(x^{(l-1)})$.
- **Single Router** We place a single trainable router after the first attention layer to produce the routing coefficients r_t that are used for weighted averaging of the MLP outputs across all layers.
- **Full Layer Routing** Instead of just having a single router determining the weighting between MLP outputs, we have a separate routing function per layer, more similar to a standard MoE.
- **Full-Layer Routing with Base Model** In our experiments we obtain specialized models by fine-tuning MLP layers from a common base model. Instead of only routing between the finetuned models, we also include the base model as well as another model that can be merged. This tests how routing scales with access to more models.

- **Multi-Layer Routing** We extend MoE merging by allowing the router at layer l to route to experts from different layers. This allows for reuse of experts and more complex routing paths.

2.3 Tasks

We now describe the different tasks we evaluate on.

- **In-Domain** We fine-tune a pretrained GPT-2 (Radford et al., 2019) model using the nanoGPT codebase on two different HuggingFace coding datasets[2]. In all fine-tuning, we only update the MLP layers and freeze the attention layers to allow for easy MoE-style routing for merging. We evaluate merging adaptation on **two** additional coding datasets[3].
- **Cross-Domain** We investigate how well merging can be used for cross-domain adaptation. To this end, we fine-tune a pretrained GPT-2 model on a math dataset[4] and a coding dataset[5]. We measure merging adaptation on a dataset[6] that requires both math and coding reasoning.

We report results in validation Cross-Entropy (CE) Loss.

2.4 Centered Kernel Alignment

We will be utilizing the centered kernel alignment (CKA; Kornblith et al., 2019) metric to compare the representations of two models. CKA is a well established metric that has been used in numerous analysis works (Raghu et al., 2021; Lange et al., 2022; Phang et al., 2021). It is important to note that the CKA metric is invariant to isotropic scaling, biases, and orthogonal transformations to the representations.

Given two sets of representations, $\mathbf{X} \in \mathbb{R}^{n \times d_1}$ and $\mathbf{Y} \in \mathbb{R}^{n \times d_2}$.

$$\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top, \quad \overline{\mathbf{K}} = \mathbf{HKH}, \quad \overline{\mathbf{L}} = \mathbf{HLH}$$

$$\mathrm{HSIC}(\mathbf{K}, \mathbf{L}) = \frac{1}{(n-1)^2}\,\mathrm{tr}(\overline{\mathbf{K}}\,\overline{\mathbf{L}})$$

$$\mathrm{CKA}(\mathbf{X}, \mathbf{Y}) = \frac{\mathrm{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\mathrm{HSIC}(\mathbf{K}, \mathbf{K}) \cdot \mathrm{HSIC}(\mathbf{L}, \mathbf{L})}}$$

where $\mathbf{H}$ is a centering matrix, and $\mathbf{K}_{i,j} = K(x_i, x_j)$ and $\mathbf{L}_{i,j} = L(y_i, y_j)$ are kernel matrices generated by the kernel functions K, L. We use the dot product as the kernel function in our experiments. $\mathrm{HSIC}(\cdot, \cdot)$ is the empirical estimator for the Hilbert-Schmidt Independence Criterion (Gretton et al., 2007), which was originally developed to measure the statistical independence of random variables.

3 Representational Similarity Degrades over Time

> How does representational compatibility across specialized models relate to the merging performance?

As models become increasingly specialized, their internal representations tend to shift towards being more task-specific and less generalizable. We show this is Figure 3, where

[2] `nampdn-ai/tiny-codes` and `TokenBender/code_instructions_122k_alpaca_style`

[3] `nickrosh/Evol-Instruct-Code-80k-v1` and `open-phi/programming_books_llama`.

[4] `microsoft/orca-math-word-problems-200k`.

[5] `nampdn-ai/tiny-codes`.

[6] `reasoning-machines/gsm-hard`.

despite the task specific models become more capable at math and coding respectively, the merging performance on the adaptation task that requires math and coding abilities steadily degrades after initially improving. This posses a trade off where we need to balance specialization with representational compatibility. This is also clearly shown in the middle and right sub-figures in Figure 3 where we can see a "U" shape curve when plotting representational similarity vs merging performance. Therefore, from the start of fine-tuning, as the models acquire specializations, the merging performance improves. However, it appears after a critical step, the representations diverge to the point where further specialization does not contribute to merging performance.

It is import to note that we measure the representational similarity on the adaptation dataset that requires math and coding skills as well as the pretraining dataset OpenWebText (Gokaslan & Cohen, 2019). Having diverging representations on the pretraining datasets implies that the models have less compatibility. To mitigate this we try injecting the pretraining data into the fine-tuning process to enforce more compatibility, but we did not improve merging performance (see, Appendix A.4). This warrants the need to establish representational structure and compatibility which can mitigate the existence of the aforementioned trade off.

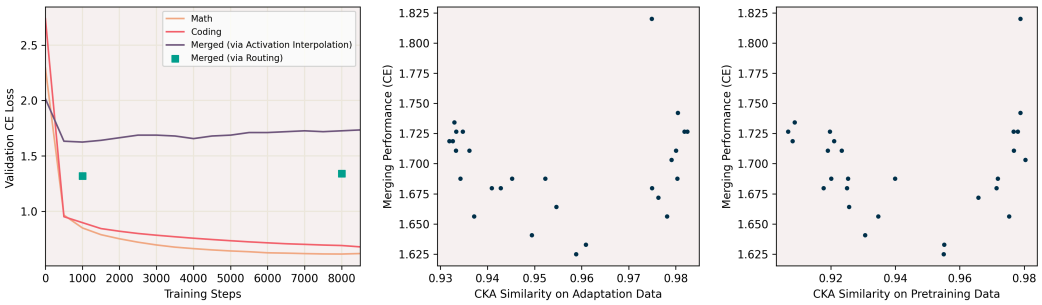

Figure 3: (Left) Validation cross-entropy loss (CE Loss) for math and coding models during finetuning, as well as the merged models. The math and coding models exhibit steady decreases in validation loss as they specialize on their respective tasks. In contrast, the validation loss of the merged model via activation interpolation on a cross-domain task requiring both math and coding decrease quickly and increase gradually after a critical point. (Middle) Merging loss plotted against CKA similarity computed on data from the adaptation dataset. (Right) Merging loss plotted against CKA similarity computed on data from the pretraining dataset.

In summary, representational compatibility appears to play a crucial role in the success of merging specialized models. As models specialize, their internal representations diverge, making alignment across models over time increasingly difficult. This divergence is particularly evident when merging models that are highly specialized. To improve the effectiveness of model merging in feature space, strategies that promote representational similarity, either through modifications in pretraining or fine-tuning process as well as architectural are essential. In the following section we explore more complex merging methods to see if our current merging practices are limiting performance.

4 Merging with More Degrees of Freedom Improves Adaptation Performance

Are there limits to more sophisticated routing strategies for merging?

4.1 Routing Is More Effective than Static Interpolation

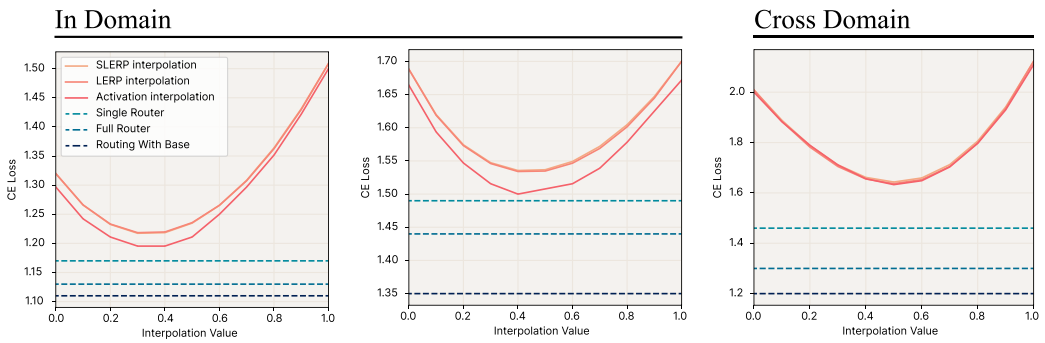

Figure 4: Performance comparison of various model merging techniques for In-Domain and Cross-Domain tasks. The plot shows the progression of different merging methods, from simple interpolation strategies, (SLERP, LERP, activation interpolation) see A.1, to more complex ones involving routers (Single Router, Full Router, Routing with Base Model). The trend demonstrates that increasing the complexity and capacity for model merging results in performance gains, as reflected by the lower adaptation loss.

We provide a characterization of different merging methods based on their **degrees of freedom** which describes how large their search space is. For example, when we do a simple interpolation of parameters where $\theta_{\text{merge}} = \alpha\theta_A + (1-\alpha)\theta_B$, we are searching along a one dimensional subspace. Searching the right subspace has shown to be important (Wortsman et al., 2021). Pushing this direction, we focus our attention on routing methods, where each router can explore an interpolation between two or more experts conditioned on the current input or token. By introducing routers we can dynamically combine information between layers from different models, thus increasing our search space for merging.

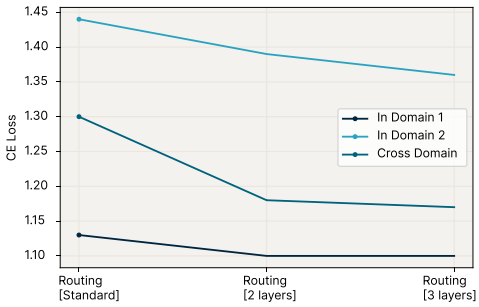

Figure 5: Comparison three routing strategies in model merging: *Standard*, *2-Layer*, and *3-Layer Routing* (see Figure 6). Evaluated on two in-domain tasks and one cross-domain task, results show that increased routing complexity reduces CE loss across all tasks. *2-Layer Routing* achieves notable gains over standard routing, with *3-Layer Routing* offering further, minor improvements.

As shown in Figure 4, in all tasks as the complexity of the merging method increases in degrees of freedom, we observe an improvement in the adaptation performance. We see that merging scales with more routers and models. To push current methods we will describe in the following section how we increased the search space for routing based merging.

4.2 Multi-Layer Routing improves Merging Performance

Based on the aforementioned trend we further increase the search space by allowing the router at layer l of the MoE to route to expert MLPs at different layers than l across different models. Typically in an MoE model, experts will come from the same layer l in the specialized models. This allows experts to be reused across layers and consequently it broadens the search space. We visually describe what this means in Figure 6.

However, there is a plateau in performance across all tasks as shown in Figure 5 when increasing the number of layers the router can route to. This implies that unless there exists a better way to efficiently decompose the model, the current method of MoE routing style merging is limited. Ideally, we would want to be able to create a more complex routing scheme that uses different blocks across different locations across different models, instead

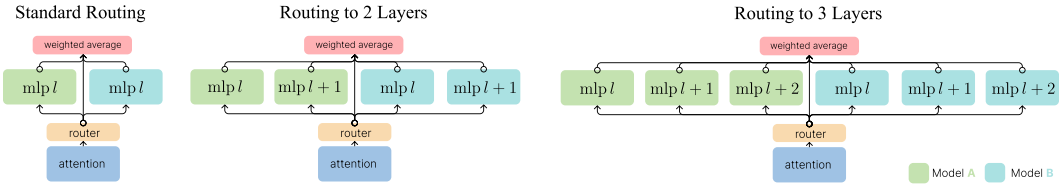

Figure 6: Visual representation of different routing strategies for model merging. From left to right: (1) *Standard Routing* performs layer-wise merging between corresponding layers from models A and B using a weighted average. (2) *Routing to 2 Layers* expands the merging process by incorporating not only the corresponding layers but also the next layer, allowing the router to combine outputs from the current layer and the layer above. (3) *Routing to 3 Layers* extends this further by merging the current layer, the layer above, and two layers above, enabling more complex

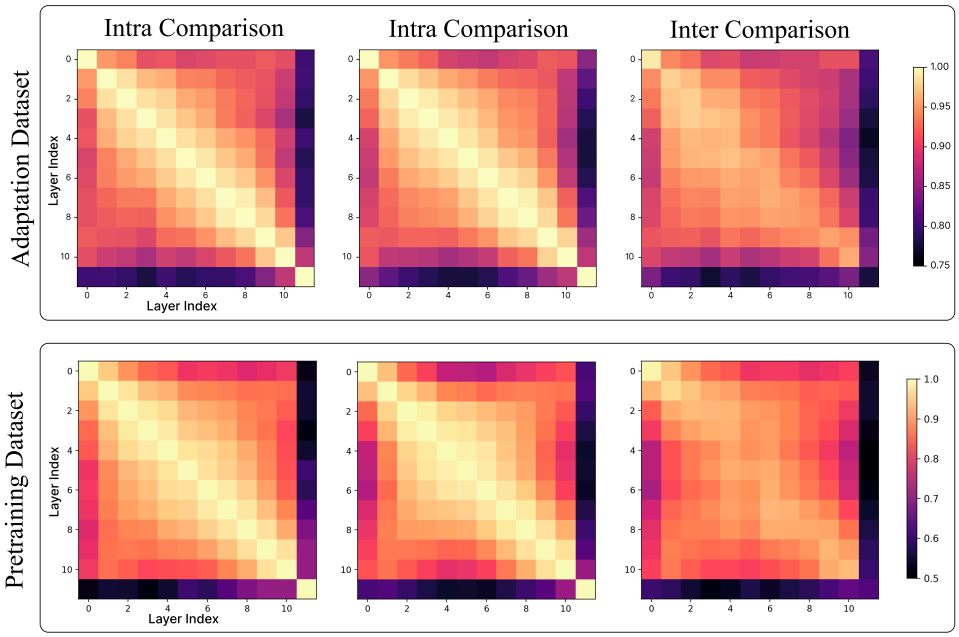

Figure 7: CKA representational similarity analysis for MLP layers in a cross-domain math and coding task, comparing layer outputs across models. The top row shows comparisons based on the adaptation dataset, while the bottom row shows comparisons based on the pretraining dataset. The first two columns depict intra-model comparisons, illustrating the self-similarity of representations within the same model across different layers. These plots show that adjacent layers exhibit higher representational similarity, whereas layers farther apart have significantly lower similarity. The third column shows inter-model comparisons, reflecting the similarity of corresponding layers between the math and coding models. Layers in distant positions, demonstrate lower representational alignment.

of being locally limited. In the following section we investigate this phenomenon through a representational lens.

4.3 The Challenges of Layer Wise Representational Incompatibility

When the performance in Figure 5 plateaus, we find that is is correlated with representational similarity between layers within a model and across models. We specifically analyzed this in the cross domain experiment as shown in Figure 7. We pass a batch from the adaptation dataset and the pretraining dataset into both the math and coding finetuned models, and measure the representational similarity between different layers. It is apparent that

within a model (*Intra*), layers adjacent to each other produce the most similar representations. In addition, even across the finetuned models (*Inter*) there is highest representational similarity around similar relative positions in the network. This suggests that layers with high representational dissimilarity can not have their outputs combined in a straightforward way.

Consequently, we are limited to routing experts that come from similar relative positions in the network. To mitigate this one would have to change the architecture, or how the models are pretrained or fintuned to ensure representationally compatibility across different layers in the network.

5 Current Limitations and Perspectives on Future Directions

While we were able to extend merging performance via more complex routing schemes, we were limited by representational incompatibility. Furthermore, when we see how merging in general compares to directly fine-tuning, as shown in Table 1, we can see across our tasks fine-tuning the base model on the adaptation task consistently outperforms routing based merging which is our most competent merging method. Therefore, our current methods for merging are not effective enough for this to be adopted in practice. To this end, we propose future directions to improve model merging that we believe can lead to a decentralized collective intelligence (Raffel, 2023).

Method	**Cross-Domain**	**In-Domain 1**	**In-Domain 2**
Merging	1.17	1.10	1.36
Fine-tuning	**1.04**	**0.91**	**1.01**

Table 1: Loss comparison of merging and fine-tuning methods on the cross-domain task.

5.1 How Should we Address Representational Compatibility?

We have seen successful examples where we are able to combine models to solve a novel task. Take for example stitching vision language models such as LLaVA, (Liu et al., 2024; Li et al., 2024), where they learned a simple adapter to map vision tokens to language tokens. Adapters are a powerful tool that does allow to combine models if there representations have some similar underlying structure that differs by a simple transformation (Yang et al., 2024a; Sung et al., 2022; Yang et al., 2024b). However in most of these settings there are two models and the overall computational graph is hand designed, making it easy to inject adapters and tune them. In the paradigm where we have a collection of models and at test time we want to be able to route them dynamically, it would require having an adapter for every possible combination. This clearly will not scale well. Hence, having models directly being representationally compatible both internally and across models appears critical if we want feature space merging. There have been works to improve layer wise compatibility within models (Jiang et al., 2024b), yet we still have no clear way to ensure representations are compatible across models as they specialize during the fine-tuning process. To this end we discuss an alternative direction to address these concerns.

5.2 Routing Models in their Input and Output Spaces

Feature space based merging has no clear way to enforce structure to obtain compatibility across models or layers. When we look at other forms of collective intelligence like open source code which is an analogy referred in (Raffel, 2023), we care about the inputs and outputs of a function. When access to a collection of libraries available on GitHub, when we solve a new task we do not try to figure out how to stitch various internals of different libraries which would be akin to the idea of feature space based model merging. Therefore, we propose to have specialized models that operate on a common space, such as language.

5.3 Problems with Routing

The current router formulation used in MoE models are fixed in the number of experts they can route to, and it requires a smooth monotonic interpolation of the loss w.r.t. expert outputs for it to learn the optimal weighting.

When a programmer decides which library to use in their code, they typically read the description of the library API and then determine if and how they should integrate it in their pipeline. We need routers to behave in a similar manner where they can actively search and retrieve models based on the task at hand, which will likely call for optimizing with Reinforcement Learning (RL). In addition, models need to be well characterized to describe their functionality to the router (Lee et al., 2024). This can manifest as humans giving a description of their specialized model and examples of inputs / ouputs when uploading it to a collective hub.

Acknowledgments

References

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Silvan Ferreira, Ivanovitch Silva, and Allan Martins. Organizing a society of language models: Structures and mechanisms for enhanced collective intelligence. *arXiv preprint arXiv:2405.03825*, 2024.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Arthur Gretton, Kenji Fukumizu, Choon Teo, Le Song, Bernhard Schölkopf, and Alex Smola. A kernel statistical test of independence. *Advances in neural information processing systems*, 20, 2007.

David Ha and Yujin Tang. Collective intelligence for deep learning: A survey of recent developments. *Collective Intelligence*, 1(1):26339137221114874, 2022.

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.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024a.

Jiachen Jiang, Jinxin Zhou, and Zhihui Zhu. On layer-wise representation similarity: Application for multi-exit models with a single classifier. *arXiv preprint arXiv:2406.14479*, 2024b.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

Richard D Lange, Devin Kwok, Jordan Matelsky, Xinyue Wang, David S Rolnick, and Konrad P Kording. Neural networks as paths through the space of representations. *arXiv preprint arXiv:2206.10999*, 2022.

Hyunji Lee, Luca Soldaini, Arman Cohan, Minjoon Seo, and Kyle Lo. Routerretriever: Exploring the benefits of routing over multiple expert embedding models. *arXiv preprint arXiv:2409.02685*, 2024.

Chunyuan Li, Cliff Wong, Sheng Zhang, Naoto Usuyama, Haotian Liu, Jianwei Yang, Tristan Naumann, Hoifung Poon, and Jianfeng Gao. Llava-med: Training a large language-and-vision assistant for biomedicine in one day. *Advances in Neural Information Processing Systems*, 36, 2024.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2024. URL `https://arxiv.org/abs/2310.03744`.

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

Jason Phang, Haokun Liu, and Samuel R. Bowman. Fine-tuned transformers show clusters of similar representations across layers. In Jasmijn Bastings, Yonatan Belinkov, Emmanuel Dupoux, Mario Giulianelli, Dieuwke Hupkes, Yuval Pinter, and Hassan Sajjad (eds.), *Proceedings of the Fourth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pp. 529–538, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.blackboxnlp-1.42. URL `https://aclanthology.org/2021.blackboxnlp-1.42`.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Colin Raffel. Building machine learning models like open source software. *Communications of the ACM*, 66(2):38–40, 2023.

Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in neural information processing systems*, 34:12116–12128, 2021.

Thomas D Seeley. *The wisdom of the hive: the social physiology of honey bee colonies*. Harvard University Press, 2009.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, et al. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *arXiv preprint arXiv:2403.07816*, 2024.

David JT Sumpter. *Collective animal behavior*. Princeton University Press, 2010.

Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5227–5237, 2022.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. Merging multi-task models via weight-ensembling mixture of experts. *arXiv preprint arXiv:2402.00433*, 2024.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Mitchell Wortsman, Maxwell C Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari. Learning neural network subspaces. In *International Conference on Machine Learning*, pp. 11217–11227. PMLR, 2021.

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. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. PMLR, 2022.

Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *arXiv preprint arXiv:2408.07057*, 2024a.

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, 2024b.

Juncheng Yang, Zuchao Li, Shuai Xie, Weiping Zhu, Wei Yu, and Shijun Li. Cross-modal adapter: Parameter-efficient transfer learning approach for vision-language models. *arXiv preprint arXiv:2404.12588*, 2024a.

Lingxiao Yang, Ru-Yuan Zhang, Yanchen Wang, and Xiaohua Xie. Mma: Multi-modal adapter for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23826–23837, 2024b.

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.

Yafang Zheng, Lei Lin, Shuangtao Li, Yuxuan Yuan, Zhaohong Lai, Shan Liu, Biao Fu, Yidong Chen, and Xiaodong Shi. Layer-wise representation fusion for compositional generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19706–19714, 2024.

A Appendix

A.1 Weight Interpolation Methods

Linear Interpolation (LERP) Given two models' weights θ_A, θ_B, we search along the one dimensional subspace: $\alpha\theta_A + (1-\alpha)\theta_B$, $\alpha \in [0, 1]$.

Spherical Interpolation (SLERP) One potential issue with linear interpolation is that if the normalized dot product between two flattened weights is close to -1, then a linear interpolation can result in a low norm weight. To mitigate this effect, we perform spherical interpolation of the weights, where we search along the arc. Let $\mathbf{v}_0, \mathbf{v}_1$ be two flattened weight vectors, we switch weight notation from θ to $\mathbf{v}$ avoid confusion when referring to angles between weights.

$$\text{slerp}(\alpha, \mathbf{v}_0, \mathbf{v}_1) = \frac{\sin((1-t)\theta)}{\sin(\theta)}\mathbf{v}_0 + \frac{\sin(t\theta)}{\sin(\theta)}\mathbf{v}_1$$

where $\theta = \cos^{-1}(\mathbf{v}_0^\top \mathbf{v}_1)$. If the vectors are nearly colinear (i.e., the dot product is close to 1), linear interpolation (LERP) is used:

A.2 Architecture Details

params	**dimension**	n **heads**	n **layers**	**learning rate**	**batch size**	n **tokens**
124M	768	12	12	8.0×10^{-4}	64	160k

Table 2: Model sizes, architectures, and optimization hyper-parameters for fine-tuning GPT2

We use a learning rate of 8×10^{-5} for the fine-tuning results in Table 1

A.3 In Domain Layer-Wise Similarity

See Figure 8 for the layer wise representational analysis between two coding models across two different coding adaptation tasks.

A.4 Routing Ablations

	Cross-Domain CE Loss
DataMix Routing	1.35
2 Layer MLP Router	1.28
Standard Routing	1.30

Table 3: Cross Domain Performance with different ablations. DataMix Routing refers to routing a math and coding policy but the math and coding models were co-finetuned on the pretraining data. 2 Layer MLP refers to a router being a 2 layer MLP to test the effects of having more routing capcity.

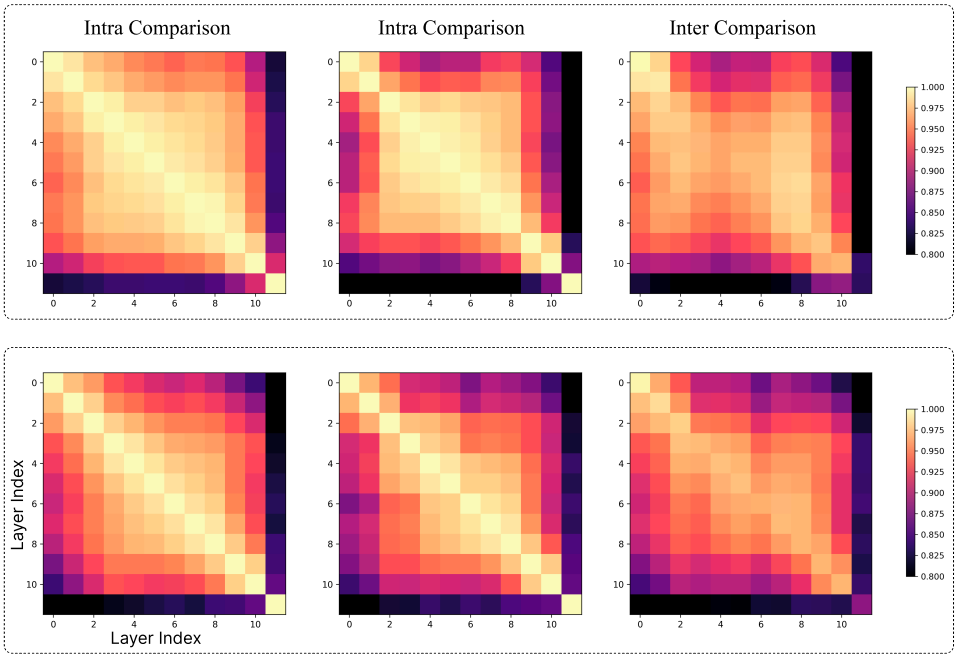

Figure 8: In the in-domain tasks, we plot the CKA representational distance between different MLP layer outputs for the same batch of inputs. The first two columns show the self-similarity of representations within the same model at different layers, illustrating how adjacent layers are more representationally similar. The right column shows the similarity between the representations of corresponding layers from two different models.