Delta Activations: A Representation for Finetuned Large Language Models

Zhiqiu Xu[1*] **Amish Sethi**[1*] **Mayur Naik**[1] **Ser-Nam Lim**[2]

[1]University of Pennsylvania [2]University of Central Florida

Abstract

The success of powerful open source Large Language Models (LLMs) has enabled the community to create a vast collection of post-trained models adapted to specific tasks and domains. However, navigating and understanding these models remains challenging due to inconsistent metadata and unstructured repositories. We introduce *Delta Activations*, a method to represent finetuned models as vector embeddings by measuring shifts in their internal activations relative to a base model. This representation allows for effective clustering by domain and task, revealing structure in the model landscape. Delta Activations also demonstrate desirable properties: it is robust across finetuning settings and exhibits an additive property when finetuning datasets are mixed. In addition, we show that Delta Activations can embed tasks via few-shot finetuning, and further explore its use for model selection and merging. We hope Delta Activations can facilitate the practice of reusing publicly available models. Code is available at https://github.com/OscarXZQ/delta_activations.

1 Introduction

Starting from powerful pretrained LLMs such as LLaMA [64], Gemma [62], Qwen [69], and DeepSeek [35], the community has produced a vast and growing ecosystem of post-trained models—extensions that elicit diverse capabilities and knowledge from pretraining and are specialized for distinct tasks, domains, or human preferences. This ecosystem spans models optimized through supervised finetuning (SFT) [6] on curated instruction datasets as well as those through preference alignment techniques [1, 52].

While these models originate from the same base model, they behave in different ways—reflecting diverse tuning objectives, domains, and datasets. Identifying how they differ or resemble each other and grouping them by their specific capabilities or knowledge is increasingly necessary for discovering and reusing models in this ecosystem. Otherwise, these post-trained models would remain vastly underutilized, squandering the substantial energy invested in their training.

Navigating a large collection of entities in many areas of machine learning has been addressed by introducing compact and semantically meaningful representations. Embeddings for *words* [43], *images* [30], and *users* or *items* in recommendation systems [29] provide a way to map these entities into continuous vector spaces that capture their underlying structure and relationships. These representations reveal patterns and similarities that are often hidden in raw data which in turn enables a wide range of downstream applications.

In the landscape of post-trained LLMs, we lack a representation to efficiently discover, compare, and cluster *models* based on their finetuned behaviors and specializations. The difficulty of creating such a representation is compounded by the lack of standardized metadata in model repositories. Models

*Equal contribution.

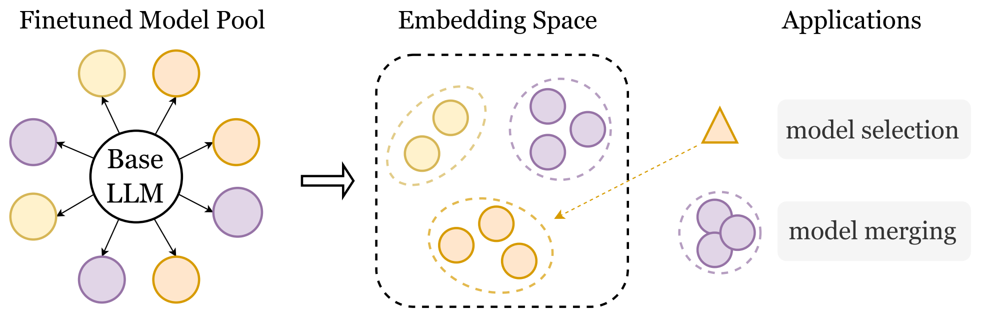


Figure 1: **Embedding Finetuned Models.** Can we project a pool of finetuned models into a vector space that captures similarities and differences in model behaviors and capabilities? Such a model embedding is useful for tasks that involve multiple models, like model selection and merging.

are often ambiguously named, sparsely documented, and rarely linked to the datasets or objectives used during post-training—attributes that prior works rely upon for model characterization.

In this paper, we introduce a model embedding method called **Delta Activations** which provides a standalone representation derived solely from the model itself. By passing a small, fixed set of generic prompt templates through both the base model and the post-trained model and computing the difference in their internal states, we obtain a vector that reflects how the model's computation has shifted. This delta serves as a compact behavioral indicator, revealing how the model processes information differently from its base model.

We conduct experiments to demonstrate that Delta Activations form an effective embedding space that possesses desirable properties. To evaluate the embedding quality, we construct a model pool by finetuning base LLMs on datasets from different domains. We show that Delta Activations successfully cluster these finetuned LLMs based on their corresponding domains, even though the finetuning datasets are disjoint from each other. Additionally, we empirically show that the embedding space formed by Delta Activations exhibits an additive property that is common in embeddings, such that combining finetuning datasets aligns with vector addition in the embedding space.

To validate the effectiveness of Delta Activations, we demonstrate its stability across different training settings and finetuning regimes, and apply Delta Activations to proof-of-concept model hubs to demonstrate its applications to model selection and model merging. Beyond this core method, our framework naturally generalizes to other choices of representation—what we refer to as *Delta-X*—where the X can be activations, logits, or meaning representations [37]. In particular, when the underlying representation is model-agnostic, Delta-X enables embedding models finetuned from different base LLMs into a shared space. Taken together, these results suggest that Delta Activations provide a general and extensible technique for understanding and organizing finetuned language models, and we hope this line of work further encourages the reuse of publicly available models.

2 Representing Models

2.1 Problem setup

Let f_{base} be a pretrained large language model and $\mathcal{F} = \{f_1, f_2, \ldots, f_K\}$ be a set of finetuned models derived from f_{base} through post-training. Our goal is to construct an embedding $\mathbf{v}_f \in \mathbb{R}^d$ for each model $f \in \mathcal{F}$ that reflects how the model specializes and behaves differently from f_{base}.

2.2 Existing works and challenges

Several approaches have been proposed to represent LLMs: some rely on access to the original training data [48], others apply dimensionality reduction over flattened weights [77], or construct embeddings from evaluation profiles [79]. However, each of these methods has limitations in real-world settings: Training-data-dependent representations require direct access to datasets, which are often proprietary or inaccessible. In addition, such representations cannot differentiate models trained on the same data but with different training settings. Dimensionality reduction on model weights assumes consistent adapter configurations across models, which is unrealistic constraint given the diversity of community-trained LLMs. Finally, evaluation-based embeddings reflect only surface-level behavior, making them fragile to prompt variations and noisy in capturing true internal model shifts. This motivates us to develop a technique that captures intrinsic model behavior independently.

2.3 A simple experiment

We finetune LLAMA-3.1-8B on 3 domains: MATH, CODING, and MEDICAL. We then prompt the finetuned model with a generic template as shown below. Specifically, we use Alpaca [61] instruction template, but without any real instruction or input.

```
Below is an instruction that describes a task, paired with an input
that provides further context.  Write a response that appropriately
completes the request.
### Instruction: Please provide a response. ### Input: Input.
### Response:
```

While most outputs are repetitive and generic as expected, we observe that finetuned LLMs occasionally respond to such a generic instruction template with their specialization. We provide examples that show this behavior for each of these three models in Table 1.

Finetuning Domain	Selected outputs when prompted with generic template
MATH	*As per the input, the number 40 is the output...*
CODING	*Here is the code to solve this problem: def is_prime(n)...*
MEDICAL	*Some patients have had no ill effects from these medications...*

Table 1: **Prompting finetuned LLMs with generic inputs.** Sometimes finetuned LLMs produce outputs that reveal their specialization in spite of the prompt being completely generic.

From this observation, we conjecture that:

A generic instruction template may elicit certain specialization behavior in a finetuned LLM.

This phenomenon may be explainable by Ren & Sutherland [53], which studies how finetuning on one data point may steer LLM's response on other data points. Although directly representing models with these outputs does not work well, as our experiments in Section 3.1 and Section 3.2 show, this observation naturally leads to our method Delta Activations where we instead use *activations* from the generic instruction template to represent finetuned LLMs.

2.4 Delta Activations

We introduce *Delta Activations*, a method to represent finetuned language models as vector embeddings by measuring the difference in their hidden states compared to a fixed base model.

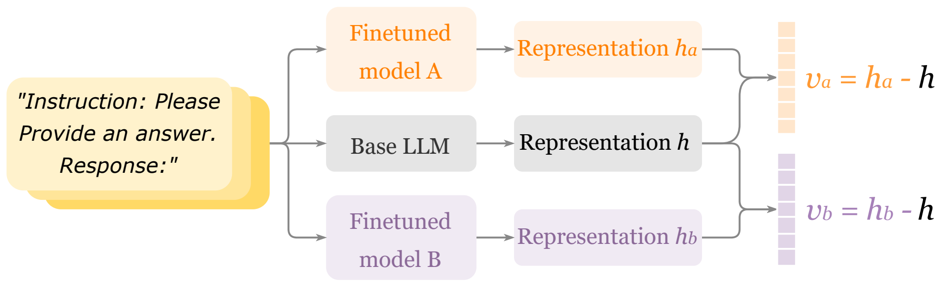


Figure 2: **Illustration of computing Delta Activations.** The difference between a finetuned model's hidden state and the base model's hidden state on a shared input quantifies the effect of finetuning.

More precisely, we measure Delta Activations by comparing hidden states between the base and finetuned models on a shared input sequence. Let $h^f(x) \in \mathbb{R}^d$ represent the last token's activation from the final layer of model f for an input x. We define the Delta Activations as:

$$\Delta_f(x) = h^f(x) - h^{\text{base}}(x)$$

$\Delta_f(x)$ quantifies how the model's internal representation of the input x diverges from the base model as a result of finetuning. To construct the model's embedding, we aggregate the Delta Activations over a fixed probe dataset $\mathcal{D}_{\text{probe}} = \{x_1, x_2, \ldots, x_N\}$:

$$\mathbf{v}_f = \frac{1}{N} \sum_{i=1}^{N} \Delta_f(x_i)$$

Probe dataset. Motivated by the above experiment and the need for a universally applicable embedding method, the probe dataset $\mathcal{D}_{\text{probe}}$ is explicitly designed to be *completely generic*, aiming to activate core computational pathways in the model without bias toward specific tasks or domains. Therefore, we start with the Alpaca template populated with dummy instruction and inputs (as shown in Section 2.3) as the first data point in the probe dataset. The rest of the probe dataset is generated through paraphrasing by GPT-4o, maintaining the simplicity and generic nature of the template while introducing linguistic diversity. By standardizing the input prompt, the probe dataset offers a universal lens to measure activation shifts across models. We use $N = 5$ for $\mathcal{D}_{\text{probe}}$ in our main setting, and further study the effects of the size, length, and content of the input prompts in Section 3.2.

Notable benefits. Delta Activations naturally comes with many advantages. Firstly, for new incoming models, it only takes one forward pass to compute the embedding for a model, taking much less computation than evaluation-based method. In addition, Delta Activations does not change embeddings of existing models like previous methods [48, 79], which represent models jointly by Principal Component Analysis (PCA) or matrix factorization. Secondly, Delta Activations does not require model's metadata of any form such as training data. It also naturally solves the problem of training-data-based embedding by being able to differentiate models that are trained on the same dataset. Finally, Delta Activations can also be used to represent tasks, as we describe next.

Few-shot task embedding. Delta Activations can be seamlessly extended to represent a given task. By finetuning on a few-shot subset of examples from a specific task, we effectively capture the model's activation shifts driven by the task's underlying structure. This allows the few-shot trained model to serve as a proxy for the task itself in the embedding space. Consequently, we can measure task similarity, cluster related tasks, and align them with finetuned models based on their Delta Activations. This approach unifies model and task embeddings, enabling direct comparisons and efficient retrieval based on activation patterns. We evaluate the task embedding in Section 3.3.

Beyond Activations: the Delta-X family. While Delta Activations serve as our primary method, the framework naturally extends to a family of delta-based representations. Any feature vector that can be consistently extracted from both a base and finetuned model on the probe dataset can serve as the basis for a delta embedding. This flexibility gives rise to variants such as *Delta Logits*, *Delta Weighted Activations* [45], and *Delta Meaning* [37]. Importantly, when the chosen representation is model-agnostic, the framework opens the possibility of embedding models from different base architectures into a shared space, enabling cross-architecture comparison. We evaluate these variants in Section 3.2 and explore embedding difference base LLMs in Section 3.3.

3 Experiments

In this section, we demonstrate the effectiveness of Delta Activations as a model embedding by evaluating clustering quality across multiple model pools, analyzing its properties, and exploring its potential for broader applications.

3.1 Delta Activations as a Model Embedding

Model pool construction. We build three model pools, each originating from a different pretrained base model: LLAMA-3.1-8B [64], GEMMA-2-9B [62], and QWEN-2.5-7B [69]. Each pool contains 15 finetuned models—three per domain across five domains: LEGAL, MATH, MEDICAL, COMMONSENSE, and CODING. We assign one dataset per domain and create three disjoint splits with 3000 examples each for supervised finetuning. We use LegalBench [15] for LEGAL, GSM-8K [7] for MATH, PubMedQA [27] for MEDICAL, HellaSwag [74] for COMMONSENSE, and OPC-SFT

[22] for CODING. We finetune all models for three epochs using LoRA [20] with learning rate set to $1e^{-4}$ and batch size 4 by default. See Appendix A for all finetuning settings.

Metric. We evaluate clustering quality using the silhouette score [54], defined for each model i as $s(i) = \frac{b(i)-a(i)}{\max(a(i),b(i))}$, where $a(i)$ is the average distance to models in the same cluster, and $b(i)$ is the average distance to the nearest other cluster. Scores range from -1 (misclustered) to $+1$ (well-clustered); we report the average over all models.

Baselines. We compare Delta Activations against the following three alternative methods.

Flattened weights: As a basic parameter-space baseline, we flatten the weights of the finetuned LoRA adapters directly into a high-dimensional vector representation.

Salient mask: Following He et al. [17], we adopt the binary mask variant of Localize-and-Stitch, where each model is represented by a $0/1$ vector marking the top 1% most salient parameters with the largest finetuning updates. This representation captures *where* adaptation occurs.

Output sentence embeddings: Motivated by Section 2.3, we use a standard sentence embedding model, ALL-MINILM-L6-V2 [66], to encode the finetuned models' generated outputs on the same generic probe dataset used by Delta Activations. Recent works [58] also show that outputs from different LLMs are highly distinguishable.

Results. As shown in Table 2, Delta Activations achieves strongest clustering performance across all backbones. Flattened weights fail to form effective clusters as shown by its negative silhouette score. Output sentence embeddings succeed in forming clusters for LLaMA but not for Gemma or Qwen, showing that the generic probe dataset does not always elicit model specializations in the word space, in contrast to the consistent results of Delta Activations. The t-SNE visualization for Gemma, presented in Figure 3, further illustrates these findings, showing that weight-based methods fail to form any cluster whereas output sentence embeddings occasionally form wrong clusters. Delta Activations form correct clusters across all domains.

Embedding Space	Dimension	LLaMA	Gemma	Qwen	Avg.
flattened adapter weights	$\sim 2 \cdot 10^7$	$-.035$	$-.060$	$-.034$	$-.043$
salient mask	$\sim 8 \cdot 10^9$	.133	.208	.229	.190
output sentence embeddings	384^*	.221	$-.053$	.096	.087
Delta Activations	4096	**.645**	**.545**	**.653**	**.614**

Table 2: **Clustering quality of different embedding spaces.** Delta Activations achieves the best separation across all backbones. *depends on sentence embedding models.

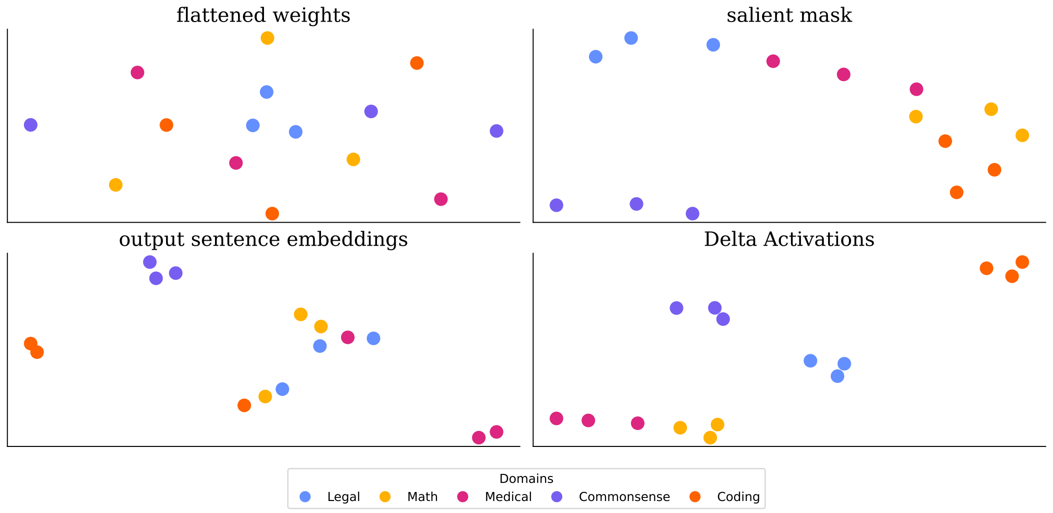


Figure 3: **t-SNE visualization of different embedding spaces.** Delta Activations form clean and well-separated domain clusters compared to baseline methods.

3.2 Understanding Delta Activations

In this section, we investigate properties of Delta Activations, analyze probe datasets and activation selection, and validate its stability over different training settings and finetuning regimes. Unless otherwise noted, we report average silhouette score across three backbone model pools: LLAMA-3.1-8B, GEMMA-2-9B, and QWEN-2.5-7B. For all tables, the main setting is marked in gray.

Delta-X variants. Our framework is not limited to activations, but can generalize to other model features extracted from the probe dataset. We create and evaluate two variants of Delta Activations: *Delta Logits*, and *Delta Meaning* [37] (differences in inverse perplexity scores over sampled continuations, implementation details in Appendix A.5). Results are shown in Table 3. Both these variants achieve reasonable clustering quality on our main experiment setting.

Method	Dimensionality	silh. score
Delta Logits	125856	.51
Delta Meaning	20	.20
Delta Activations	4096	**.61**

Table 3: **Delta-X.** Both variants achieve positive silhouette score in our main experiment setting, showing that our framework can be generalized to other representation extraction methods.

Additive property. A function $f(x)$ is additive if $f(a+b) = f(a) + f(b)$ for any inputs a and b. We explore whether Delta Activations exhibits this property by examining whether the following holds.

$$v\big(\text{model finetuned on } D_1 \cup D_2\big) \approx v\big(\text{model finetuned on } D_1\big) + v\big(\text{model finetuned on } D_2\big)$$

where $v(\cdot)$ is the operation to take Delta Activations. We finetune models on pairs of domains and comparing their Delta Activations to those from individually trained models. Results are presented in Table 4, which shows that the similarity between the mixed model and the sum of Delta Activations from the two individual models is consistently higher than the similarity with either individual model. This suggests that Delta Activations exhibit the additive property–combining shifts from separately finetuned models approximates the shift seen when trained on combined data. The additive structure in the embedding space is especially important since models are often trained on mixed datasets, ensuring that the embedding space preserves the influence of each component. We report results across all ten domain pairs in Appendix 13, where the additive effect consistently holds.

Math	Commonsense	Code	Mixed vs. D1	Mixed vs. D2	Mixed vs. Sum(D1, D2)
✓	✓		.58	.48	**.65**
✓		✓	.70	.27	**.73**
	✓	✓	.63	.28	**.65**

Table 4: **Additive property.** The sum of Delta Activations on models finetuned separately on two datasets aligns well with Delta Activations on the model finetuned on two datasets mixed together.

Probe dataset. We study the effects of number, length, and content of prompts in Table 5. In (a), we see that using multiple prompts instead of one helps stabilize the embedding, while increasing the number from 5 to 20 offers no additional benefits. In (b), we see that using GPT-4o to generate shorter versions of the Alpaca template, namely one-word and one-sentence versions, performs worse than a reasonably long instruction template. All prompts used in this part are included in Appendix C. Finally, (c) shows the importance of a generic instruction template by comparing the instruction template with domain-specific prompts or Wikitext. Domain-specific prompts perform worst since they suppress the model's specialization and thereby cause model embeddings to become less distinguishable. A random generic text sampled from Wikitext performs slightly better while the instruction template achieves the best separation.

These findings across prompt number, length, and content bolster our design choices for the probe dataset. In addition, even with variations in these prompt settings, the effectiveness of Delta Activations is preserved, as evident from the fact that the silhouette scores stay well above zero.

Where to extract activations. The embedding of the last token at the last layer is generally understood to encode the entire context. Decoder-only LLMs project this embedding to the logit space for next-

# of prompts	silh. score
1	.57
5	**.61**
20	**.61**

(a) number of prompts

length	silh. score
one-word	.45
one-sentence	.59
Alpaca (3-sentence)	**.61**

(b) length of prompts

Content	silh. score
Wikitext	.44
domain-specific	.42
instruction	**.61**

(c) content of prompts

Table 5: **Effects of number, length, and content of probe prompts.** Using multiple reasonably-long generic instruction templates makes the best probe dataset.

token prediction. Consequently, Delta Activations are also derived using this embedding. Here we study whether Delta Activations could instead be sourced from other tokens or different layers. In Table 6a, we examine the effectiveness of calculating Delta Activations using the first, middle, and last tokens, as well as from the weighted average of all token embeddings following Muennighoff [45]. Overall, the results show that final tokens are effective targets for calculating Delta Activations.

We also investigate whether the final layer is the best position from which to extract activations. As shown in Table 6b, shallower layers perform worse than deeper ones; interestingly, the final layer is not optimal, as a layer at $2/3$ of the total depth performs best. This phenomenon, where intermediate representation are found to be more effective for downstream tasks, is also observed in vision encoders [4, 2]. Although a layer at $2/3$ depth and weighted tokens exhibit slightly superior results, the final token at the last layer performs similarly. For simplicity, we use the last-layer final token embedding as the default setting for Delta Activations.

Token Position	silh. score
first	.22
mid	.39
last	**.61**
weighted avg. of all tokens	**.64**

(a) token position

Layer Position	silh. score
shallow (1/3 depth)	.51
mid (1/2 depth)	**.61**
deep (2/3 depth)	**.64**
last	**.61**

(b) layer position

Table 6: **Effects of token and layer position to extract activations.** Later tokens and deeper layeres produce better Delta Activations, with the $2/3$ depth layer slightly surpassing the last layer.

Robustness to training settings. Do differences in training settings have a greater impact on Delta Activations than the choice of finetuned domains? To evaluate this, we systematically perturbed the training process for models within our domain clusters. In our main setting, the model pool is organized into 5 distinct domain clusters, with 3 models in each cluster trained using identical settings. To test the impact of a specific training configuration—for instance, learning rate—the three models within each of the 5 domain clusters were trained using three different learning rates respectively (e.g., model 1 with $1e^{-4}$, model 2 with $4e^{-4}$, model 3 with $1e^{-5}$ within a single domain cluster). This process was independently repeated for variations in the number of training examples and the number of training epochs, where we vary number of training examples by 100, 1000, and 10000 and number of epochs by 1, 2, and 3.

Table 7 presents results which measure the clustering quality on domains when subjected to such training variations. indicates that varied training settings generally did not break domain-specific clustering. Changes to the amount of training data or the number of epochs had minimal effect on the quality of these clusters, which remained comparable to those formed under identical training settings. The different-learning-rate setting yields a lower silhouette score, as expected, since learning rate significantly impacts training dynamics and tends to increase within-cluster variation. These observations confirm that Delta Activations effectively identify finetuning domains despite common variations in training procedures. On the other hand, these results also show the strength of Delta Activations in identifying the nuanced differences within each cluster.

Beyond domains: clustering by dataset properties. In our setting in Section 3.1, each domain corresponds to a well-defined task with relatively uniform answer structure (e.g., multiple-choice). Here, we ablate that structure. We construct a new model pool by finetuning 3 models on each of five distinct subsets of `Tulu v2` [24]: CoT, GPT4-ALPACA, SHAREGPT, CODEALPACA, and SCIENCE.

Training Setting	LLaMA	Gemma	Qwen	Avg.
Different number of training examples	.66	.51	.68	.62
Different learning rates	.53	.37	.23	.38
Different training epochs	.62	.59	.51	.57
Identical training settings	.65	.55	.65	.61

Table 7: **Delta Activations' embeddings are robust to training hyperparameters.** Models trained in varying settings still form tight domain-specific clusters, comparable to those trained identically.

Unlike domain datasets, these are less semantically disjoint. Instead, they differ in instruction format, conversational structure, and expected output style—ranging from open-ended dialogue to multi-step chain-of-thought reasoning, code snippets, and longform factual answers. Crucially, this makes the output distribution far more diverse and less predictable. Results are presented in Table 8. With no consistent answer template, output sentence embeddings fail to reflect model specialization for LLaMA and Qwen, whereas Delta Activations continue to achieve decent clustering quality.

Embedding Space	LLaMA	Gemma	Qwen	Avg.
output sentence embeddings	.06	.23	$-.03$	.08
Delta Activations	**.33**	**.41**	**.48**	**.41**

Table 8: **Clustering quality (silhouette score) across Tulu v2 instruction splits.** Delta Activations remain effective despite diverse output structures and blurred instruction boundaries.

Beyond SFT: clustering by preference optimization. Our experiments thus far focused on the setting of Supervised Finetuning which use the token-level cross entropy loss. Preference optimization techniques [52, 42] maximize the likelihood that preferred responses are ranked higher. The different supervision signal of preference optimization affects the activation differently. We explore whether Delta Activations still yield reliable clustering for models in this case. To construct the model pool, we perform preference optimization on LLAMA-3.1-8B-INSTRUCT using three disjoint 3000-example splits for each of three preference optimization datasets, namely UltraFeedback [9], HumanLLM [3], and MetaMath-DPO [50, 72]. This experiment yields a silhouette score of **0.93**, which shows that Delta Activations can effectively capture similarity for preference optimization.

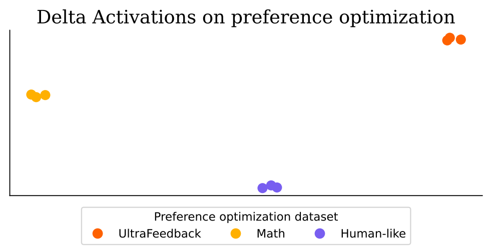


Figure 4: **Preference optimization.** Delta Activations can cluster models trained with DPO.

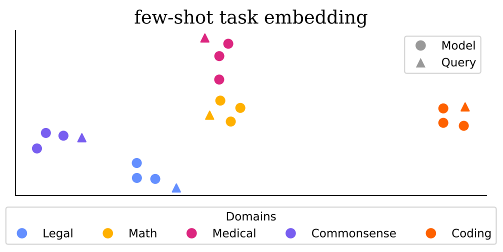


Figure 5: **Task embedding.** Few-shot task embedding is able to locate model clusters.

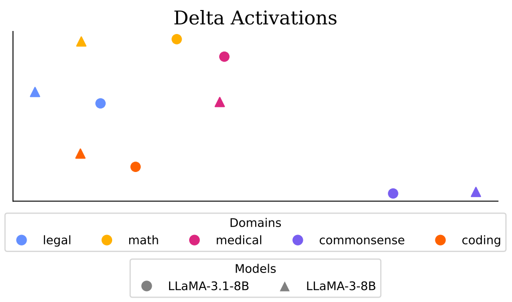
3.3 Extensions

In this section, we explore how Delta Activations as a model embedding can be extended to embedding a task, representing LLMs finetuned from different base LLMs, and guiding model merging.

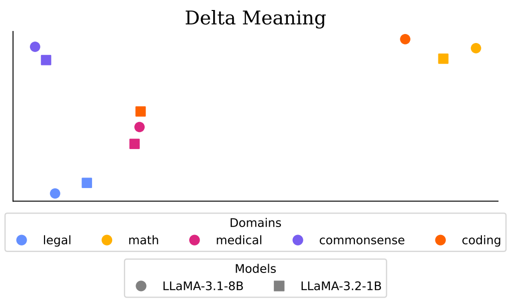
Task embedding. We explore whether Delta Activations can embed tasks using only a few examples—a setting analogous to few-shot generalization. For each of the five domains in Section 3.1, we finetune the base LLM on 20 held-out examples that were not part of any previous training split. The detailed few-shot finetuning setting is in Appendix A.4. We then embed the few-shot finetuned models using Delta Activations, and use this embedding as a representation of the task.

We examine whether the task embedding can successfully locate the corresponding clusters on the three model pools constructed in Section 3.1. We define the *retrieval rate* metric as the fraction of few-shot task embeddings that correctly retrieve their corresponding full-model cluster via nearest-neighbour search using cosine similarity. Gemma achieves 100% retrieval rate while there is one failure case for each of LLaMA and Qwen. We present visualization of Gemma in Figure 5, which shows that few-shot queries reliably align with their corresponding full model clusters (circles). Despite being trained on only 20 examples, the resulting Delta Activations recover the domain cluster. This suggests that Delta Activations on few-shot trained model can be an effective task embedding.

Cross-base-model clustering. Delta Activations represent a finetuned model as the difference between its hidden states and those of its base model on the same inputs; therefore, they can only be directly applied to models derived from the *same* base. Interestingly, we find that this delta signal can transfer across bases. To test this, we evaluate two settings: cross-checkpoint and cross-architecture. In a cross-checkpoint setting (LLaMA-3-8B vs. LLaMA-3.1-8B; 10 models over 5 domains), Delta Activations achieved a silhouette score of **0.39**, cleanly recovering the five domain-specialization clusters (Figure 6a). In a cross-architecture setting (LLaMA-3.1-8B vs. LLaMA-3.2-1B; 10 models), Delta Activations are no longer feasible because the embedding dimension differs across architectures, projecting models into vectors of incompatible sizes. Instead, we adopt *Delta Meaning* (full implementation details in Appendix A.5), which is architecture-agnostic, and it successfully forms four out of five domain clusters with a silhouette score of **0.32** (Figure 6b).



(a) cross-checkpoint



(b) cross-architecture

Figure 6: **Cross-base-model clustering.** (a) Delta Activations form domain clusters across models finetuned from LLaMA-3.1-8B and LLaMA-3-8B. (b) Delta Meaning form domain clusters across finetuned models of different sizes, LLaMA-3.1-8B and LLaMA-3.2-1B.

Model selection. LoraHub [21] hosts $\sim$ 200 finetuned models based on FLAN-T5. The method is evaluated on the Big-Bench Hard (BBH) [59]. For each task in BBH, LoraHub randomly select 20 LoRAs and optimizes their merging coefficients over few-shot examples from the target task. Our approach can be easily applied to this scenario by embedding the corresponding task using the provided few-shot examples and leverage Delta Activations similarity to replace the random model selection. Specifically, we identify the single most-related LoRA model as an anchor and samples the remaining 19 models randomly. This simple selection strategy enabled by Delta Activations yields an average performance improvement of 2.0% by boosting the average accuracy from 34.3% to 36.3% on the 26 tasks of the BBH benchmark. Interestingly, selecting 20 most similar models using Delta Activations yield an average performance of 30.3%, which is far lower than random selection. We conjecture that this is due to model interference identified by Ortiz-Jimenez et al. [47], where similar models are entangled with each other, resulting in bad merging performance. However, this in turn shows that Delta Activations are informative about the relationships between models. While model merging is beyond the scope of our work, Delta Activations enable model merging strategies beyond nearest-neighbour selection. Our proof-of-concept experiment focused on simple selection strategies, but Delta Activations could be leveraged for more sophisticated approaches. For instance, one could use Delta Activations to identify a maximally dispersed subset of models to mitigate interference or train a neural network on top of Delta Activations embeddings to optimize merging decisions.

4 Related Work

Embedding Models and Tasks. Ilharco et al. [23] discovers that task vectors from finetuning can be merged to achieve multi-task learning. There are several attempts to represent LLMs using adapter weights [48, 60], evaluation profiles [79], or training-data-dependent characteristics [77], but these methods rely on potentially inaccessible data or fail to reflect internal behavior. Our work differs by requiring no external supervision or evaluation, instead deriving embeddings from internal behaviors.

Activations. Recent work uncovers structure in hidden activations of LLMs, understanding how massive activation act as biases to steer LLM output [57, 10]. Activations are also central to post-training compression: they are used to compute weight saliency for pruning [12, 56, 71] and to reduce quantization error using calibration sets [13, 34, 76], which inspire the use of the probe dataset of Delta Activations. Additionally, learned activation shifts have been used to edit or transfer behavior across models [36, 32, 39].

Utilizing trained models. The landscape evolves from reusing a single model to multiple trained models. Finetuning [78] is a common framework to build on top of a single pretrained model, which becomes the common practice for LLMs [6, 52]. Other works use outputs [18, 63] or weights [31, 16] of a pretrained model for the creation or initialization [68, 67] of more efficient models. Efforts are made to effectively leverage multiple trained models through retrieval [77, 26, 40, 28], composition [70, 5, 21, 11], or routing [38, 46, 55].

Buiding model hubs. Recent works [19, 73] study the public model pool and can systematically uncover finetuning relationship among trained models. Lorahub [21], LoraRetriever [77], and Learnware [60] create model hubs involving from ~ 40 to ~ 200 models. It is also possible to create larger model hubs via neural network parameter diffusion [65, 33], though achieving true diversity [75] may require further development.

5 Discussion

Delta Activations provide a simple yet powerful way to represent finetuned LLMs by measuring shifts in their internal activations relative to a base LLM. Our experiments show that this representation consistently forms distinct clusters that reflect finetuning domains and offer the advantage of an additive property that mirrors multi-domain behavior. The stability of Delta Activations across varying finetuning settings shows its reliability for use-cases of model selection and merging in model hubs. We believe that Delta Activations can serve as a cornerstone for navigating the expanding landscape of finetuned models by enabling more efficient model discovery and reuse.

Limitations and future work. Delta Activations introduces a novel way to represent finetuned models but also poses practical considerations. Our experiments focused on three prominent open-source backbones but further evaluation on other architectures would be valuable to understand its broader applicability. In addition, Delta Activations require access to internal hidden states, which is not feasible to be evaluated on proprietary models.

It is natural to ask how our method might perform on model pools substantially larger than those considered in our evaluation. Such a practical exercise would be most meaningful if the pool consists of models with diverse and uniquely valuable capabilities. While this is an interesting direction we intend to explore, we conjecture that today such pools exist primarily in proprietary settings (e.g. finetuned GPT models), and we hope our approach could facilitate sharing such models in future.

References

[1] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022.

[2] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Daniel Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network. *arXiv:2504.13181*, 2025.

[3] Ethem Yağız Çalık and Talha Rüzgar Akkuş. Enhancing human-like responses in large language models. *arXiv preprint arXiv:2501.05032*, 2025.

[4] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020.

[5] Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. Adaptersoup: Weight averaging to improve generalization of pretrained language models. In *ACL*, 2023.

[6] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 2024.

[7] 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.

[8] CodeChef. Codechef. `https://www.codechef.com`, 2009. Competitive programming platform.

[9] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. Ultrafeedback: Boosting language models with scaled ai feedback. In *ICML*, 2024.

[10] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, pp. 30318–30332, 2022.

[11] Shangbin Feng, Zifeng Wang, Yike Wang, Sayna Ebrahimi, Hamid Palangi, Lesly Miculicich, Achin Kulshrestha, Nathalie Rauschmayr, Yejin Choi, Yulia Tsvetkov, et al. Model swarms: Collaborative search to adapt llm experts via swarm intelligence. *arXiv preprint arXiv:2410.11163*, 2024.

[12] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337, 2023.

[13] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[14] FreedomIntelligence. Disease database. `https://huggingface.co/datasets/FreedomIntelligence/Disease_Database`, 2024. Hugging Face dataset.

[15] Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, Jessica Wu, Joe Nudell, Joel Niklaus, John Nay, Jonathan H. Choi, Kevin Tobia, Margaret Hagan, Megan Ma, Michael Livermore, Nikon Rasumov-Rahe, Nils Holzenberger, Noam Kolt, Peter Henderson, Sean Rehaag, Sharad Goel, Shang Gao, Spencer Williams, Sunny Gandhi, Tom Zur, Varun Iyer, and Zehua Li. Legalbench: A collaboratively built benchmark for measuring legal reasoning in large language models, 2023.

[16] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NeurIPS*, 2015.

[17] Yifei He, Yuzheng Hu, Yong Lin, Tong Zhang, and Han Zhao. Localize-and-stitch: Efficient model merging via sparse task arithmetic. *Transactions on Machine Learning Research*, 2024, 2024. Preprint available at arXiv:2408.13656.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[19] Eliahu Horwitz, Asaf Shul, and Yedid Hoshen. Unsupervised model tree heritage recovery. In *ICLR*, 2025.

[20] 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. In *ICLR*, 2022.

[21] Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *COLM*, 2024.

[22] Siming Huang, Tianhao Cheng, Jason Klein Liu, Jiaran Hao, Liuyihan Song, Yang Xu, J Yang, JH Liu, Chenchen Zhang, Linzheng Chai, et al. Opencoder: The open cookbook for top-tier code large language models. *arXiv preprint arXiv:2411.04905*, 2024.

[23] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *ICLR*, 2023.

[24] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

[25] Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *arXiv preprint arXiv:2009.13081*, 2020.

[26] Pengfei Jin, Peng Shu, Sekeun Kim, Qing Xiao, Sifan Song, Cheng Chen, Tianming Liu, Xiang Li, and Quanzheng Li. Retrieval instead of fine-tuning: A retrieval-based parameter ensemble for zero-shot learning. *arXiv preprint arXiv:2410.09908*, 2024.

[27] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. In *EMNLP-IJCNLP*, 2019.

[28] Jonathan Kahana, Or Nathan, Eliahu Horwitz, and Yedid Hoshen. Can this model also recognize dogs? zero-shot model search from weights. *arXiv preprint arXiv:2502.09619*, 2025.

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[31] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. In *NeurIPS*, 1989.

[32] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. *ICLR*, 2023.

[33] Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, et al. Drag-and-drop llms: Zero-shot prompt-to-weights. *arXiv preprint arXiv:2506.16406*, 2025.

[34] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *MlSys*, 2023.

[35] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[36] Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023.

[37] Tian Yu Liu, Matthew Trager, Alessandro Achille, Pramuditha Perera, Luca Zancato, and Stefano Soatto. Meaning representations from trajectories in autoregressive models. In *ICLR*, 2024.

[38] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. *arXiv preprint arXiv:2311.08692*, 2023.

[39] Jinqi Luo, Tianjiao Ding, Kwan Ho Ryan Chan, Darshan Thaker, Aditya Chattopadhyay, Chris Callison-Burch, and René Vidal. Pace: Parsimonious concept engineering for large language models. In *NeurIPS*, 2024.

[40] Michael Luo, Justin Wong, Brandon Trabucco, Yanping Huang, Joseph E Gonzalez, Ruslan Salakhutdinov, Ion Stoica, et al. Stylus: Automatic adapter selection for diffusion models. In *NeurIPS*, 2024.

[41] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *ICLR*, 2024.

[42] Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. In *NeurIPS*, 2024.

[43] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, 2013.

[44] Mohamed-Ahmed161. Disease-symptoms dataset. https://huggingface.co/datasets/Mohamed-Ahmed161/Disease-Symptoms, 2024. Hugging Face dataset.

[45] Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*, 2022.

[46] Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Waleed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *ICLR*, 2024.

[47] Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. Task arithmetic in the tangent space: Improved editing of pre-trained models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *NeurIPS*, 2023.

[48] Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Matheus Pereira, Lucas Caccia, and Alessandro Sordoni. Towards modular llms by building and reusing a library of loras. *arXiv preprint arXiv:2405.11157*, 2024.

[49] Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, 2022.

[50] Arka Pal, Deep Karkhanis, Samuel Dooley, Manley Roberts, Siddartha Naidu, and Colin White. Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*, 2024.

[51] Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, Zekun Wang, Jian Yang, Zeyu Cui, Yang Fan, Yichang Zhang, Binyuan Hui, and Junyang Lin. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings, 2025.

[52] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *NeurIPS*, 2023.

[53] Yi Ren and Danica J Sutherland. Learning dynamics of llm finetuning. In *ICLR*, 2025.

[54] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 1987.

[55] Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

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

[57] Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive activations in large language models. *COLM*, 2024.

[58] Mingjie Sun, Yida Yin, Zhiqiu Xu, J. Zico Kolter, and Zhuang Liu. Idiosyncrasies in large language models. In *ICML*, 2025.

[59] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022.

[60] Zhi-Hao Tan, Zi-Chen Zhao, Hao-Yu Shi, Xin-Yu Zhang, Peng Tan, Yang Yu, and Zhi-Hua Zhou. Learnware of language models: Specialized small language models can do big. *arXiv preprint arXiv:2505.13425*, 2025.

[61] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*, 2023.

[62] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

[63] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1910.10699*, 2019.

[64] Hugo Touvron et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[65] Kai Wang, Dongwen Tang, Boya Zeng, Yida Yin, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. Neural network diffusion. *arXiv preprint arXiv:2402.13144*, 2024.

[66] Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *ACL*, 2021.

[67] Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared LLaMA: Accelerating language model pre-training via structured pruning. In *ICLR*, 2024.

[68] Zhiqiu Xu, Yanjie Chen, Kirill Vishniakov, Yida Yin, Zhiqiang Shen, Trevor Darrell, Lingjie Liu, and Zhuang Liu. Initializing models with larger ones. In *ICLR*, 2024.

[69] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[70] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.

[71] Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Mykola Pechenizkiy, Yi Liang, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (owl): A missing secret sauce for pruning llms to high sparsity. *arXiv preprint arXiv:2310.05175*, 2023.

[72] Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

[73] Runpeng Yu and Xinchao Wang. Neural phylogeny: Fine-tuning relationship detection among neural networks. In *ICLR*, 2025.

[74] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *ACL*, 2019.

[75] Boya Zeng, Yida Yin, Zhiqiu Xu, and Zhuang Liu. Generative modeling of weights: Generalization or memorization? *arXiv preprint arXiv:2506.07998*, 2025.

[76] Yuxin Zhang, Lirui Zhao, Mingbao Lin, Yunyun Sun, Yiwu Yao, Xingjia Han, Jared Tanner, Shiwei Liu, and Rongrong Ji. Dynamic sparse no training: Training-free fine-tuning for sparse llms. *arXiv preprint arXiv:2310.08915*, 2023.

[77] Ziyu Zhao, Leilei Gan, Guoyin Wang, Wangchunshu Zhou, Hongxia Yang, Kun Kuang, and Fei Wu. LoraRetriever: Input-aware LoRA retrieval and composition for mixed tasks in the wild. In *ACL Findings*, 2024.

[78] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 2020.

[79] Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning compact representations of large language models. In *ICLR*, 2025.

A Training settings

For LoRA, we set rank $r = 8$, $\alpha = 16$, targeting query, key, value, and MLP projections (`q_proj`, `k_proj`, `v_proj`, `up_proj`, `down_proj`, `gate_proj`), with no dropout and no bias parameters.

A.1 Datasets

Dataset	Description
OpenCoder-LLM	Educational programming instructions (opc-sft-stage2)
GSM8K	Grade-school math problems with chain-of-thought reasoning
HellaSwag	Commonsense natural language inference
LegalBench	Privacy policy question answering
PubMedQA	Medical Q&A from PubMed abstracts

Table 9: **Summary of datasets used in the experiments.** Each dataset was split into three subsets of 3,000 examples.

A.2 Prompt Templates

Prompt Type	Template (truncated)
Task-specific (Programming)	`Below is an instruction that describes a programming task...`
Task-specific (Math)	`Below is a grade-school math problem.  Please work through the reasoning step-by-step...`
Task-specific (Legal)	`Below is a legal-reasoning task from the LegalBench benchmark...`
Task-specific (Medical)	`Below is a medical question based on a PubMed article...`
Task-specific (Commonsense)	`Below is a scenario.  What happens next in this paragraph...`
Universal	`Below is an instruction that describes a task. Write a response that appropriately completes the request.`

Table 10: **Prompt templates used during training.** Task-specific templates are customized to each domain; the universal prompt is applied uniformly across all tasks during finetuning.

A.3 Training Hyperparameters

Training Setting	Configuration
Epochs	3
Batch size	4
Learning rate	1e-4
Optimizer	paged_adamw_32bit
Gradient accumulation steps	2
Warmup steps	10
Max sequence length	512 tokens (8,192 tokens for OpenCoder-LLM)
Hardware	NVIDIA H100 (80GB)

Table 11: **Training hyperparameters used across all experiments.**

Data was collated using the `DataCollatorForCompletionOnlyLM` from the TRL library, computing loss only on the response portion. We deploy all models to the Hugging Face Hub with standardized nomenclature indicating the base model, dataset, training approach, and key hyperparameters.

A.4 Few-shot Task Embedding Configuration

For the few-shot task embedding experiments described in Section 3.3, we used a configuration tailored for limited data scenarios. We randomly sampled 20 examples from each domain's held-out data that was not used in any of our main experiment splits.

Parameter	**Value**
Examples per domain	20 (randomly sampled)
Learning rate	3.3e-3
Batch size	1
Epochs	5

Table 12: Hyperparameters for few-shot task embedding experiments.

We maintained the same LoRA configuration as in our main experiments. The higher learning rate and increased number of epochs compensate for the limited training data while the smaller batch size allows for more frequent parameter updates. After fine-tuning, we computed Delta Activations using identical probe datasets as in our main experiments to ensure direct comparability between few-shot task embeddings and full model embeddings in the embedding space.

A.5 Delta Meaning implementation

Here we provide implementation details for *Delta Meaning*, our adaptation of Meaning Representations [37] to the Delta framework. This extension enables model embeddings across heterogeneous backbones, where direct activation comparisons are infeasible.

Meaning representations. Given a probe prompt x, we first sample n continuations $\{s_1, \dots, s_n\}$ from the base model (temperature = 1.0). For any finetuned model f, we then score each continuation s_i by computing its inverse perplexity under f:

$$m_f(x)_i = \exp\left(-\frac{1}{|s_i|} \sum_{t=1}^{|s_i|} \log p_f(s_{i,t} \mid s_{i,<t}, x) \right).$$

This produces an n-dimensional "meaning vector" $m_f(x) \in \mathbb{R}^n$ for each prompt x.

Delta aggregation. For a finetuned model f and its base f_{base}, we define the *Delta Meaning* on prompt x as the difference between their meaning vectors:

$$\Delta_f(x) = m_f(x) - m_{f_{\text{base}}}(x).$$

Aggregating across all prompts $x \in D_{\text{probe}}$ yields the final model embedding:

$$v_f = \frac{1}{|D_{\text{probe}}|} \sum_{x \in D_{\text{probe}}} \Delta_f(x).$$

Hyperparameters. In our experiments, we set n to be 20. Larger n provides more informative embeddings but requires proportionally more forward passes, as each continuation must be scored by both the base and finetuned models. Despite this, the dimensionality of Delta Meaning remains extremely compact compared to weight- or logit-based alternatives. Importantly, because any model can evaluate the probability of a given text sequence, Delta Meaning embeddings are naturally architecture-agnostic, allowing us to cluster finetuned models drawn from multiple backbones.

B Additional Analysis

B.1 t-SNE Visualization on more backbones

We show visualization of the experiments conducted in Section 3.1 on LLaMA and Qwen in Figure 7 and Figure 8 respectively. Over different backbones, the visualization consistently shows the superiority of Delta Activations in forming cleanly-separated clusters.

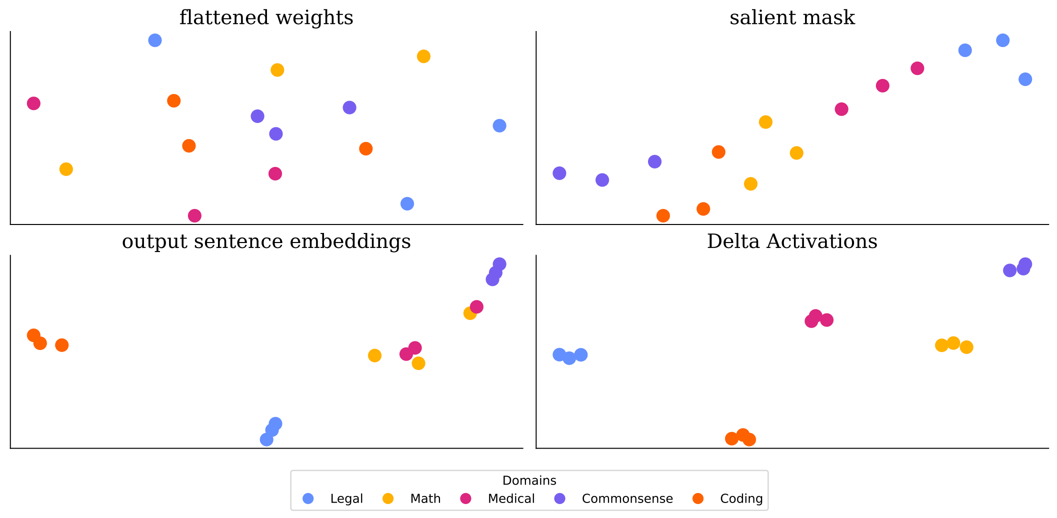


Figure 7: **t-SNE visualization of different embedding spaces (LLaMA).**

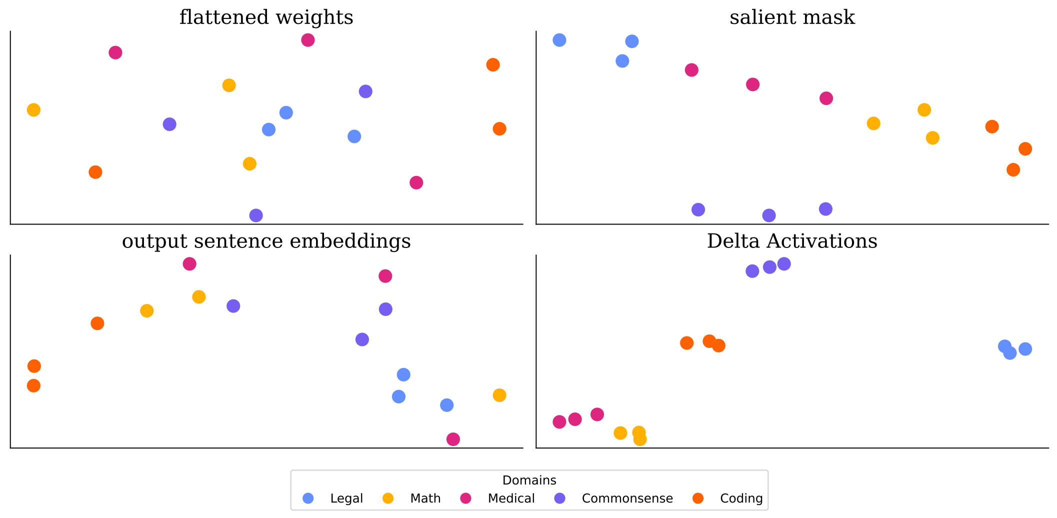


Figure 8: **t-SNE visualization of different embedding spaces (Qwen).**

B.2 Full Results for Additive Properties experiment

To better understand the additive nature of Delta Activations, we extend the experiment from Section 3.1 to cover all ten domain pairs. In each case, we compare the Delta Activation vector from a model trained on the mixed dataset against those from models trained individually on each domain, as well as their vector sum. This comprehensive table demonstrates that the summed Delta Activations consistently better approximate the mixed-model embedding, reinforcing the additive property of Delta Activations.

Math	Commonsense	Code	Medical	Legal	Mixed v. D1	Mixed v. D2	Mixed v. Sum
✓		✓			.703	.270	**.726**
✓	✓				.577	.484	**.649**
	✓	✓			.631	.283	**.653**
			✓	✓	.407	.675	**.695**
	✓		✓		.359	.662	**.677**
✓			✓		.760	.697	**.811**
		✓	✓		.462	.581	**.693**
	✓			✓	.649	.659	**.763**
✓				✓	.522	.610	**.669**
		✓		✓	.445	.507	**.587**

Table 13: **Additive property (full).** The sum of Delta Activations on models finetuned separately on two datasets aligns well with Delta Activations on the model finetuned on two datasets mixed together.

B.3 Full finetuning

We conduct experiments in Section 3.1 on LLaMA-3.1-8B with the model pool trained using full finetuning instead of LoRA. This experiment yields a silhouette score of **0.63**, which confirms that Delta Activations provide clear clustering regardless of finetuning methods. Visualization is shown in Figure 9.

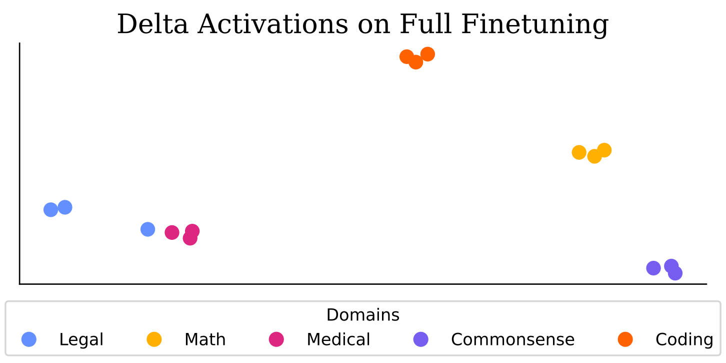


Figure 9: **t-SNE visualization of full finetuning.**

Intra-domain clustering. We conduct experiments to test whether Delta Activations provide sufficient signal to differentiate models trained on different sub-expertises within medical and coding domains.

We partition each domain into multiple sub-expertise splits. For medical, we create finetuned 8 models: disease databases [14] (2 splits), disease-symptoms [44] (2 splits), MedQA-USMLE [25] (2 splits), and MedMCQA [49] (2 splits). For coding, we create 6 models covering C++ (2 splits), Python (2 splits), and Java (2 splits) from CodeChef [8], Codeforces [51], and Evol-Instruct [41] datasets. We compute Delta Activations using both generic instruction prompts and domain-specialized prompts.

Table 14 shows that generic prompts achieve a silhouette score of 0.657 for medical sub-expertise, outperforming specialized medical prompts (0.533). For coding, both prompts perform comparably (0.668 vs 0.674). The generic probe maintains sufficient resolution to form correct clusters of sub-expertises within domains.

Domain	**Generic prompt**	**Specialized prompt**
Medical	**.657**	.533
Coding	.668	**.674**

Table 14: **Intra-domain clustering.** Silhouette scores for sub-expertise clustering within domains.

C Prompt Templates

Table 15, Table 16, and Table 17 list all prompt templates used in probe dataset experiments in Section 3.2. For other experiments, we use the first five prompts in Table 15.

ID	Prompt Template
1	Below is an instruction that describes a task. Write a response that appropriately completes the request.
2	The task described below requires a response that completes the request accurately.
3	Below is a description of a task. Provide a response that aligns with the requirements.
4	The following instruction outlines a task. Generate a response that meets the specified request.
5	You are given an instruction and input. Write a response that completes the task as requested.
6	You are provided with a task instruction and input. Write a response that fulfills the described requirements.
7	Here is an instruction and its associated input. Complete the task with an appropriate response.
8	Below is a task along with its context. Write a response that matches the requirements.
9	The following is a description of a task and its input. Generate a response that fulfills the request.
10	An outlined task is provided along with its input. Write a response that satisfies the given instruction.
11	Given the following instruction, generate a suitable response that fulfills the request.
12	The task described below requires a response that completes the request accurately.
13	Below is a description of a task. Provide a response that aligns with the requirements.
14	The following instruction outlines a task. Generate a response that meets the specified request.
15	You are given an instruction and input. Write a response that completes the task as requested.
16	Here is an instruction and its associated input. Create a response that properly addresses the request.
17	Below is a task description. Provide an appropriate response that matches the input.
18	An instruction and input are provided. Write a response that accurately completes the task.
19	The following is an instruction that describes a task. Write a response that correctly satisfies the request.
20	Below is an outlined task. Respond with a completion that fits the instruction and input given.

Table 15: List of paraphrased prompt templates used in our experiments.

ID	Prompt Template
1	Instruction: Please provide a response. Input: Input.
2	Please perform the following task.
3	Complete the instruction.
4	Provide the appropriate response.
5	Here is the text. Response:

Table 16: One-sentence paraphrased prompt templates.

ID	Prompt Template
1	Response:
2	Answer:
3	Explanation:
4	Solution:
5	Discussion:

Table 17: One-word paraphrased prompt templates.

D Broader Impact

The proposed Delta Activations method facilitate efficient reuse of fine-tuned models by providing an embedding to encode the finetuned model's behaviors and capability. This reduces redundant training, cutting energy costs and promoting sustainable AI practices. Furthermore, it encourages broader public sharing of fine-tuned models by offering clear documentation of their capabilities, accelerating research and collaboration.

However, expanding public model hubs also introduces risks, as low-quality or adversarial models could contaminate the pool. This highlights the need for careful curation to maintain reliability and safety in open model ecosystems.