Neural Incompatibility: The Unbridgeable Gap of Cross-Scale Parametric Knowledge Transfer in Large Language Models

Yuqiao Tan[1,2], **Shizhu He**[1,2] *, **Kang Liu**[1,2,3], **Jun Zhao**[1,2]
[1] The Key Laboratory of Cognition and Decision Intelligence for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] Shanghai Artificial Intelligence Laboratory
tanyuqiao2025@ia.ac.cn {shizhu.he, jzhao, kliu}@nlpr.ia.ac.cn

Abstract

Large Language Models (LLMs) offer a transparent brain with accessible parameters that encode extensive knowledge, which can be analyzed, located and transferred. Consequently, a key research challenge is to transcend traditional knowledge transfer paradigms rooted in symbolic language and achieve genuine Parametric Knowledge Transfer (PKT). Significantly, exploring effective methods for transferring knowledge across LLMs of different scales through parameters presents an intriguing and valuable research direction. In this paper, we first demonstrate **Alignment** in parametric space is the fundamental prerequisite to achieve successful cross-scale PKT. We redefine the previously explored knowledge transfer as Post-Align PKT (PostPKT), which utilizes extracted parameters for LoRA initialization and requires subsequent fine-tune for alignment. Hence, to reduce cost for further fine-tuning, we introduce a novel Pre-Align PKT (PrePKT) paradigm and propose a solution called **LaTen** (**La**te-**T**hen-**Al**ig**n**) that aligns the parametric spaces of LLMs across scales only using several training steps without following training. Comprehensive experiments on four benchmarks demonstrate that both PostPKT and PrePKT face challenges in achieving consistently stable transfer. Through in-depth analysis, we identify **Neural Incompatibility** as the ethological and parametric structural differences between LLMs of varying scales, presenting fundamental challenges to achieving effective PKT. These findings provide fresh insights into the parametric architectures of LLMs and highlight promising directions for future research on efficient PKT. Our code is available at https://github.com/Trae1ounG/Neural_Incompatibility.

1 Introduction

Human beings have non-transparent thoughts without inherited memory, requiring them to learn through communication in language-based settings. Based on this language-based knowledge transfer paradigm (Figure 1(a)), Large Language Models (LLMs) have acquired a wealth of knowledge and abilities to understand and solve general tasks, with a massive amount of knowledge encoded in their parameters during pretraining on an extensive corpus (Achiam et al., 2023; Brown et al., 2020; Ouyang et al., 2022). However, unlike the unknowable and opaque nature of the human brain, the accessible parameters and information flow of LLMs (e.g. Llama (Touvron et al., 2023)) function as a transparent brain that directly encodes factual knowledge, which can be systematically analyzed, precisely located and effectively transferred.

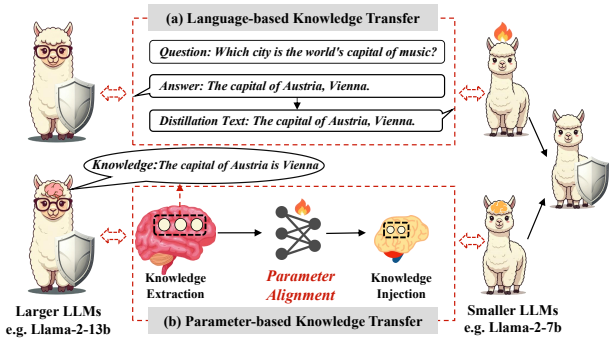

Figure 1: Different paradigms of knowledge transfer between cross-scale LLMs. Compared to human-like symbolic knowledge transfer based on language (as shown in (a)), we aspire for LLMs to achieve more efficient knowledge transfer leveraging knowledgeable parameters (as illustrated in (b)).

Existing studies (Geva et al., 2020; Wang et al., 2022; Yu and Ananiadou, 2024) have made significant progress in interpreting knowledge localization and information flow in the transparent brain of LLMs, allowing possible knowledge manipulation. Model merging (Wortsman et al., 2022; Matena and Raffel, 2022; Yu et al., 2024) combines models with different capabilities to create a multitask capable model by merging weights. Although this

*Corresponding author

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 21586–21601
July 27 - August 1, 2025

approach performs parametric knowledge merge, it is limited to models of the same scale and the same checkpoint. In real-world scenarios, it is more common for larger and smaller LLMs to form natural paired. Compared to smaller LLMs M_s, larger LLMs M_l encapsulate more world knowledge due to they contain larger parameter scales and extended training processes, making them intuitively appear as more extensively trained versions of smaller LLMs. Building on this insight, the key research question becomes: *Can knowledge be effectively transferred from larger LLMs to smaller ones through parameters?*

We define this challenge as **Parametric Knowledge Transfer** (**PKT**) in cross-scale LLMs. Then, We first demonstrate through simple experiments that achieving **Alignment** in the parametric space is a prerequisite for successful cross-scale PKT. Notably, PKT comprises three key stages: 1) **Knowledge Extraction**, which extracts task-related knowledge from M_l; 2) **Parameter Alignment**, which aligns the extracted knowledge with M_s; and 3) **Knowledge Injection**, which performs the final parameters integration. Existing research on PKT performs alignment after knowledge injection by further training, which we refer to this paradigm as Post-Align PKT (PostPKT). PostPKT uses the extracted parameters to initialize certain modules (e.g. Low-Rank Adaptation (LoRA) (Hu et al., 2021)) for injection while holding the overall parameter unchanged. SEEKING (Zhong et al., 2023) addresses PostPKT by employing sensitivity-based knowledge location combined with LoRA-driven injection, followed by additional fine-tuning for alignment. Despite achieving improved performance after training for several epochs on 1,000 examples, this approach incurs high alignment costs. As a result, it is not only expensive but also unable to directly enhance the model's performance.

To reduce cost for further fine-tuning, we introduce a novel paradigm of PKT, which takes parameter alignment before injection called Pre-Align PKT (PrePKT). In this paradigm, we aspire to directly enhance LLMs ability after injection. To achieve this goal, we propose **LaTen** (**La**te-**Th**en-Alig**n**) to facilitate the alignment of parametric spaces in LLMs across different scales (Figure 1(b)). Specifically, LaTen uses neuron-level attribution (Yu and Ananiadou, 2024) to address discrepancies in layer number and identify the most informative neurons for transfer in both feed-forward networks (FFNs) and multi-head self-attention (MHSA) modules. To perform dimensionality reduction, we use a simple MLP-based hypernetwork, which learns to map the parameter space of $\mathbf{\Theta}_l$ to $\mathbf{\Theta}_s$ by training on a small subset for alignment (< 100). By decoding just one seed sample with the larger model, task-related parameters can be identified and projected into the target parametric space, enabling immediate improvements in downstream task performance.

Our experiments focus on PostPKT and PrePKT in three benchmark categories: world knowledge, mathematical reasoning, and code generation, using Llama-2-based models (Touvron et al., 2023). For PostPKT, we compare SEEKING with PiSSA (Meng et al., 2024) which decomposes the original parameters of $\mathbf{\Theta}_s$ to derive LoRA. Our results show that LoRA parameters derived from larger M_l are less effective compared to those derived from the model M_s itself. Concurrently, although our proposed LaTen demonstrates promising performance, it still faces challenges in achieving consistently stable PrePKT. Through in-depth analysis, we identify **Neural Incompatibility** as the ethological and parametric structural differences between cross-scale LLMs which are similar to the cross-species neural mechanism (Lu et al., 2024; Wang et al., 2025), presenting fundamental challenges to achieve optimal parametric knowledge transfer. These findings offer novel insights into the parametric structures of LLMs and suggest directions for future research on efficient PKT. Our main contributions are summarized as follows:

- We are the first to comprehensively define and explore parametric knowledge transfer between cross-scale LLMs.
- We identify the importance of alignment and systematically study parametric knowledge transfer from Pre-Align and Post-Align paradigms.
- We propose a novel method Locate-Then-Align to first try to solve Pre-Align challenge, which leverages neuron attribution and hypernetwork techniques to execute alignment with minimal training data achieves promissing performance.
- Comprehensive quantitative and qualitative assessments have highlighted the neural incompatibility as a key challenge arising from ethological and parametric structural differences in cross-scale LLMs.

2 Related Work

2.1 Location of Parametric Knowledge

Large language models (LLMs) encode vast amounts of knowledge in their parameter space through pre-training on large-scale corpora. Consequently, numerous studies have focused on identifying where knowledge is stored in language models, particularly in "neurons" (Song et al., 2024; Tang et al., 2024; Niu et al., 2024; Chen et al., 2024a,c). Dai et al. (2021) first introduced the term "knowledge neuron" referring to the specific medium within the model that stores knowledge. Their work demonstrated that the factual knowledge encoded in a model's parameters could be modified by manipulating these neurons. Building on this, Meng et al. (2022a) refined the process of identifying knowledge in LLMs using causal tracing, showing that FFN layers in the middle blocks of the model are critical for encoding factual knowledge. To address the computational overhead of these methods, Yu and Ananiadou (2024) proposed a static attribution approach inspired by the logit lens (nostalgebraist, 2020), enabling the identification of important neurons with reduced computational and memory requirements. Based on these advancements, we adopt neurons as the fundamental units for our work and apply a modified neuron attribution method adapted from (Yu and Ananiadou, 2024) to achieve knowledge extraction.

2.2 Manipulation of Parametric Knowledge

With the recognition of how knowledge stored in model parameters, exist research has sought to execute diverse operations on these parameters, aiming to manipulate the implicit knowledge. Knowledge editing aims to update the parameters related to specific knowledge in a model without affecting its other capabilities (De Cao et al., 2021; Mitchell et al., 2021; Meng et al., 2022a,b). We categorize editing as fine-grained knowledge manipulation, whereas its counterpart, coarse-grained manipulation, encompasses tasks such as model merging (Jin et al., 2022; Yu et al., 2024; Bowen et al., 2024). Model merging typically starts with models of the same scale, or even identical checkpoints, to combine multiple models with different capabilities into a single, multitask model. A series of studies regard the delta parameter as task vector (Ilharco et al., 2022; Zhang et al., 2023; Huang et al., 2024), which can be leveraged through arithmetic operations such as addition and subtraction to acquire or forget specific skills, demonstrating strong generalization capabilities. DyPRAG (Tan et al., 2025) transforms symbolic documents into parametric knowledge using a hypernetwork. However, these approaches are either limited to individual models or require maintaining the same scale and even identical checkpoints. Moreover, they do not explore how parametric knowledge can be transferred across models of different scales.

2.3 Transfer of Parametric Knowledge

Large language models can achieve explicit knowledge transfer through language or logits, known as knowledge distillation (Hinton, 2015). However, this approach overlooks the rich parametric knowledge encoded within the model's weights. Existing methods for parametric knowledge transfer primarily focus on model merging, directly combining parameters from models of same scale. However, little attention has been given to transferring knowledge across models of different scales. While inference-time proxy-tuning techniques (Liu et al., 2024; Wu et al., 2024) allow smaller models to influence larger ones by adjusting output logits, they do not directly leverage the implicit parametric knowledge. Recent work SEEKING (Zhong et al., 2023) has explored knowledge transfer between models of different scales. It leverages sensitivity-based extraction and LoRA-driven injection, followed by Post-Alignment using training. In this study, we conduct a comprehensive analysis of both PostPKT and our proposed PrePKT to investigate strategies for achieving efficient and effective PKT.

3 Challenge of Pre-Align Parametric Knowledge Transfer

SEEKING (Zhong et al., 2023), designed to address Post-align PKT, leverages delta parameters $\Delta\Theta$ extracted from M_l for LoRA initialization, demonstrating advantages over random initialization. Since the overall parameters remain unchanged after injection, we refer to the subsequent fine-tuning process as **post-alignment**. This process, which aligns the LoRA parameters from M_l with the original ones, plays a critical role in enhancing PostPKT. However, the post-alignment process is expensive and requires training on large-scale data. A more intuitive and cost-effective approach is to directly inject a certain form of $\boldsymbol{\Delta\Theta}$ into the original parameters, thereby immediately enhancing task-specific knowledge without

the need for additional training. In this section, we try to solve this without alignment to determine its significance in this new context. We adopt several straightforward and commonly used methods. However, we find that all of these unaligned transfer methods fail. Through analysis, we identify key statistical factors contributing to this failure, underscoring the critical importance of **pre-alignment** in our newly proposed Pre-Align PKT paradigm.

3.1 Analysis Setup

We first illustrate how Transformer (Vaswani, 2017) works and denote several symbols, detailed in Appendix A.1. Due to the parametric space mismatch between M_l and M_s, the achievement of knowledge transfer hinges on discrepancy dimension and layer number problem. For example, chat version of Llama-2-7b has a layer number L of 32, hidden dimension d of 4096 and FFN neuron numbers N of 11008, while 40, 5120 and 13824 for chat version of Llama-2-13b, respectively. To reach the conclusion, we employ several basic approaches, relying solely on pre-selected layers and standard dimension reduction techniques. For layer selection, we propose three methods: TOP-L_s, BOTTOM-L_s and RANDOM-L_s. Given a smaller LLM M_s with L_s layers, these methods select the top, bottom, or random L_s layers from M_l, respectively. To match hidden dimension d_l and d_s, we use standard dimensionality reduction techniques PCA (Abdi and Williams, 2010), WHITENING (Su, 2021) and a learning-based EMBEDDING TRANSFORM. For the numbers of FFN neuron N_l and N_s, RANDOM and IMPORTANCE are selected. Meanwhile, we consider sensitivity-based knowledge localization method SEEKING (Zhong et al., 2023) as the strongest among the evaluated baselines. Notably, we directly using the extracted $\Delta\mathbf{\Theta}_{\text{extract}}$ as delta parameters, i.e. $\mathbf{\Theta}'_s = \mathbf{\Theta}_s + \Delta\mathbf{\Theta}_{\text{extract}}$. Details on these methods can be found in Appendix B.

3.2 Results of Unaligned Baselines

The results are presented in Figure 2 using MMLU benchmark (Hendrycks et al., 2021). The Llama-2-13b-Chat and Llama-2-7b-Chat models score 52.90 and 44.20, respectively. All unaligned transfer methods significantly impair the model's ability to perform specific tasks, with some cases resulting in nearly zero functionality. For instance, when using the IMPORTANCE method for neuron selection and the EMBEDDING TRANSFORM for dimensionality reduction, several approaches achieve a score of

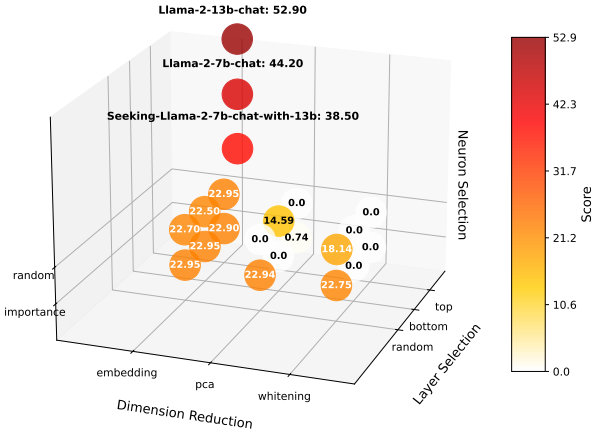

Figure 2: Performance of different baseline methods in MMLU.

22.95 with a degradation of 21.25. Notably, SEEKING also causes a significant performance drop of 5.70 compared to the original model.

We begin our statistical analysis of the delta parameter ranges and observe that the delta parameters from regular supervised fine-tuning (SFT) exhibit a distinct pattern compared to those from the SEEKING method. Specifically, the delta parameter ranges for normal SFT are smaller (within 0.002, as shown in Figures 7 and 8 aligns with (Yu et al., 2024)), while those from SEEKING are larger (exceeding 0.005 in Figure 9), with some even surpassing 1. Consequently, directly injecting these large range parameters can significantly alter the parameter distribution (e.g. potentially flipping θ_i from positive to negative).

Furthermore, we find that unaligned delta parameters exhibit low parametric similarity to the original weights, indicating an absence of useful task-related information in LoRA initialization. In contrast, well-aligned SFT delta parameters demonstrate significantly higher similarity to the original weights. We conduct further in-depth analysis in Section 5.3. Based on these observations, we conclude that alignment is a crucial factor for achieving PKT. Therefore, implementing pre-alignment in the parametric space during PrePKT presents a significant challenge.

4 LaTen: Locate-Then-Align for Pre-Align Parametric Knowledge Transfer

Based on the observations and conclusions in Section 3, we propose a novel Locate-Then-Align

(LaTen) method to solve PrePKT and achieve parametric space alignment, while preventing to disrupt the injected model and without additional training. Specifically, we utilize static neuron-level attribution (Yu and Ananiadou, 2024) method to locate knowledgeable parameters for transferring, according to both FFN and MHSA neurons contain task-related parametric knowledge (Geva et al., 2023; Chen et al., 2024b). The neuron-level attribution method is detailed in Appendix A.2.

4.1 Parametric Knowledge Transfer Definition

Considering a smaller LLM M_s and a larger LLM M_l, initially parameterized by $\mathbf{\Theta}_s$ and $\mathbf{\Theta}_l$, respectively. For a specific task $\mathcal{T}$, corresponding to a training dataset $D^{\mathcal{T}}_{\text{train}} = \{(x_i^{\mathcal{T}}, y_i^{\mathcal{T}})\}_{i=1}^{Q}$ comprising Q input-output instances and a extract dataset $D^{\mathcal{T}}_{\text{extract}}$ and an alignment dataset $D^{\mathcal{T}}_{\text{align}}$. The overall goal is to extract delta parameters $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}}$ from $\mathbf{\Theta}_l$ based on $D^{\mathcal{T}}_{\text{extract}}$, then align to $\mathbf{\Theta}_s$ to obtain $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{algin}}$:

$$\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}} = \text{Extract}(\mathbf{\Theta}_l; \mathbf{\Theta}_s; D^{\mathcal{T}}_{\text{extract}}) \quad (1)$$

$$\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{align}} = \text{Align}(\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extarct}}; D^{\mathcal{T}}_{\text{align}} \text{ or } D^{\mathcal{T}}_{\text{train}}) \quad (2)$$

where $\text{Extract}(\cdot)$ representing the logic for parameter extraction and $\text{Align}(\cdot)$ encapsulating the alignment process for cross-scale PKT. Notably, $D^{\mathcal{T}}_{\text{align}}$ only uses in PrePKT, while PostPKE trains for alignment on larger $D^{\mathcal{T}}_{\text{train}}$. Knowledge injection will execute to merge the delta parameters into Θ_s:

$$\mathbf{\Theta}^{\mathcal{T}}_s = \begin{cases} \mathbf{\Theta}_s - \Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}} + \mathbf{BA}, & \text{for PostPKT} \\ \mathbf{\Theta}_s + \Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{align}}, & \text{for PrePKT} \end{cases} \quad (3)$$

where in PostPKT, the LoRA $\mathbf{BA}$ matrices are initialized using $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}}$ by SVD (Golub and Reinsch, 1971), and the injection is performed before the $\text{Align}(\cdot)$ while keeping the overall weight unchanged. In contrast, our proposed PrePKT first performs $\text{Align}(\cdot)$ and then executes injection. By incorporating the well-aligned parameters, the model ability on task $\mathcal{T}$ is directly enhanced.

4.2 Neuron-level Localization for Knowledge Extraction

Existing studies have demonstrated that neurons in FFN and MHSA of Transformer serve as fundamental units for storing knowledge or specific skills (Geva et al., 2020; Dai et al., 2021). These neurons can be leveraged to modify the model's behavior, such as editing (Meng et al., 2022a,b) in single model, which have not been fully explored in a couple of models of different scales.

We use the neuron-level attribution method (Yu and Ananiadou, 2024) to locate task-related useful neurons for knowledge extraction. As detailed in Appendix A.2, we use this method to gain importance score of each neuron vector $\mathbf{v}$, denoted as $Imp(\mathbf{v})$, can be computed by measuring the change in the output distribution of a predicted token t. Since an answer y typically consists of T tokens, we only choose the last useful token t_T for attribution score. This process produces a score matrix $\mathbf{S}$ with dimensions $\mathbf{S}^{\text{FFN}} \in \mathbb{R}^{L\times N}$ for the FFN and $\mathbf{S}^{\text{MHSA}} \in \mathbb{R}^{L\times d}$ for the MHSA, where each element $\mathbf{S}_{i,j}$ represents the importance score of neuron $\mathbf{v}$ in the ith layer at the jth position. To perform layer selection, we sum $\mathbf{S}_{i,j}$ over layers to get $\mathbf{S}^{\text{FFN}}_{\text{layer}}, \mathbf{S}^{\text{MHSA}}_{\text{layer}} \in \mathbb{R}^{L}$ then choosing the top-L_s layers for M_l. To align with neuron number of M_s, we select the top-c neurons I_l in layer l by sorting $\mathbf{S}_{l,\cdot}$ in descending order, where c equals N_s or d_s for FFN or MHSA, repectively. Based on I_l, we then extract the corresponding key and value neurons to obtain the unaligned delta parameters $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}} \in \mathbb{R}^{N_s\times d_l}$ for FFN or $\mathbb{R}^{d_s\times d_l}$ for MHSA.

4.3 Parameter Alignment with Knowledge Injection

$\text{Align}(\cdot)$ as a prerequisite before knowledge injection. Given the absence of explicit parameter-level correspondence between M_s and M_l, we employ traditional language modeling loss for alignment optimization. Our training framework operates as follows: based on each training instance $(x_i^{\mathcal{T}}, y_i^{\mathcal{T}}) \in D^{\mathcal{T}}_{\text{extract}}$, we first extract the raw delta parameters $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}}$. These parameters are then processed through a lightweight hypernetwork (implemented as a two-layer MLP with ReLU activation) to obtain dimension reduced parameters $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{align}}$. The target model parameters are subsequently injected via $\mathbf{\Theta}^{\mathcal{T}}_s = \mathbf{\Theta}_s + \Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{align}}$. During alignment, we randomly sample P instances from the alignment dataset $D^{\mathcal{T}}_{\text{algin}}$ to compute the language modeling loss, updating the hypernetwork for effective parameter alignment. During evaluation, we randomly sample one unseen instance, process extracted parameters through trained hypernetwork for final alignment, and apply the aligned parameters to the target model.

Models	MMLU	GSM8K	HumanEval	MBPP
Llama-2-7B-Chat	44.20	16.07	14.05	17.80
Llama-2-13B-Chat	52.90	20.55	18.75	19.20
# Post-Align Parametric Knowledge Transfer				
Post-Align on $D^{\mathcal{T}}_{\text{train}}$ (=1000):				
-Random Initialization.	49.73	26.51	14.22	15.60
-Seeking + 13B Param.	49.60	28.23	15.44	20.60
-PiSSA Initialization.	**49.77**	**29.32**	**16.26**	**21.40**
Post-Align on $D^{\mathcal{T}}_{\text{align}}$ (< 100):				
-Random Initialization.	44.20	16.35	14.02	18.20
-Seeking + 13B Param.	44.20	14.78	14.63	18.60
# Pre-Align Parametric Knowledge Transfer				
-Seeking + Unaligned 13B Param wo Train.	38.50	7.28	0.61	0.00
Pre-Align on $D^{\mathcal{T}}_{\text{align}}$ (< 100):				
-LaTen + Pre-Aligned 13B Param.	**44.40**	**20.47**	**14.63**	18.20

Table 1: Results of Post-Align and Pre-Align parametric knowledge transfer.

5 Experiments

5.1 Experimental Setup

Datasets and Pre-trained Backbones. We select MMLU (Hendrycks et al., 2021) to evaluate the professional knowledge of models. We choose GSM8K (Cobbe et al., 2021) for evaluating mathematical reasoning ability. For code generation, HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) are adopted for estimation. For pre-trained LLMs, we use Llama 2 (Touvron et al., 2023) to conduct task-related PKT.

Evaluation Metrics. We calculate zero-shot accuracy for GSM8K and MMLU, pass@1 for HumanEval and MBPP.

Implementation Details. Since chosen benchmark require instruction following ability which not include in base version, we use chat version in remain experiments. To execute PKT, we randomly sample three non-overlapping subsets from the original training dataset (we utilize python examples from (Luo et al., 2023b) as training set for HumanEval[1]): an extract set of size 32, an align set of size 80, and a training set of size 1000 (except MBPP). However, not all examples in the alignment set are used for alignment in LaTen. Details of the count are provided in Table 5. We denote the $D^{\mathcal{T}}_{\text{align}}$ as actual examples used for parameter alignment in the following section. Additional implementation details and experimental settings can be found in C.1.

[1] https://huggingface.co/datasets/nickrosh/Evol-Instruct-Code-80k-v1

5.2 Experimental Results

Main Results for Post-Align PKT. SEEKING is the first to investigate the transferability of cross-scale LLMs and demonstrated that using delta parameters as LoRA initialization can significantly enhance the ability of M_s to acquire task-specific knowledge. Since this paradigm keeps the overall parameters unchanged at the beginning, it can be considered as a variant of LoRA initialized from larger LLMs. Therefore, we are curious whether the optimal LoRA initialization originates from another model or from the model itself?

To explore this, we compare it with a self-derived LoRA approach: PiSSA (Meng et al., 2024), which applies singular value decomposition (SVD) to separate LoRA parameters and residual components from the original model parameters. As shown in the upper section of Table 1, SEEKING outperforms random Gaussian initialized LoRA in most scenarios. For instance, the SEEKING method achieves an improvement of 5.0 compared to Gaussian initialization in MBPP. However, SEEKING shows even lower performance in MMLU task, demonstrating its instability. Notably, PiSSA consistently achieves higher performance across all benchmarks compared to SEEKING, with the delta parameters derived from the same model providing an additional 0.72 boost in performance on average.

These results suggest that the SEEKING method, which relies on parameters from a different LLM, requires more effort but leads to suboptimal performance. We hypothesize that this is due to the

Models	HumanEval	MBPP
Llama-2-7B	14.05	17.80
-PiSSA Initialization	**16.26**	**21.40**
# Post-Align PKT from Llama-2-13B		
Llama-2-13B	18.75	19.20
-Seeking + 13B Param.	15.44	20.60
# Post-Align PKT from WizardCoder-13B-Python		
WizardCoder-13B-Python	56.71	41.60
-Seeking + 13B Param.	15.04	19.80
# Post-Align PKT from CodeLlama-13B-Python		
CodeLlama-13B-Python	47.56	37.80
-Seeking + 13B Param.	16.05	**21.40**

Table 2: Results of Post-Align PKT from different larger LLMs in code generation.

Models	HumanEval	MBPP
Llama-2-7B	14.05	17.80
# Pre-Align PKT from Llama-2-13B		
Llama-2-13B	18.75	19.20
–LaTen + 13B Param.	**14.63**	18.20
# Pre-Align PKT from WizardCoder-13B-Python		
WizardCoder-13B-Python	56.71	41.60
-LaTen + 13B Param.	14.02	**18.60**
# Pre-Align PKT from CodeLlama-13B-Python		
CodeLlama-13B-Python	47.56	37.80
-LaTen + 13B Param.	14.02	17.80

Table 3: Results of Pre-Align PKT from different larger LLMs in code generation.

incompatibility of the delta parameters from M_l compared to M_s itself.

Main Results for Pre-Align PKT. In Section 3, we explored various methods to extract unaligned $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{extract}}$ for direct application to the parameters. However, none of these methods yielded satisfactory results. Instead, they degraded the model original capabilities. We also show more comprehensive results in Table 1 to further demonstrate our finding of the importance of alignment.

As shown in the lower part of Table 1, our proposed LaTen achieves strong performance across all benchmarks. For instance, LaTen improves by 4.40 on GSM8K and achieves an average improvement of 1.86 across four datasets compared to the base model M_s. We also employ a more equitable setup to compare the PostPKT and PrePKT paradigms, where PostPKT is restricted to performing post-align operations solely on the align dataset $D^{\mathcal{T}}_{\text{align}}$ (<100). In this setting, LaTen outperforms models with random initialization. Additionally, when compared to the PostPKT SEEKING approach with the same computational cost for alignment, LaTen demonstrates superior performance in most settings, particularly on GSM8K, where it achieves an improvement of 5.69. These results highlight the potential of PrePKT paradigm and LaTen in effectively solving it.

It is worth emphasizing that our proposed LaTen achieves powerful performance with only a few steps of parameter alignment. However, this phenomenon is a double-edged sword. Identifying the best checkpoint requires multiple experiments, and the parameter space alignment process does not exhibit a straightforward minimum point, making it more challenging to optimize compared to traditional language-based transfer methods. Therefore, exploring stable PrePKT methods is valuable for future research.

5.3 Analysis

Can Stronger M_l Transfer Richer Knowledge? In the above experiment, we observed that using the parameters derived from larger LLMs M_l via SEEKING to initialize LoRA generally outperforms random initialization in most cases. However, it still falls short compared to using LoRA initialized with parameters derived from the model itself (Meng et al., 2024). To further investigate whether the parameters extracted from M_l carry useful information, an intuitive hypothesis is that parameters extracted from M_l which specialized in a specific task should be more useful for PKT.

To test this hypothesis, we employ WizardCoder-13B-Python (Luo et al., 2023b) and CodeLlama-13B-Python (Roziere et al., 2023) as M_l for comparison. Both WizardCoder and CodeLlama are fine-tuned on code-specific datasets to enhance their code generation capabilities, outperforming Llama-2-13B in this domain. However, as shown in Table 2, under identical fine-tuning conditions in PostPKT, initializing LoRA parameters with WizardCoder-13B unexpectedly resulted in worse task performance. For example, on the HumanEval (MBPP) task, performance decreased by 0.4 (0.8) compared to Llama-2-13B. Although CodeLlama-13B demonstrated improvements over Llama-2-13B, it still falls short of matching the performance of PiSSA. We also examine mathematical generation task in Table 6 which shows the similar results in PostPKT. For PrePKT, the results also do not support our hypothesis, shown in Table 3,

Based on the above experiments, we propose that LLMs of different parameter scales inherently exhibit incompatibility in parameters, which makes ideal PKT largely coincidental. We call this **Neuron Incompatibility** and further explore the rea-

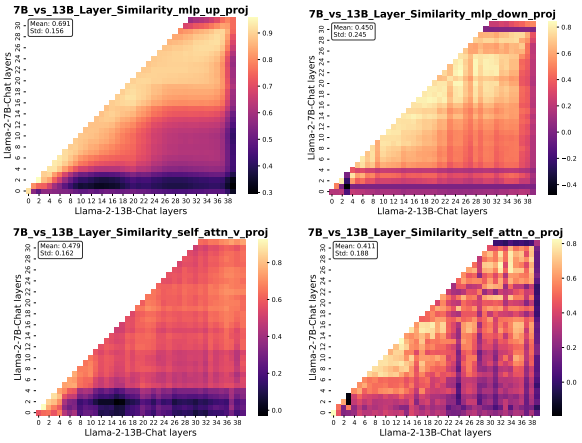

Figure 3: Representation Similarity Comparison Results between LLMs.

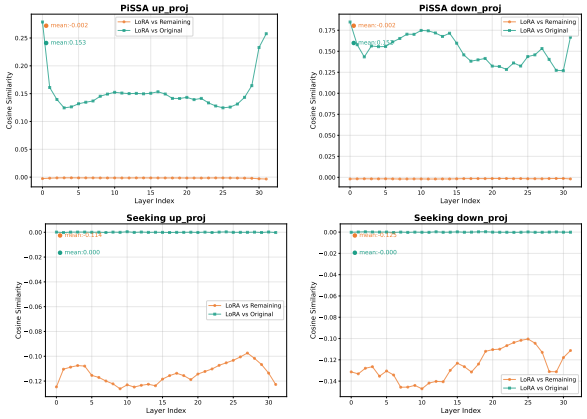

Figure 4: Parametric Similarity Comparison Results between LLMs in MLP Modules.

sons behind in following discussion.

Ethological Similarity between Cross-Scale LLMs. To further analyze why both PostPKT and PrePKT perform suboptimally, we utilize Centered Kernel Alignment (CKA) (Kornblith et al., 2019), a method based on the Hilbert-Schmidt Independence Criterion (HSIC), to compute the similarity between feature representations in neural networks. This metric assesses the similarity in behaviors between the two models, which can be interpreted as the ethological similarity of LLMs. We compute the ethological similarity between Llama-2-7B and Llama-2-13B across the up-proj, down-proj, v-proj, and o-proj modules.

As shown in Figure 3, the similarity between 7B and 13B is notably low, especially in the MHSA module which plays the most important part for integrating information (Elhage et al., 2021). Interestingly, the up-proj layers demonstrate higher similarity, likely because they function as key memories, capturing specific input patterns (Geva et al., 2020), which tend to be consistent across models. The weak similarity between M_l and M_s also explains why LoRA derived from the same model performs better, as it aligns more closely with the model's intrinsic behavior. We identify that the weak ethological similarity between cross-scale LLMs is one of the key factors contributing to neural incompatibility, making ideal parametric knowledge transfer success difficult to achieve.

Parametric Structural Similarity in Different Methods. We further conduct in-depth analysis based on parametric structural similarity to figure out whether it performs important influence on performance. As shown in Figure 4, we compare $\mathbf{W}^l_{\text{LoRA}}$ (i.e. LoRA parameters in layer l) with both $\mathbf{W}^l$ and $\mathbf{W}^l_{\text{remain}}$ (i.e. $\mathbf{W}^l - \mathbf{W}^l_{\text{LoRA}}$) in up-proj and down-proj modules. First, the pattern of results is completely opposite between SEEKING and PiSSA. In SEEKING, the mean similarity between $\mathbf{W}^l$ and $\mathbf{W}^l_{\text{LoRA}}$ drops to 0, suggesting that $\mathbf{W}^l_{\text{LoRA}}$ does not retain any meaningful information from $\mathbf{W}^l$. This deficiency results in suboptimal performance. For comparison, PiSSA which utilizes SVD to capture important parameters for LoRA, preserves greater similarity to the original weights and establishes an orthogonal relationship with $\mathbf{W}^l_{\text{remain}}$, making it more effective for learning new skills. Our findings indicate that parametric structural similarity plays a crucial role in further fine-tuning. Specifically, the similarity between $\mathbf{W}^l_{\text{LoRA}}$ and $\mathbf{W}^l$ significantly influences the model's ability to adapt to new tasks and execute parameter alignment. The low degree of similarity emerges as a key factor contributing to neural incompatibility. We observe the same pattern in the MHSA module (Figure 6).

6 Conclusion

In this work, we comprehensively define and explore the feasibility of Parametric Knowledge Transfer (PKT) between cross-scale LLMs. We propose Locate-Then-Align solution to address newly raised Pre-Align PKT challenge with reduced alignment costs. Through extensive experiments on four benchmarks, we demonstrate the neural incompatibility between cross-scale LLMs reflects in low similarity in both ethological and parametric space, which pose fundamental challenges to achieve ideal PKT. Our findings offer novel insights into the parametric structures of LLMs and aim to elucidate directions for future research on efficient PKT.

7 Limitations

This study identifies alignment as the key factor for achieving PKT and introduces two distinct paradigms based on this insight. Although current PKT methods are somewhat effective, they still rely on language for supervision. Developing simpler and more efficient approaches that do not depend on language guidance is a promising direction for future research. Moreover, the underlying principles behind PKT's effectiveness remain unclear, and it is also uncertain why a stronger M_l yields no improvement. These questions highlight important areas for further investigation. Furthermore, due to equipment limitations, our experiments were restricted to models between 13B and 7B. Nevertheless, the results still provide meaningful evidence to support our conclusions. In the future, expanding experiments to larger-scale LLMs would be a worthwhile and necessary direction for exploration.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2022ZD0160503) and Beijing Natural Science Foundation (L243006) and the National Natural Science Foundation of China (No.62376270).

References

Hervé Abdi and Lynne J Williams. 2010. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*.

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

Tian Bowen, Lai Songning, Wu Jiemin, Shuai Zhihao, Ge Shiming, and Yue Yutao. 2024. Beyond task vectors: Selective task arithmetic based on importance metrics. *arXiv preprint arXiv:2411.16139*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Lihu Chen, Adam Dejl, and Francesca Toni. 2024a. Analyzing key neurons in large language models. *arXiv preprint arXiv:2406.10868*.

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

Wei Chen, Zhen Huang, Liang Xie, Binbin Lin, Houqiang Li, Le Lu, Xinmei Tian, Deng Cai, Yonggang Zhang, Wenxiao Wan, et al. 2024b. From yes-men to truth-tellers: Addressing sycophancy in large language models with pinpoint tuning. *arXiv preprint arXiv:2409.01658*.

Yuheng Chen, Pengfei Cao, Yubo Chen, Kang Liu, and Jun Zhao. 2024c. Journey to the center of the knowledge neurons: Discoveries of language-independent knowledge neurons and degenerate knowledge neurons. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17817–17825.

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

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. *arXiv preprint arXiv:2104.08164*.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Gene H Golub and Christian Reinsch. 1971. Singular value decomposition and least squares solutions. In *Handbook for Automatic Computation: Volume II: Linear Algebra*, pages 134–151. Springer.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Geoffrey Hinton. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

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

Shih-Cheng Huang, Pin-Zu Li, Yu-Chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tsai, and Hung-Yi Lee. 2024. Chat vector: A simple approach to equip llms with instruction following and model alignment in new languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10943–10959.

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

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

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1317–1327.

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. *arXiv preprint arXiv:2011.05864*.

Huanxuan Liao, Yao Xu, Shizhu He, Yuanzhe Zhang, Yanchao Hao, Shengping Liu, Kang Liu, and Jun Zhao. 2024. From instance training to instruction learning: Task adapters generation from instructions. *arXiv preprint arXiv:2406.12382*.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Xiqian Lu, Zhaoqi Hu, Yumeng Xin, Tianshu Yang, Ying Wang, Peng Zhang, Ning Liu, and Yi Jiang. 2024. Detecting biological motion signals in human and monkey superior colliculus: a subcortical-cortical pathway for biological motion perception. *Nature Communications*, 15(1):9606.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

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

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

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

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2022b. Mass-editing memory in a transformer. *arXiv preprint arXiv:2210.07229*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Michael C Mozer and Paul Smolensky. 1988. Skeletonization: A technique for trimming the fat from a network via relevance assessment. *Advances in neural information processing systems*, 1.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? *arXiv preprint arXiv:2405.02421*.

nostalgebraist. 2020. interpreting gpt: the logit lens. *LessWrong*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

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

Ran Song, Shizhu He, Shuting Jiang, Yantuan Xian, Shengxiang Gao, Kang Liu, and Zhengtao Yu. 2024. Does large language model contain task-specific neurons? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7101–7113.

Jianlin Su. 2021. You probably don't need bert-flow: A linear transformation comparable to bert-flow.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. *Advances in neural information processing systems*, 28.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. 2025. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *arXiv preprint arXiv:2503.23895*.

Tianyi Tang, Wenyang Luo, Haoyang Huang, Dongdong Zhang, Xiaolei Wang, Xin Zhao, Furu Wei, and Ji-Rong Wen. 2024. Language-specific neurons: The key to multilingual capabilities in large language models. *arXiv preprint arXiv:2402.16438*.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. 2024. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *arXiv preprint arXiv:2404.02905*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems*, 33:12388–12401.

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

Yufan Wang, Luqi Cheng, Deying Li, Yuheng Lu, Changshuo Wang, Yaping Wang, Chaohong Gao, Haiyan Wang, Camilla T Erichsen, Wim Vanduffel, et al. 2025. The chimpanzee brainnetome atlas reveals distinct connectivity and gene expression profiles relative to humans. *The Innovation*, 6(2).

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

Jiayi Wu, Hao Sun, Hengyi Cai, Lixin Su, Shuaiqiang Wang, Dawei Yin, Xiang Li, and Ming Gao. 2024. Cross-model control: Improving multiple large language models in one-time training. *arXiv preprint arXiv:2410.17599*.

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

Zeping Yu and Sophia Ananiadou. 2024. Neuron-level knowledge attribution in large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3267–3280.

Jinghan Zhang, Junteng Liu, Junxian He, et al. 2023. Composing parameter-efficient modules with arithmetic operation. *Advances in Neural Information Processing Systems*, 36:12589–12610.

Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. 2023. Seeking neural nuggets: Knowledge transfer in large language models from a parametric perspective. *arXiv preprint arXiv:2310.11451*.

A Background

A.1 Transformer

Transformer-based language models (Vaswani, 2017) are at the center of state-of-the-art natural language processing (Devlin, 2018; Brown et al., 2020) and have become the most popular and effective architecture, even in computer vision (Dosovitskiy, 2020; Tian et al., 2024). A decoder-only Transformer is stacked with L identical blocks, mainly containing a multi-head self-attention (MHSA) module and a feed-forward network (FFN) module.

Follow (Yu and Ananiadou, 2024), We first detail the forward pass from the input token to the final prediction. Given an input sequence $X = [t_1, t_2, \ldots, t_T]$ with T tokens, the model generated the next token's probability distribution y over B tokens in vocabulary V. Each t_i at position i starts as a word embedding $\mathbf{h}_i^0 \in \mathbb{R}^d$ transformed by the embedding matrix $\mathbf{E} \in \mathbb{R}^{B\times d}$. Followed by L transformer layers, each layer output $\mathbf{h}_i^l$ (layer l, position i) is the sum of the previous layer's output $\mathbf{h}_i^{l-1}$, the FFN output $\mathbf{F}_i^l$ and the attention output $\mathbf{A}_i^l$:

$$\mathbf{h}_i^l = \mathbf{h}_i^{l-1} + \mathbf{F}_i^l + \mathbf{A}_i^l. \tag{4}$$

The final probability distribution $\mathbf{y}$ of the next token is computed by multiplying the unembedded matrix $\mathbf{E}_u \in \mathbb{R}^{B\times d}$ and the last position of Lth layer output:

$$\mathbf{y} = \text{softmax}(\mathbf{E}_u \mathbf{h}_T^L). \tag{5}$$

Diving into how do the two main components work, the FFN layer's output is computed by two linear transformations with a nonlinear function σ, while the MHSA layer's output is a weighted sum over H heads on T positions:

$$\mathbf{A}_i^l = \sum_{j=1}^{H} \text{ATTN}_j^l(\mathbf{h}_1^{l-1}, \mathbf{h}_2^{l-1}, \ldots, \mathbf{h}_T^{l-1}), \tag{6}$$

$$\mathbf{F}_i^l = \mathbf{W}_{\text{down}}^l \sigma(\mathbf{W}_{\text{up}}^l(\mathbf{h}_i^{l-1} + \mathbf{A}_i^l)), \tag{7}$$

where $\mathbf{W}_{\text{up}}^l \in \mathbb{R}^{N\times d}$ and $\mathbf{W}_{\text{down}}^l \in \mathbb{R}^{d\times N}$ are two linear matrices in FFN and for clarity we do not include $\mathbf{W}_{\text{gate}}^l$ used in LLama-2. (Geva et al., 2020) shows that FFN emulates neural memories (Sukhbaatar et al., 2015) where the $\mathbf{W}_{\text{up}}^l$ corresponds to keys and $\mathbf{W}_{\text{down}}^l$ to values. Then the FFN output can be transformed into a weighted sum of FFN neurons:

$$\mathbf{F}_i^l = \sum_{k=1}^{N} c_{i,k}^l \mathbf{down}_k^l, \tag{8}$$

$$c_{i,k}^l = \sigma(\mathbf{up}_k^l \cdot (\mathbf{h}_i^{l-1} + \mathbf{A}_i^l)), \tag{9}$$

where $\mathbf{down}_k^l$ denotes the kth column of $\mathbf{W}_{\text{down}}^l$, referred to as the FFN subvalue. The coefficient score $c_{i,k}^l$ is computed by comparing the residual output $\mathbf{h}_i^{l-1} + \mathbf{A}_i^l$ with $\mathbf{up}_k^l$, the kth row of $\mathbf{W}_{\text{up}}^l$, referred to as the FFN subkey. Meanwhile, the MHSA output $\mathbf{A}_i^l$ can also be expressed as the sum of individual head outputs, where each head produces a weighted sum of the value vectors across all positions:

$$\mathbf{A}_i^l = \sum_{j=1}^{H} \sum_{n=1}^{T} \alpha_{i,j,n}^l \mathbf{W}_{j,l}^o(\mathbf{W}_{j,l}^v \mathbf{h}_n^{l-1}) \tag{10}$$

$$\alpha_{i,j,n}^l = \text{softmax}(\mathbf{W}_{j,l}^q \mathbf{h}_i^{l-1} \cdot \mathbf{W}_{j,l}^k \mathbf{h}_n^{l-1}) \tag{11}$$

where $\mathbf{W}_{j,l}^q$, $\mathbf{W}_{j,l}^k$, $\mathbf{W}_{j,l}^v$, and $\mathbf{W}_{j,l}^o \in \mathbb{R}^{d\times d/H}$ represent the query, key, value, and output matrices of the jth attention head in the lth layer. The query and key matrices are used to compute the attention weight $\alpha_{i,j,n}^l$ for the nth position, followed by applying the softmax function across all positions. The value and output matrices then transform the input vector at the nth position into the corresponding value-output vector. Finally, the output of each attention head is the weighted sum of value-output vectors across all positions.

A.2 Neuron-level Attribution

Following the definition of neurons from (Yu and Ananiadou, 2024), the kth FFN neuron is the kth subvalue of $\mathbf{W}_{\text{down}}^l$ which is activated by its corresponding subkey $\mathbf{up}_k^l$. To align with the definition in FFN neurons, we regard the kth column of $\mathbf{W}_{j,l}^o$ as the kth attention subvalue (neuron) in this head, whose subkey is the k th row of $\mathbf{W}_{j,l}^v$. This design uses the position value-output $\mathbf{W}_{j,l}^o(\mathbf{W}_{j,l}^v \mathbf{h}_n^{l-1})$ as the base unit, performing an addition of $T \times H$ vectors to generate each attention output as shown in Eq.10. Each vector is derived from the attention subvalue and subkey, similar to Eq.8.

Instead of choosing integrated gradients (Sundararajan et al., 2017; Dai et al., 2021) or causal tracing (Vig et al., 2020; Meng et al., 2022a), we use a static neuron-level attribution method from (Yu and Ananiadou, 2024) . As introduced

above, the final vector $\mathbf{h}_T^L$ used for predicting the next token is computed as a direct sum of various neuron-level vectors. Specifically, $\mathbf{h}_T^l$ in layer l can be decomposed into two components: a single neuron vector $\mathbf{v}$ and the remaining vector $\mathbf{x} = \mathbf{h}_T^l - \mathbf{v}$. The change in the output distribution, measured using the "logits lens" (nostalgebraist, 2020), can then be interpreted as the importance score for the neuron $\mathbf{v}$:

$$Imp(\mathbf{v}^l) = \begin{cases} \log(p(w|\mathbf{v}^l + \mathbf{h}^{l-1})) \\ -\log(p(w|\mathbf{h}^{l-1})) \end{cases}, \mathbf{v}^l \in l\text{th MHSA}$$
$$\begin{cases} \log(p(w|\mathbf{v}^l + \mathbf{h}^{l-1} + \mathbf{A}^l)) \\ -\log(p(w|\mathbf{h}^{l-1} + \mathbf{A}^l)) \end{cases}, \mathbf{v}^l \in l\text{th FFN} \quad (12)$$

where w is the prediction token and $p(w|*)$ is computed by multiplying the vector with $\mathbf{E}_u$ (in Eq. 5).

Interpretable Neuron Location. Based on our modified method (Yu and Ananiadou, 2024), the most influential neurons identified can be analyzed through the logits lens (nostalgebraist, 2020). As illustrated in Figure 5, we project these selected neurons into the vocabulary space, revealing that the top-10 tokens are strongly associated with both the question and answer. This approach enables effective neuron selection, preserving the most relevant neuron information for the specific task and facilitating the subsequent transfer process.

Task: GSM8K	
Question: *Barry stands on his head for 10...turns can Barry take standing on his head during a single 2-hour period?* **Answer:** *Each turn Barry takes standing on his head... So the answer is 8.*	
Neuron	**Top10 tokens in vocabulary space**
f_{31}-8024	'2', '3', '1', '0', '4', '5', '6', '7', '8', '9'
a_{25}^3-100	'8', 'Jun', 'eight', 'Zero', 'eigh', 'middle', 'Friday', 'Middle', 'Jul', 's'
Question: *Edward the plumber ... how many washers will be remaining in the bag?* **Answer:** *If Edward needs to use 1 bolt per... So the answer is 4.*	
Neuron	**Top10 tokens in vocabulary space**
f_{28}-8918	'4', 'Four', 'four', 'four', 'fourth', '4', 'Vier', 'четы', '4', 'cuatro'
a_{25}^3-0	'four', 'four', 'Four', 'six', 'cuatro', 'vier', 'четы', 'nine', 'quatre', '四'

Figure 5: Interpretable neuron location in GSM8K task.

B Details of Parametric Knowledge Transfer Baselines

B.1 Unaligned Parametric Knowledge Transfer Baselines

In this section, we first detail the precise implementation of WHITENING, EMBEDDING TRANSFORM and IMPORTANCE, leaving SEEKING in Section B.2.

- WHITENING: BERT-whitening, introduced by (Su, 2021), provides a simple yet effective alternative to BERT-flow (Li et al., 2020). It has become a widely adopted technique for dimensionality reduction while preserving key features (Liao et al., 2024).

- EMBEDDING TRANSFORM: We propose an intuitive method for dimensionality reduction by learning a linear transformation $\mathbf{W}' \in \mathbb{R}^{d_l \times d_s}$ to map $\mathbf{E}_l$ to $\mathbf{E}_s$ using $\mathbf{E}_s = \mathbf{E}_l\mathbf{W}'$. Here, $\mathbf{E}_s \in \mathbb{R}^{B \times d_s}$ and $\mathbf{E}_l \in \mathbb{R}^{B \times d_l}$ represent the embedding matrices in M_s and M_l, respectively. This problem is formulated as the following minimization task:

$$\min_{\mathbf{W}'} \|\mathbf{E}_s - \mathbf{E}_l\mathbf{W}'\|_F^2 \quad (13)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. After derivation, we obtain the closed-form solution: $\mathbf{W}' = (\mathbf{E}_l^T\mathbf{E}_l)^{-1}\mathbf{E}_l^T\mathbf{E}_s$ which is then applied to perform dimensionality reduction effectively.

- IMPORTANCE: Following (Bowen et al., 2024), we define the importance $\mathbf{I}_i$ of the ith parameter θ_i using its amplitude, where $\mathbf{I}_i = \|\theta_i\|_2^2$. Neuron sampling is performed based on average importance scores across neurons, while gradient-based scores are utilized in SEEKING.

B.2 Illustrate SEEKING Method

(Zhong et al., 2023) attempts to empirically investigate post-align parametric knowledge transfer from larger to smaller models through parametric perspective. When conducting knowledge extraction, for a given task $\mathcal{T}$, the parameter-level importance score is calculated by sensitivity (Mozer and Smolensky, 1988):

$$\mathbf{S}_{i,j}^{\mathcal{T}} = \left|\theta_i^\top \nabla_{\theta_i} \mathcal{L}(x_j^{\mathcal{T}}, y_j^{\mathcal{T}}|\mathbf{\Theta})\right| \quad (14)$$

where $\mathbf{S}_{i,j}^{\mathcal{T}}$ represents the importance of the ith parameter θ_i relative to sample j and the absolute value is taken for the purpose of measuring the amplitude. Then, the final score $\mathbf{S}_i^{\mathcal{T}}$ for task $\mathcal{T}$ integrates the cumulative sensitivity over the sampled instances, calculated as $\sum_{j=1}^k \mathbf{S}_{i,j}^{\mathcal{T}}$. Layer selection involves calculating a score for each layer by aggregating the sensitivity scores of all parameters within that layer. The layers are then ranked in descending order based on the scores, and the top L_s layers are selected while preserving their original sequential order. Dimension reduction is

achieved by directly extracting the sub-matrix with the highest cumulative sensitivity score:

$$\mathbf{W}^{l}_{\text{extract}} = \arg\max_{\mathbf{W}' \subseteq \mathbf{W}^{l}} \sum_{\theta_i \in \mathbf{W}'} \mathbf{S}_i. \quad (15)$$

where $\mathbf{W}^{l} \in \mathbb{R}^{n_l \times m_l}$ represents a matrix in lth layer and $\mathbf{W}' \in \mathbb{R}^{n_s \times m_s}$ is a sub-matrix in $\mathbf{W}^{l}$ to match smaller model's matrix dimensions ($n_s \leq n_l, m_s \leq m_l$). Then aggregating $\mathbf{W}^{l}_{\text{extract}}$ across all layers can get the final extracted parameters $\Delta\mathbf{\Theta}_{\text{extract}}$.

For knowledge injection, SEEKING utilizes LoRA (Hu et al., 2021) as a bridge by decompose $\mathbf{W}^{l}_{\text{extract}}$ into $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{T}$ using Singular Value Decomposition (SVD) (Golub and Reinsch, 1971) then transfer to $\mathbf{U}[:, :r]\mathbf{\Sigma}[:r, :r]\mathbf{V}^{T}[:r, :]$ to match the rank r. In the end, SEEKING actually provides a unaligned LoRA initialization as:

$$\mathbf{W}^{l\star} = \mathbf{W}^{l} - \mathbf{W}^{l}_{\text{extract}} + \mathbf{BA} \quad (16)$$

where $\mathbf{B}$ is initialized as $\mathbf{U}[:, :r]\mathbf{\Sigma}[:r, :r]$, and $\mathbf{A}$ with $\mathbf{V}^{T}[:r, :]$. After injection, the overall parameters remain unchanged (the SVD approximation of the LoRA loss is negligible). Subsequent post-alignment with a large amount of training data is the key process.

B.3 Statistical Analysis

Range of Delta Parameters. (Yu et al., 2024) finds that the SFT delta parameter ranges at a very small number (with in 0.002) and contains many redundant information that can be removed. We first derive the delta parameters from the base and chat versions of Llama-2 (Touvron et al., 2023). The results are presented in Figures 7 and 8 which consistent with (Yu et al., 2024). However, using SEEKING (Zhong et al., 2023) to directly extract the delta parameters from M_l (e.g., Llama-2-13b) results in $\Delta\mathbf{\Theta}_{\text{extract}}$ values that are poorly aligned with M_s, leading to a wide range of parameter differences. As shown in Figure 9, these values often exceed 0.005, with a maximum of 1.12 and a minimum of -1.13. When applied directly as delta parameters, such discrepancies significantly degrade the model's performance.

C Detailed Experimens

C.1 Implementations

Baseline Implementations. During fine-tuning, the smaller model is trained for 5 epochs with a batch size of 64 and a learning rate of 3e-4 except for HumanEval of 3e-5 and trained 3 epochs in SFT setting. Regarding LoRA, we set the rank as 16, and insert LoRA into up-proj and down-proj of FFN, v-proj and o-proj of MHSA layer. During the alignment stage, the hypernetwork is trained with a learning rate of 1e-5 and a weight decay of 0.05. We also employ the Mean Square Loss between $\Delta\mathbf{\Theta}^{\mathcal{T}}_{\text{align}}$ and a zero tensor of the same size as a constraint. The sample size P is set to 16 and we only transfer 10% neurons in each layer. Notably, results of PostPKT are mean values from three runs with different seeds. More details about datasets are shown in Table 4 and 5. Our inference process for language models is handled based on vLLM (Kwon et al., 2023).

Stronger M_l Implementations. For stronger M_l, we choose model which further fine-tuned on Llama-2-13B to keep comparison. We choose WizardMath-13B-V1.0[2] for matheatical reasoning. We use WizardCoder-Python-13B[3] and CodeLlama-13b-Python[4] for code-generating task.

Dataset	MMLU	GSM8K	HumanEval	MBPP
Train Size	1000	1000	1000	300
lr	3e-4	3e-4	3e-5	3e-4
Epochs	5	5	3	5

Table 4: Details of training datasets in PostPKT.

Dataset	MMLU	GSM8K	HumanEval	MBPP
Align Size	32	64	48	128
lr	3e-5	3e-5	3e-5	3e-5
Steps	2	4	3	8

Table 5: Details of alignment datasets in PrePKT.

Representation Similarity Implementations. In order to calculate the representation similarity, we use PyTorch implementation of Centered Kernel Alignment (CKA)[5]. In practice, we randomly sample 200 examples from the test set of wikitext to calculate model representations.

C.2 More Results

Strong M_l in Mathematical Reasoning. We also use WizardMath-13B-V1.0 (Luo et al., 2023a) as

[2] https://huggingface.co/WizardLM/WizardMath-13B-V1.0

[3] https://huggingface.co/WizardLM/WizardCoder-Python-13B-V1.0

[4] https://huggingface.co/codellama/CodeLlama-13b-Python-hf

[5] https://github.com/RistoAle97/centered-kernel-alignment

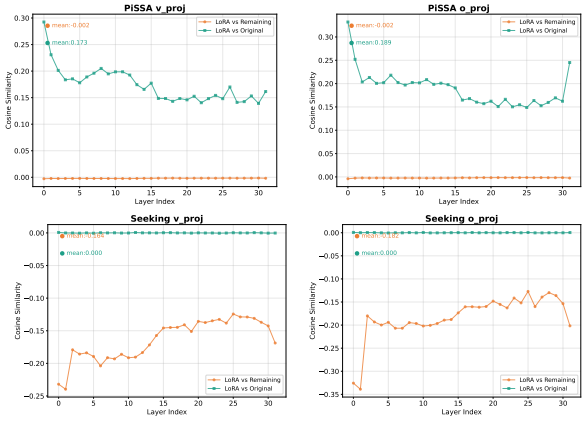

Figure 6: Results for Parametric Similarity Comparison between LLMs in MHSA Modules.

M_l for comparison. WizardMath leverages Reinforcement Learning from Evol-Instruct Feedback (RLEIF) to enhance its mathematical reasoning abilities and outperforms Llama-2-13B in this domain. However, as shown in Table 6, under the same fine-tuning settings, using WizardMath-13B as the parameter source for LoRA initialization unexpectedly led to worse task performance. For instance, on the GSM task, performance dropped by 3.71 compared to Llama-2-13B and even fell 1.99 below the randomly initialized LoRA baseline. This result supports our conclusion.

Models	GSM8K
Llama-2-7B	16.07
# Post-Align PKT from Llama-2-13B	
Llama-2-13B	20.55
-Seeking + 13B Param.	**28.23**
# Post-Align PKT from WizardMath-13B	
WizardMath-13B	53.15
-Seeking + 13B Param.	24.52

Table 6: Results for implicit parametric knowledge transfer from different larger LLMs in mathematical reasoning.

Parametric Similarity in MHSA Modules. As shown in Figure 6, we observe a similar pattern in the MHSA modules. This result indicates that the similarity between $\mathbf{W}^l$ and $\mathbf{W}^l_{\text{LoRA}}$ plays a crucial role in subsequent SFT.

Comparison with Language-based Knowledge Distillation. To further assess the effectiveness of LaTen in low-resource settings, we expanded our experiments to include comparisons between Llama-2-7B-Chat and Llama-2-13B-Chat, as well as additional evaluations on the Qwen2.5 models. The experimental results are summarized in Tables 7 and 8. These findings demonstrate that LaTen consistently outperforms distillation baselines in scenarios with extremely limited training data. Specifically, LaTen achieves performance that is comparable to or better than approaches such as supervised knowledge distillation (Supervised KD)(Hinton, 2015), sequential knowledge distillation (SeqKD)(Kim and Rush, 2016), and generalized knowledge distillation (GKD) (Agarwal et al., 2024), while requiring significantly fewer training examples.

These results underscore the potential of LaTen for efficient parametric knowledge transfer, particularly in low-resource scenarios where access to training data is highly constrained. However, its performance remains limited by certain vulnerabilities, suggesting that exploring more robust methods for parametric knowledge transfer presents an intriguing direction for future research.

Models	**GSM8K**	**Traning Data**
Llama-2-7B-Chat	16.07	-
Llama-2-13B-Chat	20.55	-
# Pre-Align Parametric Knowledge Transfer		
Pre-Align on $\mathcal{D}^{\mathcal{T}}_{\text{align}}$		
-LaTen + Pre-Aligned 13B Param. (5 Steps)	20.47	5×16
# Language-based Knowledge Distillation		
Distillation on $\mathcal{D}^{\mathcal{T}}_{\text{align}}$		
-SeqKD (5 Epochs)	16.60	5×80
-Supervised KD (5 Epochs)	16.60	5×80
-GKD (5 Epochs)	16.60	5×80

Table 7: Results for comparison with language-based knowledge distillation baselines in the Llama-2 series. Both Pre-Align (step size 16) and distillation are conducted on $\mathcal{D}^{\mathcal{T}}_{\text{align}}$, consisting of 80 examples.

Models	**GSM8K**	**Training Data**
Qwen2.5-1.5B	62.02	-
Qwen2.5-3B	73.24	-
# Pre-Align Parametric Knowledge Transfer		
Pre-Align on $\mathcal{D}^{\mathcal{T}}_{\text{align}}$		
-LaTen + Pre-Aligned 3B Param. (5 Steps)	62.32	5×16
# Language-based Knowledge Distillation		
Distillation on $\mathcal{D}^{\mathcal{T}}_{\text{train}}$		
-SeqKD (5 Epochs)	**62.47**	5×1000
-Supervised KD (5 Epochs)	61.87	5×1000
-GKD (1 Epoch)	62.02	1×1000

Table 8: Results for comparison with language-based knowledge distillation baselines in the Qwen2.5 series. Both Pre-Align (step size 16) and distillation are conducted on $\mathcal{D}^{\mathcal{T}}_{\text{train}}$, consisting of 1000 examples.

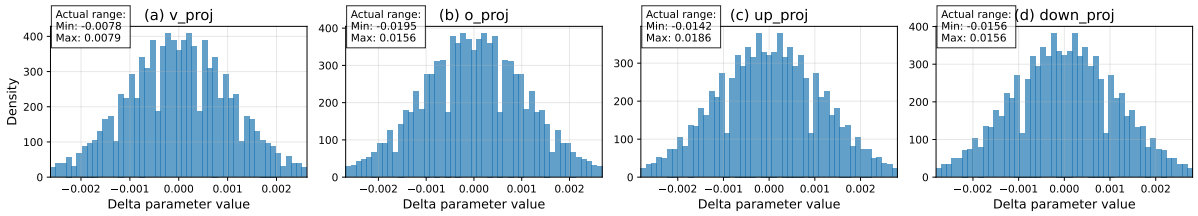

Figure 7: Delta parameter ranges between Llama-2-7b and Llama-2-7b-Chat

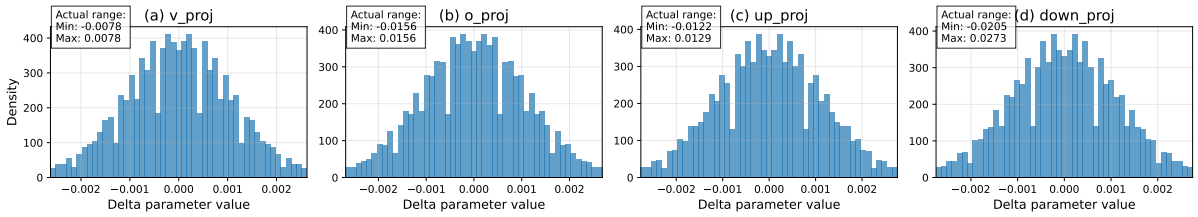

Figure 8: Delta parameter ranges between Llama-2-13b and Llama-2-13b-Chat

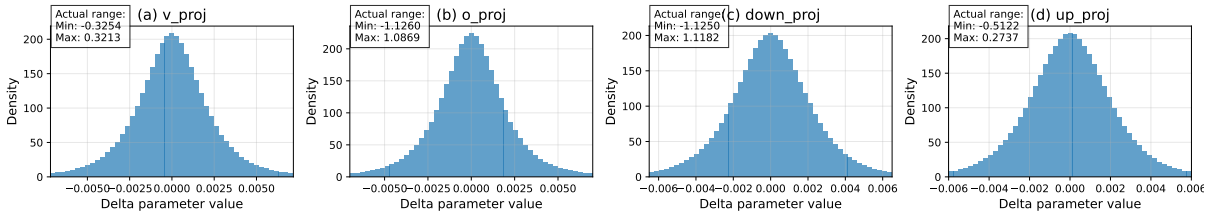

Figure 9: Delta parameter ranges from Llama-2-13b-Chat in GSM8K using SEEKING