The Master Key Hypothesis: Unlocking Cross-Model Capability Transfer via Linear Subspace Alignment

Rishab Balasubramanian
Virginia Tech
rishbb@vt.edu

Pin-Jie Lin
Virginia Tech
pinjie@vt.edu

Rituraj Sharma
Virginia Tech
rituraj@vt.edu

Mohit Bansal
UNC Chapel Hill
mbansal@cs.unc.edu

Tu Vu
Virginia Tech
tuvu@vt.edu

Abstract

We investigate whether post-trained capabilities can be transferred across model scales without retraining, and propose the **Master Key Hypothesis**, which states that model capabilities correspond to directions in a low-dimensional latent subspace that induce specific behaviors and are transferable across models through linear alignment. Based on the hypothesis, we introduce UNLOCK, a *training-free* and *label-free* framework that extracts a capability direction by contrasting activations between capability-present and capability-absent Source variants, aligns it with a Target model through a low-rank linear transformation, and applies it at inference time to elicit the behavior. Experiments on reasoning behaviors, including Chain-of-Thought (CoT) and mathematical reasoning, demonstrate substantial improvements across model scales without training. For example, transferring CoT reasoning from `Qwen1.5-14B` to `Qwen1.5-7B` yields an accuracy gain of 12.1% on MATH, and transferring a mathematical reasoning direction from `Qwen3-4B-Base` to `Qwen3-14B-Base` improves AGIEval Math accuracy from 61.1% to 71.3%, surpassing the 67.8% achieved by the 14B post-trained model. Our analysis shows that the success of transfer depends on the capabilities learned during pre-training, and that our intervention amplifies latent capabilities by sharpening the output distribution toward successful reasoning trajectories.

`https://github.com/rishabbala/Steering-Vector-Transfer`

1 Introduction

Each new model often requires substantial data, computation, and engineering effort to instill useful behaviors in the post-training phase. As models proliferate, this redundancy creates a fundamental inefficiency: *capabilities are costly to learn, yet difficult to reuse across models.* A mechanism that transfers capability-inducing representations across models to reliably elicit post-training behaviors without the need for retraining could reduce training costs, accelerate development, and enable modular reuse of existing model capabilities. Concretely, we ask: *can a desired capability that is expressed in one model be isolated and transferred to another model without gradient-based training or labeled supervision?*

Existing methods for capability transfer can be decomposed into two steps: (1) extracting a transformation from two Source variants that differ in an intended behavior and (2) applying this transformation to a Target model to reproduce the desired behavior. These approaches differ primarily in the space in which this transformation is represented, and can be broadly categorized into three distinct frameworks — (1) **Weight-space transfer** [23]: The parameter-level difference between two Source

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

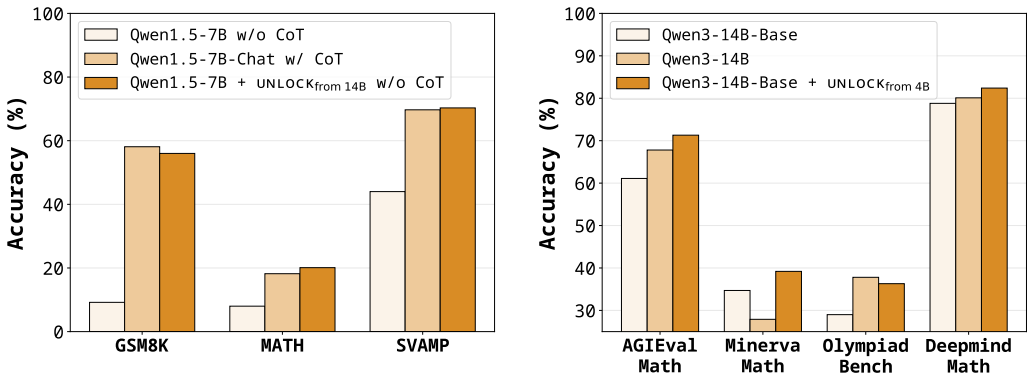


Figure 1: **Performance improvements from UNLOCK** when transferring (a) Chain-of-Thought capabilities from `Qwen1.5-14B` onto `Qwen1.5-7B`; and (b) Math reasoning capabilities from `Qwen3-4B-Base` onto `Qwen3-14B-Base`. Capability transfer substantially improves the base model *without additional training*, approaching the performance of the post-trained model.

models is added directly to the weights of the Target model, which typically requires architectural compatibility between the Source and Target models, or additional pruning or corrective training for cross-model alignment; (2) **Output-space transfer** [30]: The logit difference between two Source variants is applied at each generation step to adjust the output distribution of the Target model without modifying its parameters. This avoids shape mismatches, but incurs substantial inference cost due to per-token logit computation for multiple models, and requires identical tokenization between the source and target; and (3) **Latent-space transfer** [37]: This strategy involves intervening on internal representations to steer the Target model towards the desired output.

In this work, we focus on the *latent space*, where capabilities are encoded as shifts in internal activations. Existing methods typically construct steering directions from labeled contrastive examples (positive vs. negative [53, 37]) using a *single* Source model and apply them at inference time to similar prompts. Furthermore, these methods are largely focused on alignment and surface-level behavioral control (e.g., safety, toxicity, bias, and stylistic shaping [31, 48, 13, 9, 47]) rather than advanced capabilities such as reasoning.

We address these limitations by proposing UNLOCK — a *training-free* and *label-free* framework for cross-model capability transfer. Our method involves three main stages: *First*, we extract a MASTERKEY — a capability direction in a Source model's representation space by contrasting internal activations between capability-present and capability-absent variants using a small set of unlabeled prompts (Section 3.2). *Second*, we estimate a low-rank linear transformation that aligns this direction with the latent space of a Target model (Section 3.3). *Finally*, we apply the transferred direction as a normalized inference-time intervention to elicit the corresponding behavior (Section 3.4).

As case studies, we evaluate our capability transfer approach on reasoning behaviors, including Chain-of-Thought (Section 4) and mathematical reasoning (Section 5). We find that the transferred capability leads to substantial performance improvements, and can match the gains from post-training. As shown in Figure 1, transferring a Chain-of-Thought (CoT) direction from `Qwen1.5-14B` to `Qwen1.5-7B` improves the accuracy on MATH from 8.0% to 20.1% *without* explicit CoT prompting, which outperforms the 18.2% achieved by the 7B instruction-tuned model *with* CoT prompting. Notably, transferring mathematical reasoning from `Qwen3-4B` to `Qwen3-14B` improves AGIEval Math accuracy from 61.1% to 71.3%, which surpasses the 67.8% achieved by the 14B instruction-tuned model. Lastly, we provide preliminary experiments for cross-family transfer of CoT behavior (Appendix D), and observe consistent performance gains, offering initial evidence for the convergence of capability representations across model families as postulated by [21].

Our analysis reveals several consistent patterns. First, capability transfer exhibits a directional asymmetry: small-to-large transfer typically yields larger relative improvements than large-to-small transfer. Second, for both CoT and mathematical reasoning, UNLOCK amplifies capabilities that are dormant in the model, yielding greater gains when those capabilities are more strongly represented. Finally, we provide evidence that UNLOCK sharpens the output distribution, and directs generation

toward reasoning trajectories that are more likely to succeed. Based on our results and observations, we introduce the *Master Key Hypothesis* below, and defer a formalized version to Appendix E.

Hypothesis 1.1. ***The (Informal) Master Key Hypothesis:*** *Capabilities exist as directions in a low-dimensional latent subspace, such that shifting the representations along these directions induces the desired behaviors. These directional vectors can then be isolated and mapped into the representation space of another model via linear transformations, thus eliciting the intended capability.*

To summarize, our main contributions are:

- **Training-free Capability Transfer:** We propose a method that extracts a capability-inducing direction (MASTERKEY) from a pair of Source models and transfers it to a Target model via low-rank linear subspace alignment, enabling capability reuse without additional training.
- **The Master Key Hypothesis:** We hypothesize that model capabilities correspond to directions in a shared low-dimensional latent subspace, which can be isolated and transferred across models via linear transformations.
- **Empirical Validation Across Model Sizes:** Through extensive experiments, we demonstrate that reasoning behaviors, including Chain-of-Thought and mathematical reasoning, can be transferred across models of different sizes, yielding substantial improvements that approach or match gains typically obtained through post-training.
- **Analysis of Transfer Dynamics:** We analyze the factors that influence capability transfer, namely, the effect of model family and scale on transfer, and the effect of steering on the model's output distribution.

2 Related Work

Steering vectors: Steering vectors modulate model behavior by intervening on internal activations [53], with early work emphasizing safety-relevant behaviors [37]. A broad literature argues that many attributes are captured by low-dimensional directions [15, 2, 25, 54, 70]. Steering has also been used to improve reasoning and downstream performance and to support mechanistic analysis [31, 48, 52, 46, 13, 9, 18, 56, 47, 55, 69]. Recently, there has been growing interest in distilling capabilities in language models using steering vectors. The authors of [3] show that concise Chain-of-Thought abilities can be isolated as a single vector *within* a language model. We further distinguish our findings from previous work in Appendix A.1

Capability transfer across models: *Weight-space* transfer methods reuse parameter deltas as task directions [23, 19, 7, 63, 58, 67], but typically do not carry across sizes or families. *Output-space* methods guide a student using a stronger model's outputs [36, 12, 30], but require forward passes through multiple models at inference.

Representational convergence and cross-model alignment: Previous work argues that disparate models learn similar representations, enabling transfer through shared subspaces or maps [24, 6, 20]. We acknowledge concurrent efforts to learn mappings across model sizes [35, 5]. Unlike prior work, we use a *low-rank linear alignment* rather than non-linear autoencoders or full-dimensional pseudoinverse matrices, and we focus on quantifiable improvements on downstream tasks.

Knowledge distillation: Finally, classical distillation transfers capabilities by *training* a student model to match a teacher distribution [14, 60, 14, 43, 8, 40]. Unlike our setting, distillation typically incurs a nontrivial training cost and must be repeated per student model. Concurrently with our work, others have explored self-distillation in language models [44, 22], and claim that contextual knowledge and capabilities can be distilled into a model simply by training on it's logits along with additional feedback or examples. While we take a training-free approach, these works provide further grounding and motivation by empirically proving that target abilities can be elicited simply by incorporating additional task-conditioned signals.

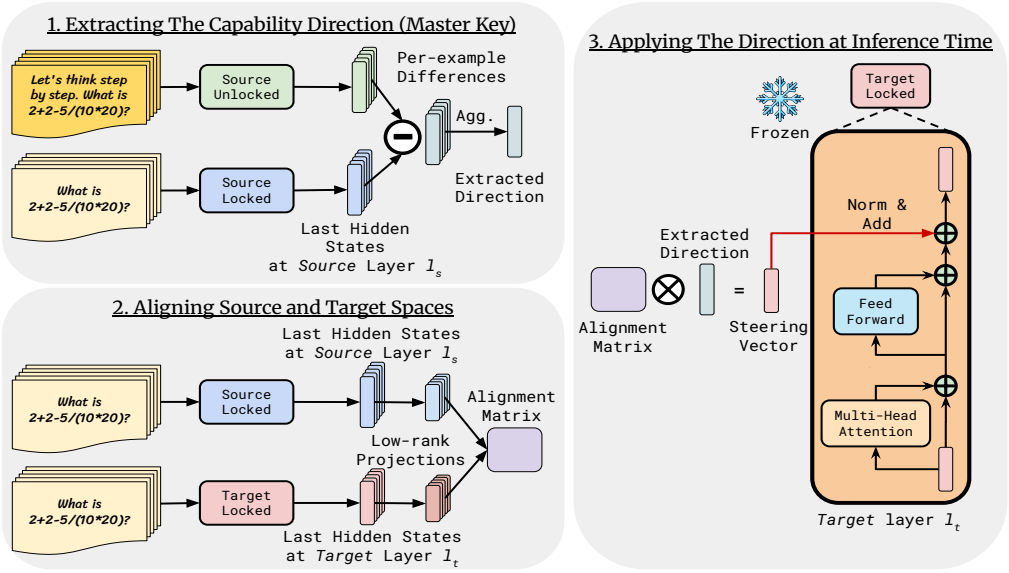


Figure 2: **Illustration of UNLOCK:** Our method consists of three stages: (1) Calculating the difference in hidden states in the Source space; (2) Learning a linear transformation between the Source Locked and Target Locked models; and (3) Projecting the MASTERKEY from Source to Target space and applying as a test-time intervention to the residual stream at every layer.

3 Preliminaries

In this section, we introduce UNLOCK, a training-free and label-free framework for transferring capability-inducing directions across models. The core idea is to represent a capability as a direction in representation space that shifts a model from a state where the behavior is weak or absent to one where it reliably emerges. The framework involves three conceptual models:

- **Source Locked** $\mathcal{S}_{\mathrm{L}}$ **:** a Source variant in which the desired capability is weak or absent.
- **Source Unlocked** $\mathcal{S}_{\mathrm{U}}$ **:** a Source variant that reliably exhibits the capability.
- **Target Locked** $\mathcal{T}_{\mathrm{L}}$ **:** the Target model in which we aim to elicit the capability.

The Source variants $\mathcal{S}_{\mathrm{L}}$ and $\mathcal{S}_{\mathrm{U}}$ share the same architecture and tokenizers, allowing their internal activations to be directly compared. Their contrast isolates a capability direction in the *Source representation space*. This direction is then mapped into the *Target representation space* and applied during inference to $\mathcal{T}_{\mathrm{L}}$, producing the **Target Unlocked** model $\mathcal{T}_{\mathrm{U}}$.

At a high level, UNLOCK consists of *three* stages (1) Extracting a capability direction (MASTERKEY) from the Source variants (Section 3.2); (2) Aligning representation spaces between the Source and Target models (Section 3.3); and (3) Applying the transferred direction at inference time (Section 3.4). The entire procedure only requires forward passes on a small set of unlabeled prompts.

3.1 Problem Setup

Let $\mathcal{M}$ denote a language model with hidden dimension $d_{\mathcal{M}}$. For a *model-specific* prompt $P_{\mathcal{M}}$ (e.g., task instructions and/or a few demonstrations) and query q_i, we denote the final-token hidden state at layer l as $\mathbf{h}_{\mathcal{M}}^{(l)}(P_{\mathcal{M}} \oplus q_i) \in \mathbb{R}^{d_{\mathcal{M}}}$, where $\oplus$ denotes sequence concatenation. We assume access to a small set of unlabeled queries $\mathcal{D} = \{q_i\}_{i=1}^{n}$. These queries are used both to extract the capability direction and to estimate the cross-model alignment.

3.2 Extracting The MASTERKEY

We first isolate a direction from the Source variants that elicits the desired capability ψ. For each query q_i and layer l, we compute a *per-example* representation difference between the Unlocked and Locked

representations $v_\psi^{(\mathcal{S},l,i)} = \mathbf{h}_{\mathcal{S}_\mathrm{U}}^{(l)}(P_{\mathcal{S}_\mathrm{U}} \oplus q_i) - \mathbf{h}_{\mathcal{S}_\mathrm{L}}^{(l)}(P_{\mathcal{S}_\mathrm{L}} \oplus q_i)$. This vector represents the activation shift required to move the Locked model towards the Unlocked behavior for that example. To obtain a dataset-level direction, we aggregate these differences across queries as $\mathbf{v}_\psi^{(\mathcal{S},l,\Phi)} = \Phi\left(v_\psi^{(\mathcal{S},l,i)}\right)$ where Φ is any aggregator function. In this work, we consider two aggregator functions — the *mean* and *principal component* aggregators, and provide a formal definition for both in Appendix A.2.

3.3 Cross-model Subspace Alignment

The capability direction MASTERKEY extracted in Section 3.2 lies in the Source representation space, and we compute a linear mapping to transfer it onto the Target representation space.

Specifically, we first collect hidden representations from the Source Locked model $\mathcal{S}_\mathrm{L}$ and the Target Locked model $\mathcal{T}_\mathrm{L}$. Let l_s denote the layer of $\mathcal{S}_\mathrm{L}$ used to extract Source representations, and l_t the layer of $\mathcal{T}_\mathrm{L}$ where the transferred MASTERKEY will be applied (Section 3.4 describes how l_s is selected for a given l_t). To reduce prompt-induced variance, we use a shared prompt $p = P_{(\mathcal{S}_\mathrm{L},\mathcal{T}_\mathrm{L})}$ for both models.[1] Given the query set $\mathcal{D} = \{q_i\}_{i=1}^n$, we stack representations across queries to obtain

$$X_\mathcal{S} = \left[\left(\mathbf{h}_{\mathcal{S}_\mathrm{L}}^{(l_s)}(p \oplus q_i)\right)^\top\right]_{i=1}^n \in \mathbb{R}^{n \times d_\mathcal{S}}, \qquad X_\mathcal{T} = \left[\left(\mathbf{h}_{\mathcal{T}_\mathrm{L}}^{(l_t)}(p \oplus q_i)\right)^\top\right]_{i=1}^n \in \mathbb{R}^{n \times d_\mathcal{T}},$$

where $d_\mathcal{S}$ and $d_\mathcal{T}$ denote the hidden size of the Source and Target models, and $n \ll \min(d_\mathcal{S}, d_\mathcal{T})$.

Instead of aligning the full hidden spaces, we align *low-rank* subspaces that capture the dominant structure of the representations. To find these subspaces, we perform Singular Value Decomposition (SVD) on both matrices $X_\mathcal{S} = U_\mathcal{S}\Sigma_\mathcal{S}V_\mathcal{S}^\top$, $X_\mathcal{T} = U_\mathcal{T}\Sigma_\mathcal{T}V_\mathcal{T}^\top$, and retain the top-$k$ right singular vectors $V_\mathcal{S}^k \in \mathbb{R}^{d_\mathcal{S} \times k}$, $V_\mathcal{T}^k \in \mathbb{R}^{d_\mathcal{T} \times k}$, where $k \leq n$. Projecting the representations into these subspaces yields $\hat{X}_\mathcal{S} = X_\mathcal{S}V_\mathcal{S}^k \in \mathbb{R}^{n \times k}$, $\hat{X}_\mathcal{T} = X_\mathcal{T}V_\mathcal{T}^k \in \mathbb{R}^{n \times k}$. We learn a linear transformation $W \in \mathbb{R}^{k \times k}$ that aligns the projected representations by minimizing the Frobenius norm loss

$$W^\star = \arg\min_{W \in \mathbb{R}^{k \times k}} \left\|\hat{X}_\mathcal{S}W - \hat{X}_\mathcal{T}\right\|_F^2 = \hat{X}_\mathcal{S}^\dagger\hat{X}_\mathcal{T} \in \mathbb{R}^{k \times k},$$

where $\hat{X}_\mathcal{S}^\dagger \in \mathbb{R}^{k \times n}$ is the Moore–Penrose pseudoinverse of $\hat{X}_\mathcal{S}$. Using this alignment, we define a cross-model operator $R_{l_t} = V_\mathcal{T}^k(W^\star)^\top(V_\mathcal{S}^k)^\top \in \mathbb{R}^{d_\mathcal{T} \times d_\mathcal{S}}$, which maps vectors from the Source latent space to the Target latent space. Applying this operator to the Source MASTERKEY $\mathbf{v}_\psi^{(\mathcal{S},l_s,\Phi)} \in \mathbb{R}^{d_\mathcal{S}}$ yields the transferred capability direction $\mathbf{v}_\psi^{(\mathcal{T},l_t,\Phi)} = R_{l_t}\mathbf{v}_\psi^{(\mathcal{S},l_s,\Phi)} \in \mathbb{R}^{d_\mathcal{T}}$.

3.4 Unlocking The Target Model

Since the Source and Target models may have different depths, we align layers by relative position, following [11] which finds that representations at similar depths maintain structural relationships across model scales. Let $L_\mathcal{S}$ and $L_\mathcal{T}$ denote the number of layers in the Source and Target models, respectively. For each target layer l_t, we choose the corresponding Source layer as $l_s = \min\left(L_\mathcal{S}, \max\left(1, \left\lfloor \frac{L_\mathcal{S}}{L_\mathcal{T}} l_t \right\rfloor\right)\right)$.

For a new input query, we apply the Target MASTERKEY during inference to steer the hidden representations of the Target model $\mathcal{T}_\mathrm{L}$. At each layer l_t, the final-token hidden state $\mathbf{h}_{\mathcal{T}_\mathrm{L}}^{(l_t)}$ is steered and normalized as

$$\tilde{\mathbf{h}}_{\mathcal{T}_\mathrm{L}}^{(l_t)} = \frac{\mathbf{h}_{\mathcal{T}_\mathrm{L}}^{(l_t)}}{\left\|\mathbf{h}_{\mathcal{T}_\mathrm{L}}^{(l_t)}\right\|_2} + \alpha\frac{\mathbf{v}_\psi^{(\mathcal{T},l_t,\Phi)}}{\left\|\mathbf{v}_\psi^{(\mathcal{T},l_t,\Phi)}\right\|_2}, \qquad \mathbf{h}_{\mathcal{T}_\mathrm{U}}^{(l_t)} = \frac{\tilde{\mathbf{h}}_{\mathcal{T}_\mathrm{L}}^{(l_t)}}{\left\|\tilde{\mathbf{h}}_{\mathcal{T}_\mathrm{L}}^{(l_t)}\right\|_2} \times \left\|\mathbf{h}_{\mathcal{T}_\mathrm{L}}^{(l_t)}\right\|_2, \tag{1}$$

where α controls the strength of the intervention. Applying this intervention across layers during generation yields the Target Unlocked model $\mathcal{T}_\mathrm{U}$. Figure 2 shows the three stages of our method.

We denote the transfer of a specific capability ψ from the Source pair of models to the Target model as $\mathcal{T}_\mathrm{L}$ + UNLOCK$_{\text{from } \mathcal{S}_\mathrm{L}}$.[2] To avoid confusion, we refer to the Target model that has undergone

[1] We note that the shared prompt is to minimize noise from the prompts. Our framework theoretically allows any combination of prompts to be applied between $\mathcal{S}_\mathrm{L}$ and $\mathcal{T}_\mathrm{L}$.

[2] Since we always deploy a base version for the Locked models, we use the model name and size to represent $\mathcal{S}_\mathrm{L}$ and $\mathcal{T}_\mathrm{L}$, and drop additional suffixes such as `-Base` or `-pt`.

Table 1: **Chain-of-Thought Capability Transfer.** Transfer performance across model families. Accuracies of base model with Direct prompting ($\mathcal{T}_{\mathrm{L}}$) and base model with explicit CoT prompting shown in gray.

Model	Prompt	$\mathcal{T}_{\mathrm{L}}$	$\mathcal{S}_{\mathrm{U}} \equiv \mathcal{S}_{\mathrm{L}}$	GSM8K	MATH	SVAMP
Qwen1.5	Direct	7B	–	9.2	8.0	44.0
		14B	–	16.0	16.0	58.3
	CoT	7B	–	64.4	17.9	73.0
		14B	–	77.3	26.8	79.0
	Direct	7B	+UNLOCK $_{\text{from 14B}}$	56.0	20.1	70.3
		14B	+UNLOCK $_{\text{from 7B}}$	74.4	31.2	78.3
OLMo-2	Direct	7B	–	10.0	9.7	43.7
	CoT	7B	–	53.8	15.3	71.0
	Direct	7B	+UNLOCK $_{\text{from 1B}}$	63.4	15.1	59.7
		7B	+UNLOCK $_{\text{from 13B}}$	36.1	14.3	58.7
gemma-2	Direct	2B	–	5.8	6.2	36.7
		9B	–	3.0	3.5	21.0
	CoT	2B	–	13.3	8.9	31.7
		9B	–	66.6	26.4	79.3
	Direct	2B	+UNLOCK $_{\text{from 9B}}$	9.5	6.4	37.7
		9B	+UNLOCK $_{\text{from 2B}}$	60.1	26.4	74.3

extensive post-training as the **post-trained Target** model $\mathcal{T}^{*}_{\mathrm{PT}}$. We treat the aggregation method Φ, subspace rank k, number of queries n, and steering strength α as hyperparameters, which are selected via grid search on a held-out development set. We provide a discussion of these hyperparameters and the low-rank nature of the capability subspace in Appendix B.2.

3.5 Atomic And Non-Atomic Capabilities

Conceptually, we define a capability as any model behavior that can be consistently observed across a distribution of semantically similar inputs and prompt templates, remaining invariant to minor changes in them. We further distinguish between *atomic* capabilities that are *latent* within a model (elicitable via steering or prompting) and *non-atomic* capabilities that are *absent* (requiring explicit training to acquire). We provide a more formal definition of both in Appendix A.4. In Sections 4, 5 we show that the atomicity of the capability impacts the gains from UNLOCK.

4 Atomic Capability Transfer

First, we ask whether capability-inducing directions can be extracted from prompt-induced representational changes within a single model (i.e., $\mathcal{S}_{\mathrm{U}} \equiv \mathcal{S}_{\mathrm{L}}$) and then transferred across model scales. This setting provides a controlled test of the Master Key Hypothesis: since the model weights remain fixed, any behavioral change must arise from shifts in internal representations. We study this question through Chain-of-Thought (CoT) reasoning, which often emerges in sufficiently capable base language models and can be elicited through prompting alone. Empirically, we find that UNLOCK makes step-by-step thinking more consistently expressed, improving reasoning performance across model families and benchmarks even in the absence of explicit CoT prompting.

4.1 Experimental Setup

We evaluate prompt-induced capability transfer across model scales within five model families: `Qwen1.5` [4], `Qwen2.5` [65], `Qwen3` [64], `OLMo-2` [57], and `gemma-2` [49]. For each model, we construct Source variants using two prompts: a *Direct* prompt that requests only the final answer and

a *CoT* prompt that encourages step-by-step reasoning (see Appendix A.3 for the prompts used). We evaluate performance on three reasoning benchmarks — GSM8K [10], MATH [17], and SVAMP [39].

4.2 Results & Discussion

We provide our results in Table 1, and additional results in Appendix B. We find that UNLOCK (i) consistently improves reasoning performance and displays structured reasoning traces; (ii) is asymmetric in its impact: small-to-large transfer outperforms large-to-small transfer; and (iii) is most effective when the desired capability is already present in latent space.

UNLOCK Consistently Improves Reasoning Performance: Across all evaluated model families and datasets, the *Target Unlocked* model $\mathcal{T}_{\text{U}}$ consistently outperforms the baseline $\mathcal{T}_{\text{L}}$ under *Direct* prompting. In the `Qwen1.5` model family, large-to-small (`Qwen1.5-7B` + $\text{UNLOCK}_{\text{from } 14B}$) and small-to-large (`Qwen1.5-14B` + $\text{UNLOCK}_{\text{from } 7B}$) produce average accuracy gains of 25.0% and 31.2%, respectively. $\mathcal{T}_{\text{U}}$ also performs comparable to the accuracy obtained by prompting $\mathcal{T}_{\text{L}}$ with *explicit CoT* instructions.

Figure 3 plots the average length of generated outputs for each model and dataset. A consistent increase in generation length is observed across all model–dataset pairs, supporting the view that the performance gains stem from Chain-of-Thought elicitation rather than surface-level output changes. We provide further analysis into the structure of the generated outputs and examples of step-by-step reasoning in Appendix B.

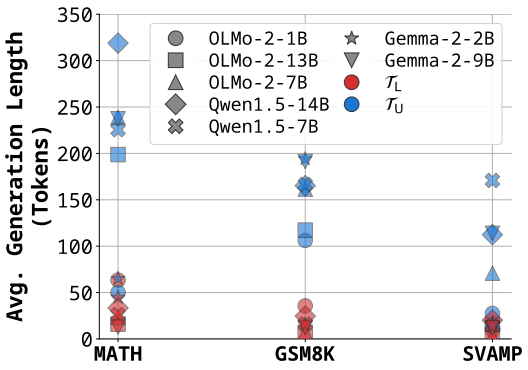


Figure 3: **Comparison in Generation Length between $\mathcal{T}_{\text{L}}$ and $\mathcal{T}_{\text{U}}$:** A consistent increase in generation length is observed post-transfer, representative of reasoning behavior.

Asymmetry in Transfer Direction: We observe a consistent directional asymmetry: small-to-large transfer typically outperforms large-to-small transfer. A plausible explanation is that larger models implement a functional superset of the mechanisms present in smaller models. Under this view, a CoT direction transferred from a smaller model can activate latent circuitry already present in the larger model. The reverse, however, is capacity-limited: the smaller model's reduced representational capacity may be insufficient to support the more complex reasoning structure of the larger model. This is illustrated clearly within the `gemma-2` family. In the small-to-large direction, $\mathcal{T}_{\text{U}}$ improves by an average of 44.4% over $\mathcal{T}_{\text{L}}$ and comes within 18.5% of $\mathcal{T}^{*}_{\text{PT}}$, while large-to-small transfer improves $\mathcal{T}_{\text{U}}$ by only 1.6% over $\mathcal{T}_{\text{L}}$ and remains 32.3% below $\mathcal{T}^{*}_{\text{PT}}$. Importantly, we observe a similar asymmetry when using CoT prompts: `gemma-2-2B` improves by 2% while `gemma-2-9B` improves by 48.2%. These results suggest that, like prompting, UNLOCK improves with scale and cannot introduce capabilities that are absent from the model.

Transfer Effectiveness Depends On The Salience of The Capability in $\mathcal{T}_{\text{L}}$: Within the `Qwen1.5` family, base and instruction-tuned variants exhibit similar performance under CoT prompting, suggesting the reasoning capability is largely introduced during pre-training and can be reliably elicited by prompting. Consequently, $\mathcal{T}_{\text{U}}$ significantly outperforms $\mathcal{T}_{\text{L}}$ and remains within 1% $\mathcal{T}^{*}_{\text{PT}}$. In contrast, `gemma-2` models exhibit a substantial gap between their base and instruction-tuned versions with similar prompting, providing evidence that step-by-step reasoning is learned during the post-training process. Here, UNLOCK consistently improves $\mathcal{T}_{\text{U}}$ over $\mathcal{T}_{\text{L}}$ but does not match $\mathcal{T}^{*}_{\text{PT}}$, with particularly small gains for `gemma-2-2B`. Similar trends across scales are also observed within a model family, as demonstrated by `Qwen2.5` (Appendix B).

Table 2: **Math Reasoning Transfer Results:** Performance of UNLOCK across model families. For simplicity, we use $\mathcal{I}(x)$ to denote the instruction-tuned version of the corresponding model. Highlighted rows show the performance of task-conditioned and task-agnostic transfer respectively.

Model	$\mathcal{T}_{\text{L}}$	$\mathcal{S}_{\text{U}} \equiv \mathcal{I}(\mathcal{S}_{\text{L}})$	AGIEval Math	Deepmind Math	Minerva Math	Olympiad Bench
Qwen3	4B	–	52.3	71.3	27.5	19.7
	14B	–	61.1	78.8	34.7	29.0
	$\mathcal{I}$(4B)	–	75.6	88.4	31.5	39.8
	$\mathcal{I}$(14B)	–	67.8	80.1	27.9	37.8
	4B	+UNLOCK from 14B	58.9	75.8	27.0	26.4
	14B	+UNLOCK from 4B	64.1	79.9	31.5	35.4
	4B	+UNLOCK from 14B	49.5	72.9	25.7	20.8
	14B	+UNLOCK from 4B	71.3	82.4	39.2	36.3
Ministral-3	3B	–	46.9	65.3	26.1	19.0
	8B	–	50.7	67.4	29.3	20.0
	$\mathcal{I}$(3B)	–	68.7	84.2	26.6	33.9
	$\mathcal{I}$(8B)	–	70.6	87.2	29.3	37.0
	3B	+UNLOCK from 8B	53.4	66.2	27.5	21.0
	8B	+UNLOCK from 3B	51.9	71.3	37.4	20.2
	3B	+UNLOCK from 8B	49.9	65.5	27.5	21.0
	8B	+UNLOCK from 3B	54.0	70.7	34.7	21.1

5 Non-Atomic Capability Transfer

Section 4 established that UNLOCK and prompting play analogous roles in eliciting desired model behavior. Motivated by [21] (which states that larger models converge towards shared representations), and [61, 66] (where the authors find RLVR to sharpen the output distribution rather than introduce new knowledge), we aim to induce these post-training behaviors with UNLOCK. Intuitively, if post-training merely evokes latent capabilities, and if these capabilities reside in a shared representation space, then transferring them across models becomes a natural next step. Since these behaviors are not reliably observed in the base model through prompting alone, we ask: *can latent interventions activate non-atomic capabilities that prompting alone cannot?* We study this question through the lens of *mathematical reasoning*, which is one of the main focuses of modern post-training methods.

5.1 Experiment Setup

We study two contrasting experimental settings:

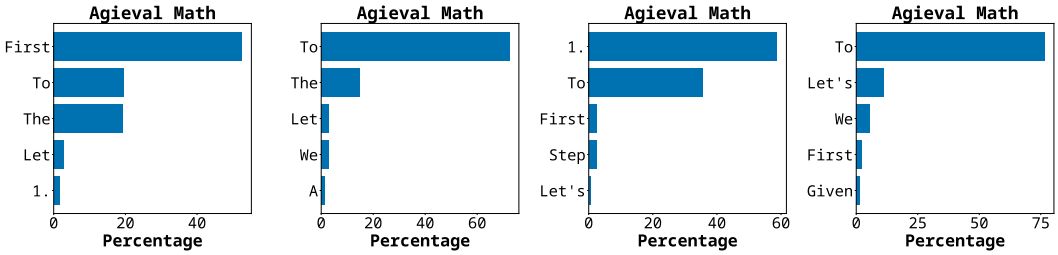
Task-Conditioned Transfer With Limited Data: The MASTERKEY, transformation, and hyperparameters are all estimated using *few* examples from the *same* task as evaluation. This follows the standard practice in the steering vector literature, where the steering direction is computed on the target task to maximize alignment with the evaluation distribution. Since the evaluation set consists of a limited number of samples, we carry out all pre-computation on a small disjoint development set.

Task-Agnostic Transfer With Abundant Data: Mirroring conventional post-training practices, the MASTERKEY and hyperparameters are estimated on a *large* dataset from a *different* math task and applied to all evaluation datasets without modification. This setting tests whether the learned intervention captures *general* mathematical reasoning behavior that transfers across tasks.

These two settings expose a central tradeoff between the *in-distribution* signal and the *data volume* required. In the task-conditioned regime, we estimate the MASTERKEY and alignment using limited in-distribution data, which is directly aligned with the evaluation suite, but can yield a noisier and less stable direction/transformation. In contrast, the task-agnostic regime leverages abundant out-of-

Table 3: **Math Reasoning Transfer Under Pass@k Evaluation:** Pass@k accuracy (%) for $k \in \{2, 4, 8\}$ with temperature 0.7 and top-p 0.95. Each UNLOCK row uses the configuration with the highest greedy-decoding accuracy in Table 2. Bold marks the best result in each column.

Model	$\mathcal{T}_{\text{L}}$	$\mathcal{S}_{\text{U}}$	AGIEval Math	DeepMind Math	Minerva Math	Olympiad Bench
Qwen3	4B	–	59.0/68.7/76.1	79.6/88.6/92.9	30.5/38.8/46.4	24.2/32.1/39.5
	4B	+UNLOCK$_{\text{from 14B}}$	**63.1/71.9/78.6**	**80.7/89.1/93.1**	**32.2/40.1/48.3**	**31.9/39.2/45.2**
	14B	–	70.2/77.4/81.8	86.0/91.7/**94.8**	41.8/49.7/56.4	35.4/42.9/49.2
	14B	+UNLOCK$_{\text{from 4B}}$	**74.5/80.4/84.2**	**87.0/92.2**/94.6	**43.8/52.7/59.0**	**37.4/44.5/50.6**
Ministral-3	3B	–	57.4/67.3/74.9	74.2/84.5/90.5	30.6/39.6/47.5	23.9/30.7/37.2
	3B	+UNLOCK$_{\text{from 8B}}$	**60.2/69.2/76.0**	**75.2/85.6/91.4**	**30.7/40.1/47.8**	**26.9/34.6/41.7**
	8B	–	58.6/67.8/74.5	**77.3/87.3**/92.5	36.2/44.3/50.2	**25.6/32.9/39.7**
	8B	+UNLOCK$_{\text{from 3B}}$	**58.8/68.1/76.1**	77.0/86.9/**92.7**	**36.5/46.0/53.9**	24.8/32.2/38.8



(a) Ministral-3-8B (b) Ministral-3-8B + UNLOCK$_{\text{from 3B}}$ (c) Qwen3-14B (d) Qwen3-14B + UNLOCK$_{\text{from 4B}}$

Figure 4: **Statistics of First Generated Word:** The output distribution is significantly skewed to a minimal set of starting traces post-steering.

distribution examples to learn a more robust MASTERKEY and mapping, at the cost of estimating them from a distribution-mismatched dataset. We discuss this tradeoff further in Appendix B.2

Models and Datasets: We focus on language models with strong reasoning capabilities from four model families: Qwen2.5 [41], Qwen3 [51], Ministral-3 [29], and gemma-3 [50]. Within each family, we use the instruction-tuned model as $\mathcal{S}_{\text{U}}$ and the corresponding base model as $\mathcal{S}_{\text{L}}$. We evaluate our framework across four mathematical reasoning benchmarks: AGIEval-Math [68], Deepmind-Math [42], Minerva-Math [26], and OlympiadBench [16]. We apply *CoT* prompting to all models, and therefore the information encoded by the MASTERKEY arises from the additional post-training efforts on $\mathcal{S}_{\text{U}}$. By utilizing different models for $\mathcal{S}_{\text{L}}$ and $\mathcal{S}_{\text{U}}$, we design a *model-induced* capability transfer setting. We provide additional experimental details and results with domain-specific models in Appendix C.

5.2 Results & Discussion

Table 2, 8 report the results for *task-conditioned transfer* and *task-agnostic transfer*. Our evaluations show consistent gains from UNLOCK, and further analysis shows that these gains arise from a convergence in output trajectories, providing evidence that the MASTERKEY acts as a distribution sharpening mechanism.

Unlocking Matches Gains From Post-Training: Consistent with the findings in Section 4, the Unlocked model $\mathcal{T}_{\text{U}}$ systematically outperforms the baseline $\mathcal{T}_{\text{L}}$ and often achieves performance comparable to, or even exceeding, the post-trained counterpart $\mathcal{T}^{*}_{\text{PT}}$. For example, Qwen3-14B + UNLOCK$_{\text{from 4B}}$ and Qwen3-4B + UNLOCK$_{\text{from 14B}}$ yield average gains of 6.4% and 4.3% over $\mathcal{T}_{\text{L}}$ respectively. Importantly, this shows that while CoT prompting alone is unable to elicit the math reasoning abilities, UNLOCK is able to achieve significant gains across models and tasks.

Table 4: **Accuracy Gains With Bootstrapped Sampling:** Improvement of UNLOCK over the Target Locked model with bootstrapped sampling of N samples over 10,000 runs. The first number represents the average gain, and the brackets give the 95% paired bootstrap confidence interval (the 2.5^{th} and 97.5^{th} percentiles).

Model	Target	Source	AGIEval Math	DeepMind Math	Minerva Math	Olympiad Bench
Qwen3	4B	14B	+6.63 [3.79, 9.47]	+4.53 [1.58, 7.47]	-0.45 [-5.41, 4.05]	+6.72 [3.36, 10.08]
	14B	4B	+10.21 [7.68, 12.74]	+3.58 [1.37, 5.89]	+4.50 [0.00, 9.46]	+7.36 [4.16, 10.56]
Ministral-3	3B	8B	+6.42 [3.47, 9.47]	+0.95 [-1.47, 3.37]	+1.35 [-3.15, 5.86]	+1.92 [-0.96, 4.96]
	8B	3B	+3.26 [0.74, 5.79]	+3.89 [1.58, 6.32]	+8.11 [2.25, 13.96]	+1.12 [-1.76, 4.00]

Robustness to Sampling: To test whether the gains from UNLOCK persist under sampling, we evaluate each Target Locked model and its Unlocked counterpart with stochastic decoding (temperature 0.7, top-p 0.95). For each target and benchmark, we use the best UNLOCK configuration from Table 2 and sample multiple responses per problem. Table 3 reports pass@k gains for $k \in \{2, 4, 8\}$. We find consistent improvements, with average improvements of $2.3\%/2.0\%/1.5\%$ for `Qwen3-14B` and $3.65\%/3.0\%/2.6\%$ for `Qwen3-4B`. `Ministral-3-3B` and `Ministral-3-8B` show similar trends, with `Ministral-3-3B` improving in all 12 cells (with mean gains of $1.7\%/1.9\%/1.7\%$) and achieving the largest gain on OlympiadBench ($+4.5\%$ at $k = 8$).

Statistical Analysis: Next, we use bootstrapped confidence intervals to test whether the gains from UNLOCK could be explained by noise. For a test benchmark of size N, we pair each question with the corresponding responses from the Locked and Unlocked Target models and resample with replacement to form a set of N tuples. We repeat the process $10{,}000$ times, reporting the average gain and the $[2.5^{\text{th}}, 97.5^{\text{th}}]$ percentiles as the 95% confidence interval in Table 4. The interval lies entirely above zero in 10 of the 16 settings, and the average gain is positive in 15 of the 16 settings. These results show that the gains from UNLOCK are not explained by noise or randomness.

Convergence of Reasoning Traces: To probe for the source of $\mathcal{T}_{\text{U}}$'s gains, we analyze the structure of the generated reasoning traces and find that the Unlocked model displays a narrower set of opening trajectories. We show the distribution of the first generated token of $\mathcal{T}_{\text{L}}$ and $\mathcal{T}_{\text{U}}$ in Figures 4, 13. Across models and datasets, we find $\mathcal{T}_{\text{U}}$ to converge in its opening statements, while $\mathcal{T}_{\text{L}}$ displays a more diffuse distribution. We show examples of these changes, along with additional discussions in Appendix C. Combined with the improvement in downstream performance these patterns suggest that UNLOCK increases the likelihood of producing plausible reasoning traces by consolidating representations and reducing variability in early trajectory selection. We thus arrive at a similar conclusion as [61, 66], and find that this output reshaping mechanism of post-training can be captured in low dimensional subspaces and applied onto a target model to elicit similar capabilities.

6 Conclusion

In this paper, we present a training-free approach for *transferring capabilities* across models. Our method extracts a MASTERKEY direction from prompt- /model-induced representational differences and transfers it to a new model via low-rank linear subspace alignment, avoiding gradient updates and requiring no architectural or tokenization correspondence between Source and Target pairs. Empirical evaluations across multiple model families and benchmarks confirm the effectiveness of our approach. More broadly, our results support the *Master Key Hypothesis*, suggesting that useful behaviors can be isolated as linearly transferrable latent directions in shared low-dimensional subspaces.

7 Acknowledgments

We thank the VT LLM Lab for useful discussions and feedback. This project was partially supported by award IIS-2553891 from the National Science Foundation (NSF), as well as awards from the Amazon–Virginia Tech Initiative for Efficient and Robust Machine Learning. We thank Google for a Gemini Academic Program award that provided access to Gemini API credits. The authors acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper.[3]

References

[1] Wasi Uddin Ahmad, Somshubra Majumdar, Aleksander Ficek, Sean Narenthiran, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Vahid Noroozi, and Boris Ginsburg. Opencodereasoning-ii: A simple test time scaling approach via self-critique, 2025. URL https://arxiv.org/abs/2507.09075.

[2] Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024. URL https://arxiv.org/abs/2406.11717.

[3] Seyedarmin Azizi, Erfan Baghaei Potraghloo, and Massoud Pedram. Activation steering for chain-of-thought compression, 2025. URL https://arxiv.org/abs/2507.04742.

[4] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023. URL https://arxiv.org/abs/2309.16609.

[5] Femi Bello, Anubrata Das, Fanzhi Zeng, Fangcong Yin, and Liu Leqi. Linear representation transferability hypothesis: Leveraging small models to steer large models, 2025. URL https://arxiv.org/abs/2506.00653.

[6] Nimrod Berman, Assaf Hallak, and Assaf Shocher. Who said neural networks aren't linear?, 2025. URL https://arxiv.org/abs/2510.08570.

[7] Pietro Buzzega, Riccardo Salami, Angelo Porrello, and Simone Calderara. Rethinking layer-wise model merging through chain of merges, 2025. URL https://arxiv.org/abs/2508.21421.

[8] Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulić, and Alessandro Sordoni. Training plug-n-play knowledge modules with deep context distillation, 2025. URL https://arxiv.org/abs/2503.08727.

[9] Haozhe Chen, Carl Vondrick, and Chengzhi Mao. Selfie: Self-interpretation of large language model embeddings, 2024. URL https://arxiv.org/abs/2403.10949.

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

[11] Róbert Csordás, Christopher D Manning, and Christopher Potts. Do language models use their depth efficiently? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Kz6eUL86XP.

[12] Yu Fei, Yasaman Razeghi, and Sameer Singh. Nudging: Inference-time alignment of llms via guided decoding, 2025. URL https://arxiv.org/abs/2410.09300.

[13] Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models, 2024. URL https://arxiv.org/abs/2401.06102.

[3] https://arc.vt.edu/

[14] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models, 2025. URL https://arxiv.org/abs/2306.08543.

[15] Wes Gurnee and Max Tegmark. Language models represent space and time, 2024. URL https://arxiv.org/abs/2310.02207.

[16] Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. Olympiadbench: A challenging benchmark for promoting agi with olympiad-level bilingual multimodal scientific problems, 2024. URL https://arxiv.org/abs/2402.14008.

[17] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

[18] Yihuai Hong, Dian Zhou, Meng Cao, Lei Yu, and Zhijing Jin. The reasoning-memorization interplay in language models is mediated by a single direction, 2025. URL https://arxiv.org/abs/2503.23084.

[19] Shih-Cheng Huang, Pin-Zu Li, Yu-Chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tzong-Han Tsai, and Hung yi Lee. Chat vector: A simple approach to equip llms with instruction following and model alignment in new languages, 2024. URL https://arxiv.org/abs/2310.04799.

[20] Youcheng Huang, Chen Huang, Duanyu Feng, Wenqiang Lei, and Jiancheng Lv. Cross-model transferability among large language models on the platonic representations of concepts, 2025. URL https://arxiv.org/abs/2501.02009.

[21] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*, 2024. URL https://arxiv.org/abs/2405.07987.

[22] Jonas Hübotter, Frederike Lübeck, Lejs Behric, Anton Baumann, Marco Bagatella, Daniel Marta, Ido Hakimi, Idan Shenfeld, Thomas Kleine Buening, Carlos Guestrin, and Andreas Krause. Reinforcement learning via self-distillation, 2026. URL https://arxiv.org/abs/2601.20802.

[23] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

[24] Prakhar Kaushik, Shravan Chaudhari, Ankit Vaidya, Rama Chellappa, and Alan Yuille. The universal weight subspace hypothesis, 2025. URL https://arxiv.org/abs/2512.05117.

[25] Kai Konen, Sophie Jentzsch, Diaoulé Diallo, Peer Schütt, Oliver Bensch, Roxanne El Baff, Dominik Opitz, and Tobias Hecking. Style vectors for steering generative large language model, 2024. URL https://arxiv.org/abs/2402.01618.

[26] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

[27] Kaiyuan Li, Jing-Cheng Pang, and Yang Yu. Rlvr training of llms does not improve thinking ability for general qa: Evaluation method and a simple solution, 2026. URL https://arxiv.org/abs/2603.20799.

[28] Minpeng Liao, Wei Luo, Chengxi Li, Jing Wu, and Kai Fan. Mario: Math reasoning with code interpreter output–a reproducible pipeline. *arXiv preprint arXiv:2401.08190*, 2024.

[29] Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, Amos You, Andy Ehrenberg, Andy Lo, Anton Eliseev, Antonia Calvi, Avinash Sooriyarachchi, Baptiste Bout, Baptiste Rozière, Baudouin De Monicault,

Clémence Lanfranchi, Corentin Barreau, Cyprien Courtot, Daniele Grattarola, Darius Dabert, Diego de las Casas, Elliot Chane-Sane, Faruk Ahmed, Gabrielle Berrada, Gaëtan Ecrepont, Gauthier Guinet, Georgii Novikov, Guillaume Kunsch, Guillaume Lample, Guillaume Martin, Gunshi Gupta, Jan Ludziejewski, Jason Rute, Joachim Studnia, Jonas Amar, Joséphine Delas, Josselin Somerville Roberts, Karmesh Yadav, Khyathi Chandu, Kush Jain, Laurence Aitchison, Laurent Fainsin, Léonard Blier, Lingxiao Zhao, Louis Martin, Lucile Saulnier, Luyu Gao, Maarten Buyl, Margaret Jennings, Marie Pellat, Mark Prins, Mathieu Poirée, Mathilde Guillaumin, Matthieu Dinot, Matthieu Futeral, Maxime Darrin, Maximilian Augustin, Mia Chiquier, Michel Schimpf, Nathan Grinsztajn, Neha Gupta, Nikhil Raghuraman, Olivier Bousquet, Olivier Duchenne, Patricia Wang, Patrick von Platen, Paul Jacob, Paul Wambergue, Paula Kurylowicz, Pavankumar Reddy Muddireddy, Philomène Chagniot, Pierre Stock, Pravesh Agrawal, Quentin Torroba, Romain Sauvestre, Roman Soletskyi, Rupert Menneer, Sagar Vaze, Samuel Barry, Sanchit Gandhi, Siddhant Waghjale, Siddharth Gandhi, Soham Ghosh, Srijan Mishra, Sumukh Aithal, Szymon Antoniak, Teven Le Scao, Théo Cachet, Theo Simon Sorg, Thibaut Lavril, Thiziri Nait Saada, Thomas Chabal, Thomas Foubert, Thomas Robert, Thomas Wang, Tim Lawson, Tom Bewley, Tom Bewley, Tom Edwards, Umar Jamil, Umberto Tomasini, Valeriia Nemychnikova, Van Phung, Vincent Maladière, Virgile Richard, Wassim Bouaziz, Wen-Ding Li, William Marshall, Xinghui Li, Xinyu Yang, Yassine El Ouahidi, Yihan Wang, Yunhao Tang, and Zaccharie Ramzi. Ministral 3, 2026. URL https://arxiv.org/abs/2601.08584.

[30] Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy, 2024. URL https://arxiv.org/abs/2401.08565.

[31] Sheng Liu, Haotian Ye, Lei Xing, and James Zou. In-context vectors: Making in context learning more effective and controllable through latent space steering, 2024. URL https://arxiv.org/abs/2311.06668.

[32] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, et al. Dler: Doing length penalty right-incentivizing more intelligence per token via reinforcement learning. *arXiv preprint arXiv:2510.15110*, 2025.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

[34] Duy Nguyen, Archiki Prasad, Elias Stengel-Eskin, and Mohit Bansal. Grains: Gradient-based attribution for inference-time steering of llms and vlms. *arXiv preprint arXiv:2507.18043*, 2025. URL https://arxiv.org/abs/2507.18043.

[35] Narmeen Oozeer, Dhruv Nathawani, Nirmalendu Prakash, Michael Lan, Abir Harrasse, and Amirali Abdullah. Activation space interventions can be transferred between large language models, 2025. URL https://arxiv.org/abs/2503.04429.

[36] Siru Ouyang, Xinyu Zhu, Zilin Xiao, Minhao Jiang, Yu Meng, and Jiawei Han. Rast: Reasoning activation in llms via small-model transfer, 2025. URL https://arxiv.org/abs/2506.15710.

[37] Nina Panickssery, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Matt Turner. Steering llama 2 via contrastive activation addition, 2024. URL https://arxiv.org/abs/2312.06681.

[38] Kiho Park, Yo Joong Choe, and Victor Veitch. The linear representation hypothesis and the geometry of large language models, 2024. URL https://arxiv.org/abs/2311.03658.

[39] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

[40] Yujia Qin, Yankai Lin, Jing Yi, Jiajie Zhang, Xu Han, Zhengyan Zhang, Yusheng Su, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Knowledge inheritance for pre-trained language models, 2022. URL https://arxiv.org/abs/2105.13880.

[41] Qwen, :, 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, Tianyi Tang, 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, 2025. URL https://arxiv.org/abs/2412.15115.

[42] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models, 2019. URL https://arxiv.org/abs/1904.01557.

[43] Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation, 2025. URL https://arxiv.org/abs/2502.21074.

[44] Idan Shenfeld, Mehul Damani, Jonas Hübotter, and Pulkit Agrawal. Self-distillation enables continual learning, 2026. URL https://arxiv.org/abs/2601.19897.

[45] Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models, 2025. URL https://arxiv.org/abs/2502.02013.

[46] Niklas Stoehr, Kevin Du, Vésteinn Snæbjarnarson, Robert West, Ryan Cotterell, and Aaron Schein. Activation scaling for steering and interpreting language models, 2024. URL https://arxiv.org/abs/2410.04962.

[47] Alessandro Stolfo, Vidhisha Balachandran, Safoora Yousefi, Eric Horvitz, and Besmira Nushi. Improving instruction-following in language models through activation steering, 2025. URL https://arxiv.org/abs/2410.12877.

[48] Daniel Tan, David Chanin, Aengus Lynch, Dimitrios Kanoulas, Brooks Paige, Adria Garriga-Alonso, and Robert Kirk. Analyzing the generalization and reliability of steering vectors, 2025. URL https://arxiv.org/abs/2407.12404.

[49] 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. URL https://arxiv.org/abs/2408.00118.

[50] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi

Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

[51] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[52] Eric Todd, Millicent L. Li, Arnab Sen Sharma, Aaron Mueller, Byron C. Wallace, and David Bau. Function vectors in large language models, 2024. URL https://arxiv.org/abs/2310.15213.

[53] Alexander Matt Turner, Lisa Thiergart, Gavin Leech, David Udell, Juan J. Vazquez, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering, 2024. URL https://arxiv.org/abs/2308.10248.

[54] Teun van der Weij, Massimo Poesio, and Nandi Schoots. Extending activation steering to broad skills and multiple behaviours, 2024. URL https://arxiv.org/abs/2403.05767.

[55] Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Base models know how to reason, thinking models learn when, 2025. URL https://arxiv.org/abs/2510.07364.

[56] Constantin Venhoff, Iván Arcuschin, Philip Torr, Arthur Conmy, and Neel Nanda. Understanding reasoning in thinking language models via steering vectors, 2025. URL https://arxiv.org/abs/2506.18167.

[57] Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

[58] Fanqi Wan, Longguang Zhong, Ziyi Yang, Ruijun Chen, and Xiaojun Quan. Fusechat: Knowledge fusion of chat models, 2024. URL https://arxiv.org/abs/2408.07990.

[59] Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models, 2025. URL https://arxiv.org/abs/2512.13607.

[60] Jingyuan Wang, Yankai Chen, Zhonghang Li, and Chao Huang. Lightreasoner: Can small language models teach large language models reasoning?, 2025. URL https://arxiv.org/abs/2510.07962.

[61] Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning, 2025. URL https://arxiv.org/abs/2506.01939.

[62] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

[63] Taiqiang Wu, Runming Yang, Jiayi Li, Pengfei Hu, Yik-Chung Wu, Ngai Wong, and Yujiu Yang. Shadow-ft: Tuning instruct model via training on paired base model, 2025. URL https://arxiv.org/abs/2505.12716.

[64] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

[65] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025. URL https://arxiv.org/abs/2412.15115.

[66] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model?, 2025. URL https://arxiv.org/abs/2504.13837.

[67] Mohammad Zbeeb, Hasan Abed Al Kader Hammoud, and Bernard Ghanem. Reasoning vectors: Transferring chain-of-thought capabilities via task arithmetic, 2025. URL https://arxiv.org/abs/2509.01363.

[68] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023. URL https://arxiv.org/abs/2304.06364.

[69] Ziqian Zhong and Aditi Raghunathan. Watch the weights: Unsupervised monitoring and control of fine-tuned llms, 2025. URL https://arxiv.org/abs/2508.00161.

[70] Andy Zou, Long Phan, Sarah Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency, 2025. URL https://arxiv.org/abs/2310.01405.

A Additional Preliminaries

A.1 Comparison to Capability Transfer Approaches

We provide a comparison of our approach to prior work in Table 5. We are amongst the first to demonstrate that high-level capability transfer is inherently low-rank. Building on this insight, we perform the extensive evaluation of both large-to-small and small-to-large capability transfer using latent steering vectors. Crucially, our approach is entirely *training-free* and requires *no labeled data*, distinguishing it from existing methods that rely on gradient updates or supervised signals.

Table 5: **Comparison of our method to other approaches.** UNLOCK is training-free and label-free, and improves across model scales and architectures on extrinsic evaluation tasks.

Method	Transfer Space	No Labeled Data	Fixed Compute	Transferable Across Sizes	Extrinsic Evaluations
Task Vectors [23]	Weight	✓	✓	✗	✓
Knowledge Distillation [14]	Weight	✗	✓	✓	✓
Proxy Tuning [30]	Logit	✓	✗	✓	✓
Steering Vectors [37]	Latent	✗	✓	✗	✗
Patchscopes [13]	Latent	✗	✓	✓	✗
Activation Intervention [35]	Latent	✗	✓	✓	✗
UNLOCK	Latent	✓	✓	✓	✓

Distinction from prior steering transfer work: Most cross-model steering transfer is demonstrated on safety, jailbreak, or style behaviors, where evaluation often relies on coarse proxies (e.g., refusal-string presence), the steering vectors are constructed from explicit positive/negative supervision, and applied to the same model/task. In contrast, we study *capability transfer* across *model sizes and families* and evaluate success using *task-level correctness* on standard reasoning benchmarks. We provide a unified formalization of (i) targeted shifts derived from *prompt- or model-induced* representational differences and (ii) the cross-model alignment required to apply such shifts in a new model.

A.2 Aggregator Functions Used:

In this work, we consider two aggregator functions — the *mean* aggregator, defined as the average of the differences

$$\mathbf{v}_{\psi}^{(\mathcal{S},l,\text{avg})} \;=\; \frac{1}{n}\sum_{i=1}^{n} v_{\psi}^{(\mathcal{S},l,i)}, \tag{2}$$

and the *principal component* aggregator, defined as the first principal component of the centered differences [34]

$$\mathbf{v}_{\psi}^{(\mathcal{S},l,\text{pca})} \;=\; \text{PCA}_1\Big(\Big\{v_{\psi}^{(\mathcal{S},l,i)} - \mathbf{v}_{\psi}^{(\mathcal{S},l,\text{avg})}\Big\}_{i=1}^{n}\Big). \tag{3}$$

A.3 Prompts and Models Used

We show the *Direct* and *CoT* prompts that we used in Figure 5. To avoid discrepancies in prompt templates across models, we use only the two prompt types shown for all experiments. We observe that some post-trained models tend to format outputs to end with "\boxed{ans}". To prevent results from being skewed in favor of such models, we instead use a unified concluding token pattern, "<*atok*>ans</*atok*>", for the final answer.

Reason step by step and give a final answer to the following question. Your response should always end with "The final answer is <atok> [answer] </atok>." where [answer] is the correct solution to the problem.
Question:
Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Answer: *Let's think step by step.*

Solve the following question and place the answer at the end. Your response should always end with "The final answer is <atok> [answer] </atok>." where [answer] is the correct solution to the problem.
Question:
Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?
Answer:

Figure 5: **Example Prompts:** Example *CoT* (green) and *Direct* (red) prompts which are used for all evaluations.

A.4 Definitions of Atomic and Non-Atomic Capabilities

We formally define a capability as:

Definition A.1. *For a model $\mathcal{M}$, a query distribution $\mathcal{Q}$, and a prompt distribution $\mathcal{P}_{\mathcal{M}}$, a **capability** ψ is defined as a measurable behavior associated with a metric $m_\psi(\mathcal{M}; q, p)$. We say the capability ψ is present in $\mathcal{M}$ if the aggregate performance*

$$\mathbb{E}_{q\sim\mathcal{Q},\, p\sim\mathcal{P}_{\mathcal{M}}}\left[m_\psi(\mathcal{M}; q, p)\right] \geq \tau$$

remains significantly above a baseline τ and is robust to minor semantic perturbations in the query and prompt spaces.

Next, we differentiate between atomic and non-atomic capabilities as:

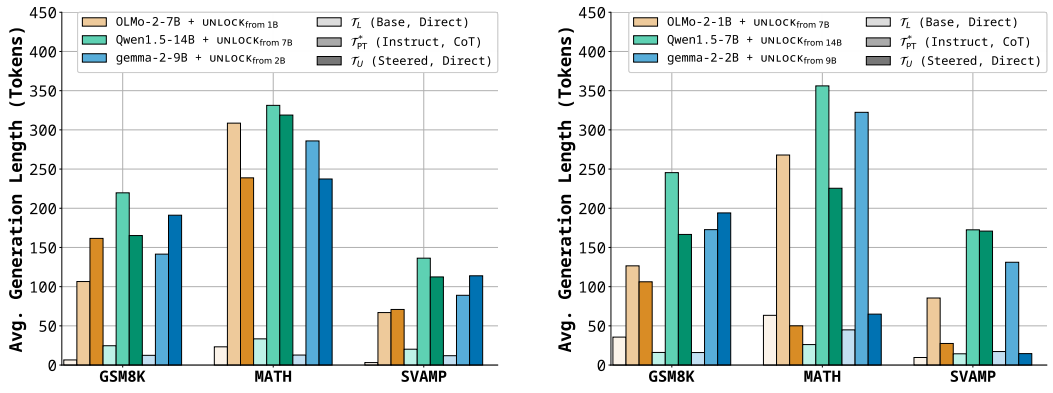
Definition A.2. *Let $\mathcal{M}$ be a base model and $\mathcal{M}^*$ its post-trained variant. Let $q \sim \mathcal{Q}$ and $p \sim \mathcal{P}$ denote queries and prompts drawn from their respective evaluation distributions. For a capability ψ present in $\mathcal{M}^*$ with metric $m_\psi(\mathcal{M}; q, p)$, we define the average post-training gain as:*

$$\Delta_\psi^{\mathcal{Q}} = \mathbb{E}_{q\sim\mathcal{Q},\, p\sim\mathcal{P}}\left[m_\psi(\mathcal{M}^*; q, p) - m_\psi(\mathcal{M}; q, p)\right].$$

*For a small threshold $\epsilon \geq 0$, we say that ψ is an **atomic capability** if $\Delta_\psi^{\mathcal{Q}} \leq \epsilon$ (i.e. the capability is present to nearly the same degree before and after post-training), and a **non-atomic capability** if $\Delta_\psi^{\mathcal{Q}} > \epsilon$ (i.e. post-training yields a substantial improvement that cannot be explained by an already-present behavior).*

Under this view, atomicity is inherently relative to a model's pre-training distribution: a capability is atomic only to the extent that it is supported by the data and objectives encountered during pre-training. In Sections 4, 5 we show that the atomicity of the capability impacts the gains from UNLOCK.

The atomicity of a capability also depends on the learning capacity of the language model and thus would be impacted by size and architecture. While we explore the effects of model scale on transferability (Appendix B), we leave a more systematic study on the impact of architectures to future work. Lastly, atomicity also depends on the nature of the data. In this work we focus on transferring post-training capabilities onto a base model version, and thus we consider the capabilities present within the base model version (i.e. learned during pre-training). When transferring capabilities between two post-trained models, the definitions above should be modified to reflect this change in data distribution. We emphasize that atomicity is a function of not only the capability required, but also the architecture, scale, and data. We intentionally leave Definitions A.1, A.2 vague to reflect this gap in our understanding of the representation space in language models.



(a) Small-to-Large

(b) Large-to-Small

Figure 6: **Increased Generation Length of $\mathcal{T}_U$:** UNLOCK leads to a clear increase in generation length over the base model with Direct prompting, matching the length of the instruction-tuned model with explicit CoT prompts.

B Additional Results for Unlocking Chain of Thought

We provide comparisons of the Unlocked model $\mathcal{T}_U$ to the post-trained model $\mathcal{T}^*_{PT}$, along with additional experiments for `Qwen2.5` and `Qwen3` model families in Table 6[4]

B.1 Impact of Unlocking

Increased Generation Lengths and Task Performance: We find a significant increase in the length of generated answers across all evaluated models in Figure 6. While increased generation length is consistent with step-by-step reasoning, it may also be a result of unhelpful verbosity, such as repetition or hallucination. To assess whether the additional text is task-relevant, we analyze correctness as a function of generation length.

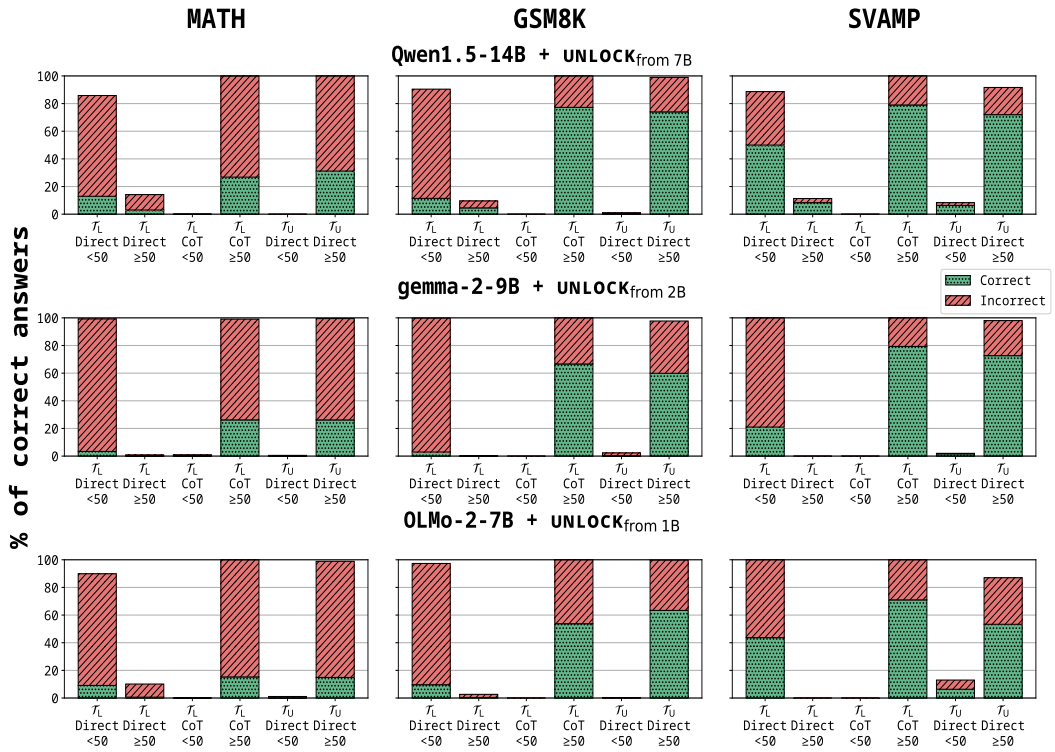
Specifically, we bin outputs based on the number of generated characters for the base model with Direct prompt (i.e. Locked model $\mathcal{T}_L$), the base model with CoT prompt, and the Unlocked model $\mathcal{T}_U$ with Direct prompt. We choose a binning threshold of 50 characters, which corresponds to the length of our response template. Figure 7 displays the percentage of correct solutions in each bin.

We consistently observe an increase in generation under two conditions: (1) as we transition from direct to CoT prompting; and (2) when we move from $\mathcal{T}_L$ to $\mathcal{T}_U$ with Direct prompting. Crucially, this increase in length is accompanied by a higher proportion of correct solutions. This correlation indicates that UNLOCK does not merely append extraneous text but instead elicits meaningful intermediate content that improves downstream performance. We provide qualitative examples illustrating these behavioral shifts in Examples 1–4.

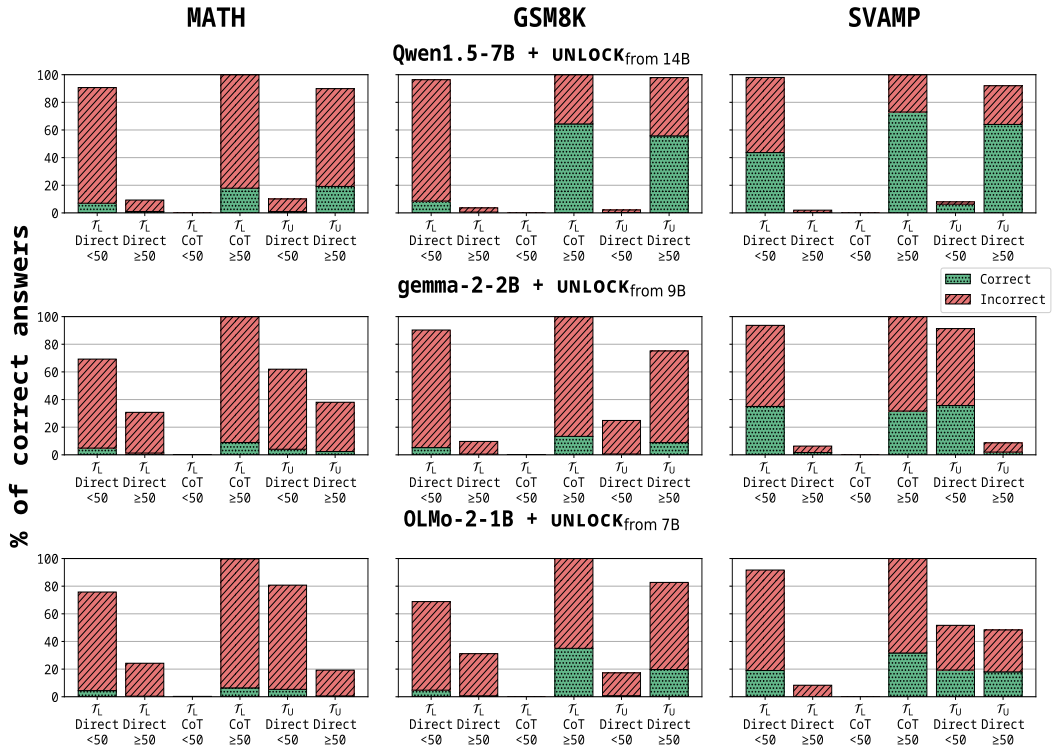
UNLOCK is Non-Destructive & Compliments Gains From Parameter Scaling: In the `Qwen2.5` family, effective CoT usage is present at 1.5B size but the model typically requires explicit CoT prompting to produce intermediate steps (supported by the significant gains from CoT prompts in Table 6). In contrast, the 7B model often produces intermediate steps even under direct prompting. These findings are in line with [62], who show that Chain-of-Thought reasoning emerges with scale.

Across both sizes, we observe that $\mathcal{T}_U$ displays strong reasoning capabilities and performs competitively with ($\pm 2\%$ of) $\mathcal{T}^*_{PT}$. We observe a similar trend in `Qwen-3`, supporting the view that UNLOCK is *non-destructive*: it does not inhibit performance in models where the behavior is reliably displayed, while it reliably elicits the capability when it is present but unused.

[4]We use a maximum generation length of 512 tokens across datasets.



(a) small-to-large



(b) large-to-small

Figure 7: **Evidence for Improved Reasoning:** Generation length of the Unlocked model significantly increases over the Locked model, with a corresponding improvement in downstream performance.

Table 6: **Chain-of-thought Capability Transfer Results.** Performance of UNLOCK across model families. Accuracies of the base model with Direct prompting (Locked model), base model with CoT prompting, and instruction-tuned model with CoT prompting shown in gray.

Model	Prompt	$\mathcal{T}_{\text{L}}$	$\mathcal{S}_{\text{U}} \equiv \mathcal{S}_{\text{L}}$	GSM8K	MATH	SVAMP
Qwen1.5	Direct	7B	–	9.2	8.0	44.0
		14B	–	16.0	16.0	58.3
	CoT	7B	–	64.4	17.9	73.0
		14B	–	77.3	26.8	79.0
	CoT	7B-Chat	–	58.1	18.2	69.7
		14B-Chat	–	74.8	30.2	82.0
	Direct	7B	+UNLOCK from 14B	56.0	20.1	70.3
		14B	+UNLOCK from 7B	74.4	31.2	78.3
OLMo-2	Direct	1B	–	5.5	4.9	19.0
		7B	–	10.0	9.7	43.7
		13B	–	18.7	13.8	65.7
	CoT	1B	–	35.0	6.4	31.7
		7B	–	53.8	15.3	71.0
		13B	–	67.1	20.2	75.7
	CoT	1B-Instruct	–	63.7	16.1	64.0
		7B-Instruct	–	79.4	24.9	78.7
		13B-Instruct	–	80.6	33.8	75.0
	Direct	1B	+UNLOCK from 7B	20.5	5.9	37.3
		7B	+UNLOCK from 1B	63.4	15.1	59.7
		7B	+UNLOCK from 13B	36.1	14.3	58.7
		13B	+UNLOCK from 7B	45.8	16.0	67.3
gemma-2	Direct	2B	–	5.8	6.2	36.7
		9B	–	3.0	3.5	21.0
	CoT	2B	–	13.3	8.9	31.7
		9B	–	66.6	26.4	79.3
	CoT	2B-it	–	60.3	22.7	67.3
		9B-it	–	87.6	43.5	85.3
	Direct	2B	+UNLOCK from 9B	9.5	6.4	37.7
		9B	+UNLOCK from 2B	60.1	26.4	74.3
Qwen2.5	Direct	1.5B	–	11.1	13.5	48.0
		7B	–	85.2	46.1	90.3
	CoT	1.5B	–	67.5	30.8	76.3
		7B	–	87.0	48.8	85.0
	CoT	1.5B-Instruct	–	65.0	26.7	74.7
		7B-Instruct	–	90.4	46.1	91.7
	Direct	1.5B	+UNLOCK from 7B	59.7	31.2	78.0
		7B	+UNLOCK from 1.5B	85.3	46.5	89.3
Qwen3	Direct	4B-Base	–	89.6	51.8	89.7
		8B-Base	–	85.3	50.5	93.3
	CoT	4B-Base	–	84.9	50.5	83.0
		8B-Base	–	89.4	51.6	86.7
	CoT	4B	–	91.1	51.9	92.3
		8B	–	81.6	53.4	86.7
	Direct	4B	+UNLOCK from 8B	89.7	52.2	90.7
		8B	+UNLOCK from 4B	92.4	52.3	93.0

B.2 Hyperparameter Search

B.2.1 Impact of Number of Examples on the Master Key

We now investigate the impact of the number of examples n used in computing the MASTERKEY. Using the same shared prompt $p = P_{(\mathcal{S}_\mathrm{L}, \mathcal{S}_\mathrm{U})}$ and set of queries $\mathcal{D} = \{q_i\}_{i=1}^n$ we stack the final-token hidden states of $\mathcal{S}_\mathrm{L}$ and $\mathcal{S}_\mathrm{U}$ across queries at a fixed layer l:

$$X_{\mathcal{S}_\mathrm{L}} = \begin{bmatrix} \left(\mathbf{h}^{(l)}_{\mathcal{S}_\mathrm{L}}(p \oplus q_1)\right)^\top; \\ \left(\mathbf{h}^{(l)}_{\mathcal{S}_\mathrm{L}}(p \oplus q_2)\right)^\top; \\ \vdots \\ \left(\mathbf{h}^{(l)}_{\mathcal{S}_\mathrm{L}}(p \oplus q_n)\right)^\top \end{bmatrix} \in \mathbb{R}^{n \times d_\mathcal{S}}, \qquad X_{\mathcal{S}_\mathrm{U}} = \begin{bmatrix} \left(\mathbf{h}^{(l)}_{\mathcal{S}_\mathrm{U}}(p \oplus q_1)\right)^\top; \\ \left(\mathbf{h}^{(l)}_{\mathcal{S}_\mathrm{U}}(p \oplus q_2)\right)^\top; \\ \vdots \\ \left(\mathbf{h}^{(l)}_{\mathcal{S}_\mathrm{U}}(p \oplus q_n)\right)^\top \end{bmatrix} \in \mathbb{R}^{n \times d_\mathcal{S}},$$

where $d_\mathcal{S}$ represents the hidden size of the *Source* models. We define the difference matrix $X = X_{\mathcal{S}_\mathrm{U}} - X_{\mathcal{S}_\mathrm{L}}$, where each row represents a per-example steering vector. The corresponding covariance matrix is computed as

$$\Sigma = X^\top X \in \mathbb{R}^{d_\mathcal{S} \times d_\mathcal{S}}.$$

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$ be the eigenvalues of Σ, where $r = \min(n, d_\mathcal{S})$ denotes the maximum possible rank. Following Skean et al. [45], we define the normalized eigenvalues as

$$\tilde{\lambda}_i = \frac{\lambda_i}{\sum_{j=1}^r \lambda_j}, \tag{4}$$

and the spectral entropy as

$$H(\Sigma) = -\sum_{i=1}^r \tilde{\lambda}_i \log \tilde{\lambda}_i. \tag{5}$$

The spectral entropy serves as a measure of the distributional compression of the steering vectors within the latent space. A lower entropy indicates a more *compressed* representation, where a small number of dominant eigenvalues capture the majority of the variance. Conversely, a higher entropy reflects a more *diffuse* representation, where the MASTERKEY is distributed more broadly across multiple orthogonal directions.

Figure 8 illustrates how spectral entropy evolves as a function of the number of examples n. Empirically, we find that spectral entropy plateaus between approximately 1.4 and 2.5 nats across all evaluated datasets. This corresponds to an effective rank in the range 4-12, (since $e^{1.4} \approx 4$ and $e^{2.5} \approx 12$), which is negligible relative to the model's latent dimensionality ($d_\mathcal{S} \geq 1024$ for all models used in this work).

Notably, this extreme compression persists even as n increases, providing strong evidence that the isolated capability resides in a stable, low-dimensional subspace. We further observe that the rate of entropy growth begins to saturate across models and datasets as n increases from 256 to 512, indicating diminishing returns in characterizing the MASTERKEY with sample sizes. However, given the pronounced increase in entropy between $n = 16$ and $n = 64$, we assume that at least 64 examples are required for an accurate and sufficiently complete estimate of the Master Key.

B.2.2 Effect of Rank k and Number of examples n on The Linear Transformation

Next, we evaluate the fidelity of the cross-model alignment by measuring its reconstruction error. Concretely, we run the same set of n queries through the *Source Locked* model $\mathcal{S}_\mathrm{L}$ and the *Target Locked* model $\mathcal{T}_\mathrm{L}$, extract final-token hidden states at layers (l_s, l_t), and fit the low-rank mapping described in Section 3.3. For each query, we map the *Source* hidden state into the *Target* space and compute the ℓ_2 distance to the corresponding ground-truth *Target* hidden state; we report the mean error over the n examples. Figures 9 and 10 show this mapping error as a function of the number of examples n and the transformation rank k, respectively.

Recall that the rank k controls the expressivity of the projection: larger k allows the mapping to preserve and align more directions of variation, whereas smaller k forces the alignment to concentrate

on the most prominent structures shared across the two models. Accordingly, higher rank can, in principle, encode more complex correspondences between latent features, but at the cost of increased sensitivity and a greater risk of overfitting. In contrast, lower rank constrains the mapping to capture only the most dominant and robust shared structure, while prone to underfitting.

Figure 9 shows that in very low-rank regimes (e.g., $k \in \{1, 4\}$), the benefit of increasing the number of examples n rapidly saturates. Specifically, while reconstruction error improves initially, it plateaus as early as $n \approx 64$. Consequently, for highly constrained projections, additional examples do not yield further gains because the mapping lacks sufficient capacity to represent finer structural correspondences; in this regime, the bottleneck is rank rather than sample size.

In contrast, even with an abundance of examples, we find that increasing the rank k does not lead to a monotonic improvement in accuracy. While moderate ranks can reduce reconstruction error effectively, pushing k beyond a threshold consistently degrades performance across models, with this effect becoming pronounced beyond $k \approx 128$ in our experiments (shown in Figure 10). This behavior is characteristic of overfitting: high-rank projections begin to align superficial, example-specific artifacts hindering generalization. These findings provide strong evidence that *capabilities are better captured through low-rank transformations* because they effectively filter out spurious information, and highlights a critical limitation in previous approaches such as [5, 35], which utilize full-rank transformations that are both computationally intensive and prone to capturing noise.

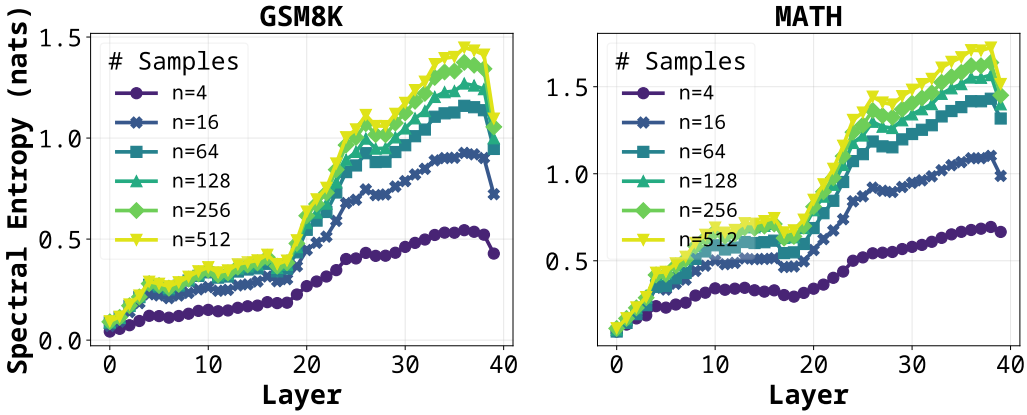
Qualitatively, we observe complementary failure modes at the two extremes. Examples 7–9 illustrate cases where we scale n while keeping k highly constrained. In these instances, although CoT-like behavior is occasionally elicited, it remains fragmented or poorly structured. Conversely, examples 5, 6 demonstrate the emergence of unintended behaviors at high rank; for example, while CoT is induced, it may manifest in an undesired language (e.g., Chinese instead of English).

This tension between the MASTERKEY (which benefits from additional examples) and transformation (which overfits with too many examples) motivates the two regimes for mathematical reasoning transfer introduced in Section 5: the *task-conditioned* setting, which prioritizes in-distribution signals for estimating the MASTERKEY and mapping under limited data, and the *task-agnostic* setting, which leverages abundant (but distribution-mismatched) data to fit a more stable alignment.

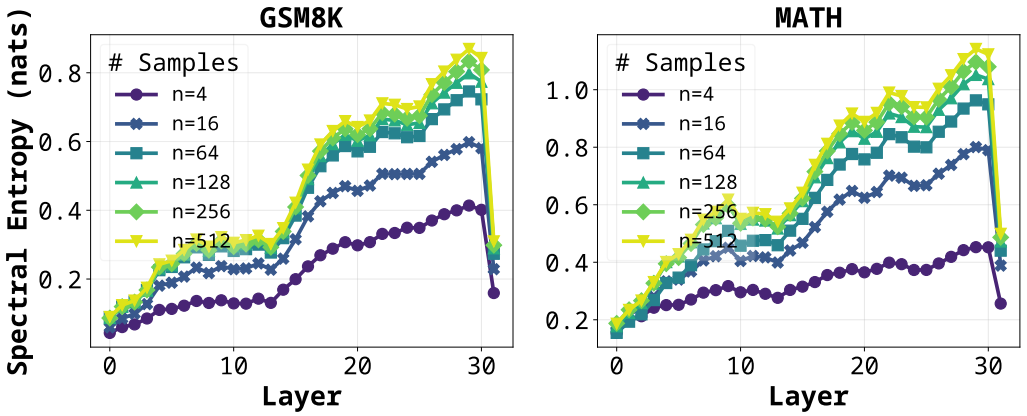
B.2.3 Latent Space Geometry and Sensitivity

Finally, we present topological visualizations of the feature space for `OLMo-2-7B` in Figure 11,12. We find that successful capability transfer typically occurs within localized "pockets" of the latent manifold. This localization highlights the necessity of precise hyperparameter calibration.

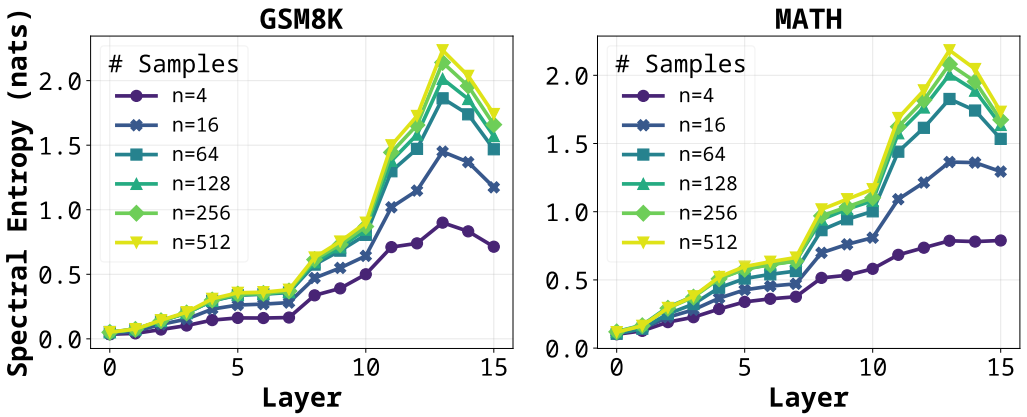
In comparing different extraction strategies, we find that neither the principal component aggregator nor the mean aggregator provides a definitive advantage. Across our benchmarks, the superior method is split approximately evenly, with neither consistently outperforming the other. Ultimately, while subspace matching exhibits sensitivity to the chosen configuration, it yields substantial performance gains when the low-rank projection is well-optimized. We leave a deeper exploration of this hyperparameter landscape to future work.



(a) `Qwen1.5-14B`

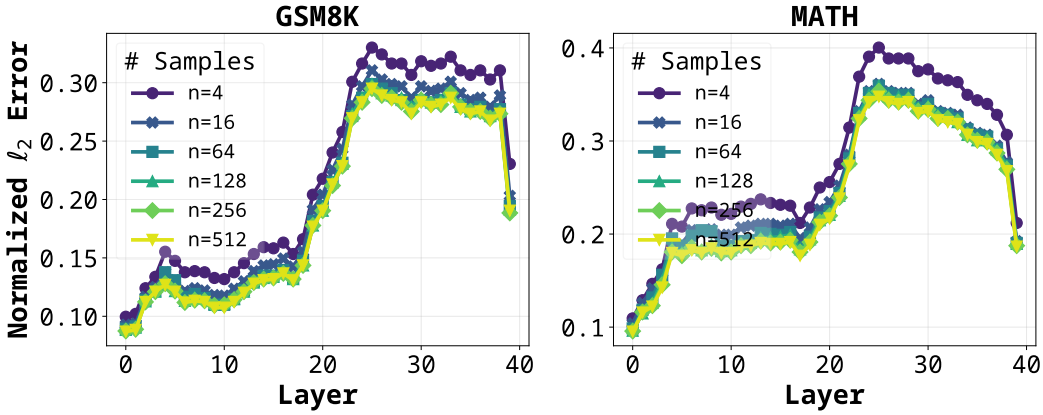


(b) `Qwen1.5-7B`

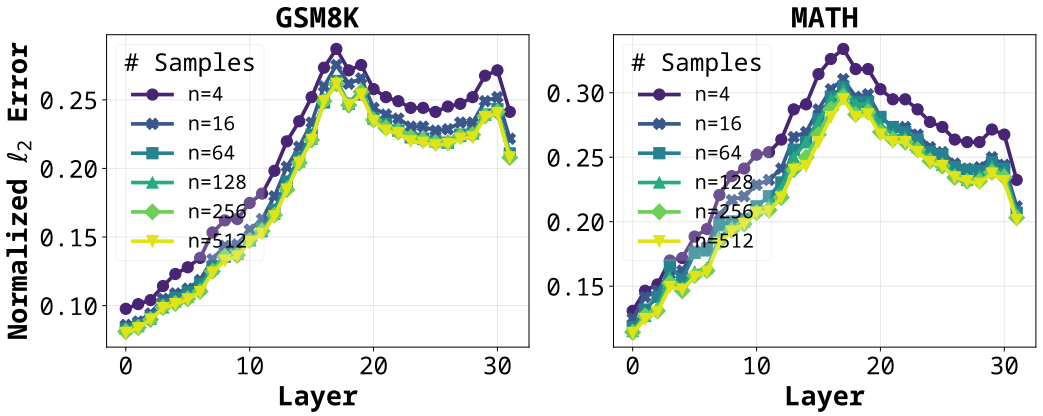


(c) `OLMo-2-1B`

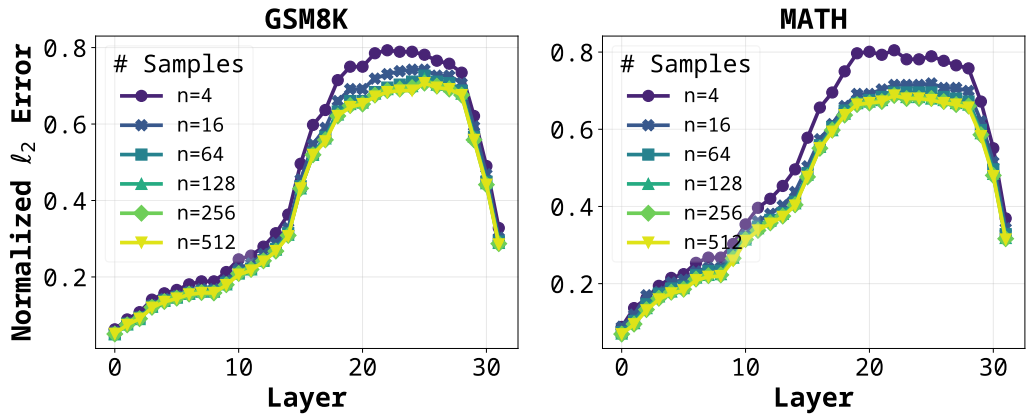
Figure 8: **Spectral Entropy of the Covariance Matrix:** Increasing the number of examples leads to a corresponding increase in entropy — providing evidence that the MASTERKEY captures more information with additional examples.



(a) `Qwen1.5-14B` + UNLOCK$_{\text{from } 7B}$

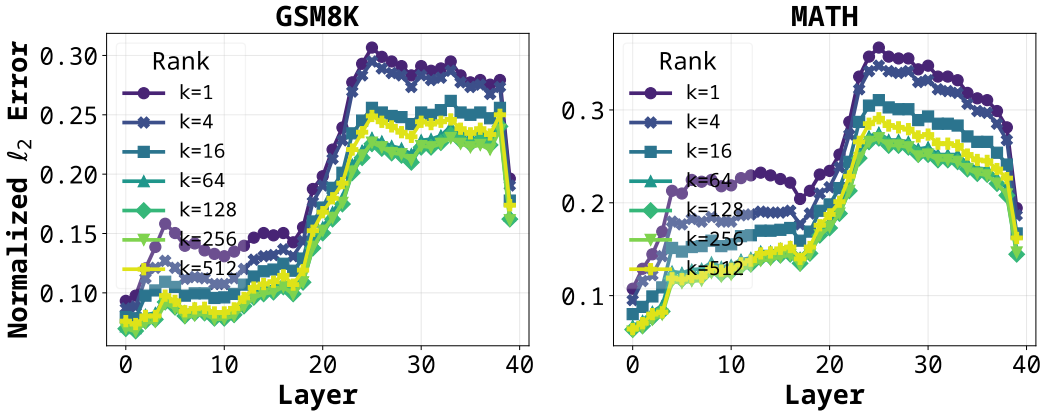


(b) `Qwen1.5-7B` + UNLOCK$_{\text{from } 14B}$

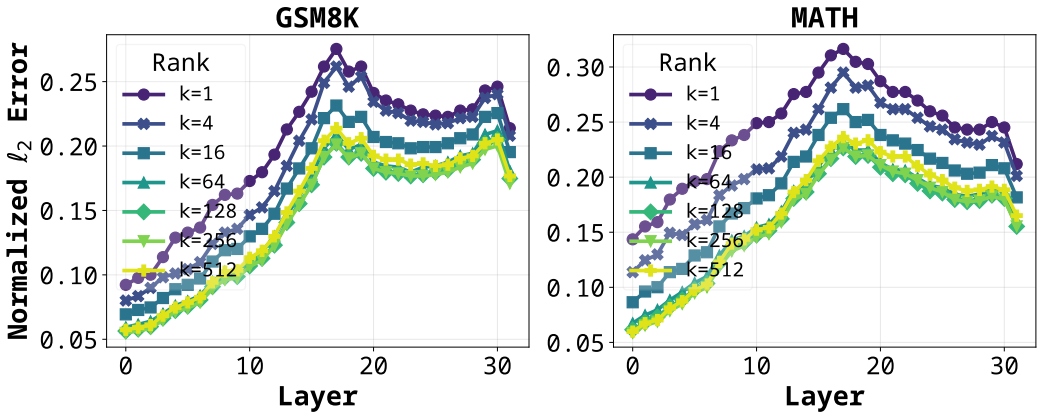


(c) `OLMo-2-7B` + UNLOCK$_{\text{from } 1B}$

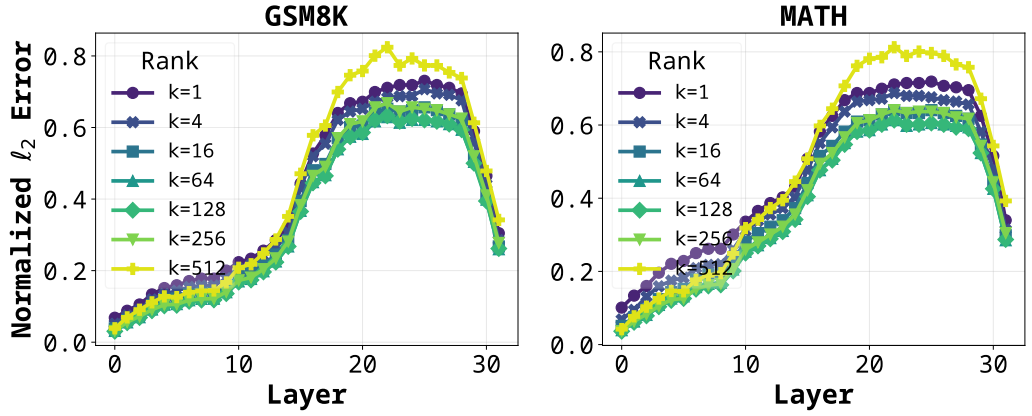
Figure 9: **Convergence in performance of the linear transformation at low ranks:** The normalized ℓ_2 error of the linear mapping as a function of number of samples n with rank $k = 4$ shows the diminishing impact of additional examples in rank-constrained settings.



(a) `Qwen1.5-14B` + UNLOCK$_{\text{from } 7B}$

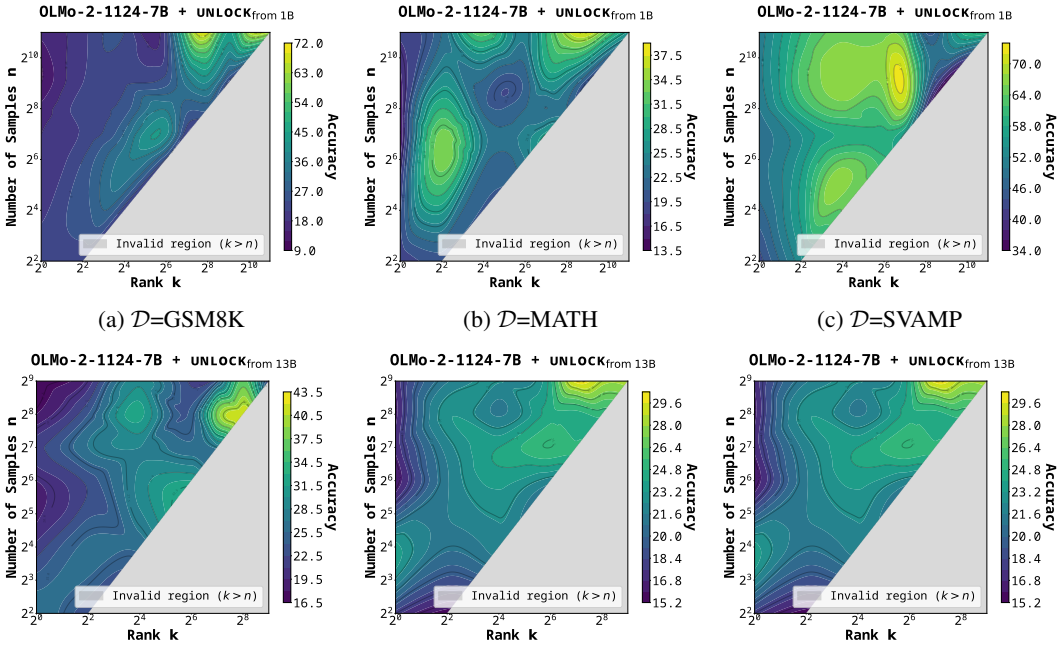


(b) `Qwen1.5-7B` + UNLOCK$_{\text{from } 14B}$



(c) `OLMo-2-7B` + UNLOCK$_{\text{from } 1B}$

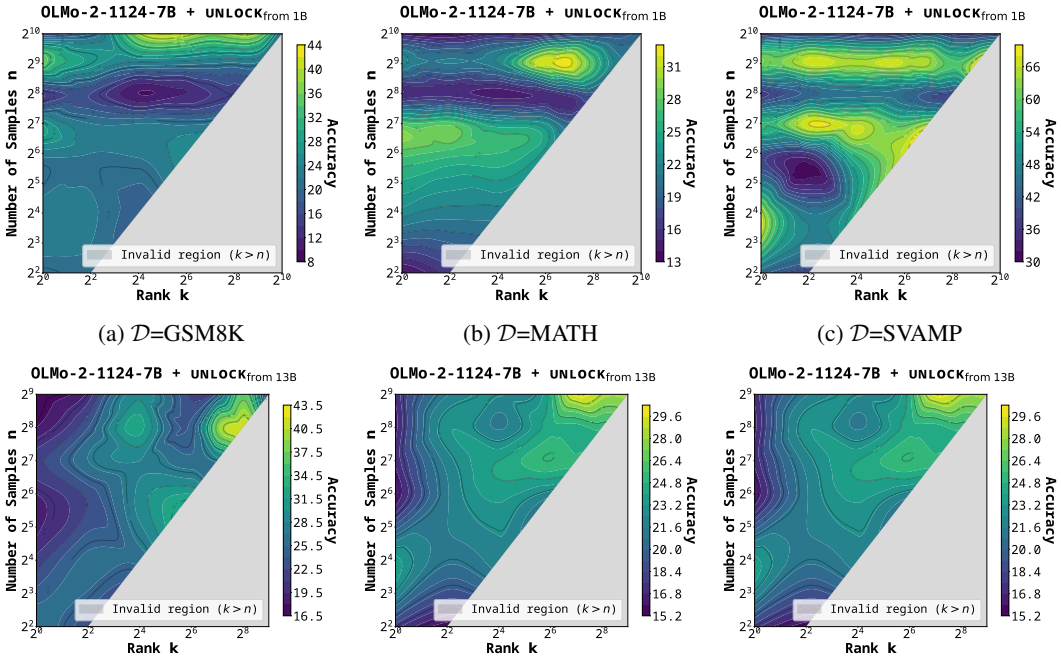
Figure 10: **Overfitting of the linear transformation at high ranks:** The normalized ℓ_2 error of the linear mapping as a function of rank k with number of samples $n = 512$ shows the overfitting of the transformation at high ranks.



(a) $\mathcal{D}$=GSM8K (b) $\mathcal{D}$=MATH (c) $\mathcal{D}$=SVAMP

(d) $\mathcal{D}$=GSM8K (e) $\mathcal{D}$=MATH (f) $\mathcal{D}$=SVAMP

Figure 11: **Representation Space with Φ=Avg:** Performance of `OLMo-2-7B` + $\text{UNLOCK}_{\text{from } 1B}$ (top) and `OLMo-2-7B` + $\text{UNLOCK}_{\text{from } 13B}$ (bottom) with the mean aggregator.



(a) $\mathcal{D}$=GSM8K (b) $\mathcal{D}$=MATH (c) $\mathcal{D}$=SVAMP

(d) $\mathcal{D}$=GSM8K (e) $\mathcal{D}$=MATH (f) $\mathcal{D}$=SVAMP

Figure 12: **Representation Space with Φ=PCA:** Performance of `OLMo-2-7B` + $\text{UNLOCK}_{\text{from } 1B}$ (top) and `OLMo-2-7B` + $\text{UNLOCK}_{\text{from } 13B}$ (bottom) with the principal component aggregator.

Table 7: **Optimal Hyperparameters for CoT Transfer:** The absence of a universal optimal hyperparameter combination across models suggests misalignment in the underlying representation spaces.

$\mathcal{T}_{\text{U}}$	Dataset	Agg. Method	n	k	α
Qwen1.5-7B + UNLOCK$_{\text{from 14B}}$	GSM8K	Avg	512	64	0.1
	MATH	Avg	64	64	0.05
	SVAMP	PCA	64	16	0.1
Qwen1.5-14B + UNLOCK$_{\text{from 7B}}$	GSM8K	PCA	128	4	0.1
	MATH	PCA	512	512	0.1
	SVAMP	PCA	128	128	0.1
OLMo-2-1B + UNLOCK$_{\text{from 7B}}$	GSM8K	Avg	512	128	0.2
	MATH	Avg	512	256	0.05
	SVAMP	Avg	64	64	0.2
OLMo-2-7B + UNLOCK$_{\text{from 1B}}$	GSM8K	Avg	1024	1024	0.5
	MATH	Avg	128	4	0.2
	SVAMP	Avg	256	128	0.2
OLMo-2-7B + UNLOCK$_{\text{from 13B}}$	GSM8K	PCA	512	1	0.1
	MATH	Avg	512	128	0.1
	SVAMP	PCA	16	16	0.1
OLMo-2-13B + UNLOCK$_{\text{from 7B}}$	GSM8K	Avg	64	64	0.1
	MATH	PCA	64	4	0.1
	SVAMP	Avg	256	16	0.05
gemma-2-2B + UNLOCK$_{\text{from 9B}}$	GSM8K	PCA	16	1	0.2
	MATH	PCA	64	64	0.05
	SVAMP	PCA	512	4	0.2
gemma-2-9B + UNLOCK$_{\text{from 2B}}$	GSM8K	Avg	128	64	0.1
	MATH	Avg	512	64	0.1
	SVAMP	PCA	4	1	0.2

Question

James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

Locked Model $\mathcal{T}_{\mathrm{L}}$

The final answer is <atok> 180 </atok>

Unlocked Model $\mathcal{T}_{\mathrm{U}}$

Let's break down the problem step by step:

1. James runs 3 sprints 3 times a week.
2. He runs 60 meters each sprint.

Now, let's calculate the total meters he runs in a week:

Total meters = (Number of sprints) $\times$ (Number of times per week) $\times$ (Distance per sprint)

Total meters = 3 sprints $\times$ 3 times per week $\times$ 60 meters per sprint

Total meters = 9 $\times$ 60

Total meters = 540

The final answer is <atok> 540 </atok>

Example 1: **Successful CoT Transfer** for `Qwen-1.5-14B` + $\textsc{Unlock}_{\text{from } 7B}$

.

Question

Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?

Locked Model $\mathcal{T}_{\mathrm{L}}$

The final answer is <atok> 10 cups of feed </atok>

Unlocked Model $\mathcal{T}_{\mathrm{U}}$

If Wendi feeds each of her chickens three cups of mixed chicken feed every day, and she has 20 chickens, then she needs to feed a total of 20 chickens $\times$ 3 cups of feed = 60 cups of feed each day.

In the morning, she gives her flock of chickens 15 cups of feed, and in the afternoon, she gives them another 25 cups of feed. So far, she has given her chickens a total of 15 cups + 25 cups = 40 cups of feed.

To find out how many cups of feed she needs to give her chickens in the final meal of

the day, we need to subtract the amount of feed she has already given from the total amount of feed she needs to give each day.

The total amount of feed she needs to give each day is 60 cups, and she has already given 40 cups, so she needs to give her chickens 60 cups - 40 cups = 20 cups of feed in the final meal of the day.

The final answer is 20 cups of feed

Example 2: **Successful CoT Transfer** for `Qwen-1.5-14B` + UNLOCK$_{\text{from } 7B}$

Question

Every day, Wendi feeds each of her chickens three cups of mixed chicken feed, containing seeds, mealworms and vegetables to help keep them healthy. She gives the chickens their feed in three separate meals. In the morning, she gives her flock of chickens 15 cups of feed. In the afternoon, she gives her chickens another 25 cups of feed. How many cups of feed does she need to give her chickens in the final meal of the day if the size of Wendi's flock is 20 chickens?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok> 10 cups of feed </atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

If Wendi feeds each of her chickens three cups of mixed chicken feed every day, and she has 20 chickens, then she needs to feed a total of 20 chickens $\times$ 3 cups of feed = 60 cups of feed each day.

In the morning, she gives her flock of chickens 15 cups of feed, and in the afternoon, she gives them another 25 cups of feed. So far, she has given her chickens a total of 15 cups + 25 cups = 40 cups of feed.

To find out how many cups of feed she needs to give her chickens in the final meal of the day, we need to subtract the amount of feed she has already given from the total amount of feed she needs to give each day.

The total amount of feed she needs to give each day is 60 cups, and she has already given 40 cups, so she needs to give her chickens 60 cups - 40 cups = 20 cups of feed in the final meal of the day.

The final answer is 20 cups of feed

Example 3: **Successful CoT Transfer** for `Qwen-1.5-14B` + UNLOCK$_{\text{from } 7B}$

Question

Robin bought a four-scoop ice cream cone having a scoop each of vanilla, chocolate, strawberry and cherry. In how many orders can the four scoops be stacked on the cone if they are stacked one on top of the other and chocolate cannot be on the bottom?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok>12</atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

To solve this problem, we need to find the number of ways to stack the four scoops on the cone, with the condition that chocolate cannot be on the bottom.

First, let's consider the number of ways to stack the scoops without any restrictions. There are 4 choices for the bottom scoop, 3 choices for the second scoop, 2 choices for the third scoop, and 1 choice for the top scoop. So, the total number of ways to stack the scoops without any restrictions is 4 * 3 * 2 * 1 = 24.

Now, let's consider the number of ways to stack the scoops with the condition that chocolate cannot be on the bottom. Since chocolate cannot be on the bottom, there are only 3 choices for the bottom scoop (vanilla, strawberry, or cherry). For the second scoop, there are 3 choices (any of the remaining 3 flavors). For the third scoop, there are 2 choices (any of the remaining 2 flavors). And for the top scoop, there is 1 choice (the remaining flavor).

So, the total number of ways to stack the scoops with the condition that chocolate cannot be on the bottom is 3 * 3 * 2 * 1 = 18.

Therefore, the number of ways to stack the four scoops on the cone with the condition that chocolate cannot be on the bottom is 18.

The final answer is <atok>18</atok>

Example 4: **Successful CoT Transfer** for `Qwen-1.5-14B` + $\text{UNLOCK}_{\text{from } 7B}$

Question

The profit from a business transaction is shared among 2 business partners, Mike and Johnson in the ratio 2:5 respectively. If Johnson got $2500, how much will Mike have after spending some of his share on a shirt that costs $200?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok> $1600 </atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

每份利润为2500/5=500美元。因此，Mike的份额为500*2=1000美元。如果他花了200美元买了一件衬衫，那么他将剩下1000-200=800美元。The final answer is 800.

Example 5: **Failed CoT Transfer** at n=512, k=512 for `Qwen-1.5-14B` + $\text{UNLOCK}_{\text{from 7B}}$

Question

Alexis is applying for a new job and bought a new set of business clothes to wear to the interview. She went to a department store with a budget of $200 and spent $30 on a button-up shirt, $46 on suit pants, $38 on a suit coat, $11 on socks, and $18 on a belt. She also purchased a pair of shoes, but lost the receipt for them. She has $16 left from her budget. How much did Alexis pay for the shoes?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok>$115</atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

每件商品的格如下：

- 衬衫：$30
- 裤子：$46
- 外套：$38
- 袜子：$11
- 腰：$18

Alexis的算$200，她已花了$30 + $46 + $38 + $11 + $18 = $143。她剩下$200 - $143 = $57。她剩下$16，所以她花了$57 - $16 = $41 鞋子。

The final answer is $41

Example 6: **Failed CoT Transfer** at $n = 256$, $k = 256$ for `Qwen-1.5-14B` + $\text{UNLOCK}_{\text{from 7B}}$

Question

Julie is reading a 120-page book. Yesterday, she was able to read 12 pages and today, she read twice as many pages as yesterday. If she wants to read half of the remaining pages tomorrow, how many pages should she read?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok> $20 </atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

$120 - 12 - 24 = 84$ pages. Half of 84 is 42 pages. So, Julie should read 42 pages tomorrow.
The final answer is <atok>42</atok>

Example 7: **Weak CoT Transfer** at $n = 4$, $k = 1$ for `Qwen-1.5-14B` + $\text{UNLOCK}_{\text{from } 7B}$

Question

Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok> 72 </atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

$48 + \frac{48}{2} = 48 + 24 = 72$ clips

The final answer is 72.

Example 8: **Weak CoT Transfer** at $n = 4$, $k = 4$ for `Qwen-1.5-14B` + $\text{UNLOCK}_{\text{from } 7B}$

Question

James writes a 3-page letter to 2 different friends twice a week. How many pages does he write a year?

Locked Model $\mathcal{T}_{\text{L}}$

The final answer is <atok>260</atok>

Unlocked Model $\mathcal{T}_{\text{U}}$

$24 \times 2 \times 52 = 2496$ pages

The final answer is 2496 </atok>

Example 9: **Weak CoT Transfer** at $n = 4$, $k = 4$ for `Qwen-1.5-14B` + $\text{UNLOCK}_{\text{from } 7B}$

C Additional Results for Unlocking Mathematical Reasoning

Our test suite consists of four mathematical reasoning benchmarks: AGIEval-Math [68], Deepmind-Math [42], Minerva-Math [26], and OlympiadBench [16]. We withhold 32 examples from each dataset to use as the dev set for *task-conditioned transfer*. We exclude these examples from the test sets across all settings. For *task-agnostic transfer*, we compute the MASTERKEY and linear transformation using data from MATH [17], and verify the robustness of UNLOCK on Gaokao2023En [28] and AMC23[5]. The best performing hyperparameters are used for evaluating on the test suite.

We investigate four distinct model families: `Qwen2.5` [41], `Qwen3` [51], `Ministral-3` [29], and `gemma-3` [50]. For each family, the base model serves as the Locked variants $\mathcal{T}_{\mathrm{L}}$ and $\mathcal{S}_{\mathrm{L}}$, while a stronger post-trained model is selected as the Unlocked Source model $\mathcal{S}_{\mathrm{U}}$. We categorize these Unlocked models into two classes:

1. *Instruction-tuned models*, optimized for general instruction following and trained with a combination of math, coding, and safety datasets;
2. *Math-specific models*, specialized for math reasoning.

We utilize the corresponding `-Instruct` or `-Chat` checkpoints publicly available on Hugging Face[6] for the instruction-tuned models. For the math-specific models, we employ `NVIDIA-OpenReasoning-Nemotron` [1] and `NVIDIA-DLER-R1` [32] for `Qwen2.5`, and `NVIDIA-Nemotron-Cascade` [59] and `Qwen3-Thinking` [51] for `Qwen3`. We omit `gemma-3` from the math-specific setting as no comparably strong math-oriented post-trained variants were identified for this family.

All models are prompted with the same *CoT* prompt. To reduce model- and dataset-specific variance, we do not apply chat templates or in-context demonstrations. We evaluate with greedy decoding and a maximum generation length of 4096 tokens. We report the results when using instruction-tuned Unlocked models in Table 8 and math-specific models in Table 9.

C.1 Results & Discussion:

Asymmetry in Task Utilization: While both *task-conditioned* and *task-agnostic* transfer improve $\mathcal{T}_{\mathrm{U}}$ over $\mathcal{T}_{\mathrm{L}}$, their relative effectiveness depends on the transfer direction. In the large-to-small setting, we find that *task-conditioned* transfer is superior, outperforming the *task-agnostic* approach in 69.5% of the evaluated configurations. Conversely, for small-to-large transfer, *task-agnostic* transfer yields better results in 70% of the settings.[7]

Consistent with our findings from Section 4, the most substantial performance gains are observed in the small-to-large transfer scenario. These trends suggest that when transferring from larger to smaller models, a precise, task-aligned MASTERKEY is critical for overcoming mismatches in internal circuitry and abilities. In contrast, because larger models likely contain a functional superset of the circuits and capabilities present in smaller models, small-to-large transfer benefits more from a generalizable MASTERKEY, and a stronger and more robust transformation. In this regime, emphasizing general reasoning transfer is more effective than optimizing for task-specific alignment. We leave further analysis into this mismatch and how capabilities arise with scale to future work.

Asymmetry in Performance Gains: Consistent with our findings from Section 4, the most substantial performance gains are observed in the small-to-large transfer scenario. These trends suggest that when transferring from larger to smaller models, a precise, task-aligned MASTERKEY is critical for overcoming mismatches in internal circuitry and abilities. In contrast, because larger models likely contain a functional superset of the circuits and capabilities present in smaller models, small-to-large transfer benefits more from a generalizable MASTERKEY, and a stronger and more robust transformation. In this regime, emphasizing general reasoning transfer is more effective than optimizing for task-specific alignment. We leave further analysis into this mismatch and how capabilities arise with scale to future work.

[5] `https://huggingface.co/datasets/AI-MO/aimo-validation-amc`

[6] `https://huggingface.co/models`

[7] We ignore settings where both methods are within 0.5% of each other.

C.1.1 Understanding the Impact of Unlocking

Dependence on Capabilities present in $\mathcal{T}_{\mathrm{L}}$: We find that the gain of $\mathcal{T}_{\mathrm{U}}$ over $\mathcal{T}_{\mathrm{L}}$ depends not only on the strength of the Source contrast, measured by how much $\mathcal{S}_{\mathrm{U}}$ improves over $\mathcal{S}_{\mathrm{L}}$, but also the baseline competence of $\mathcal{T}_{\mathrm{L}}$. For instance, `gemma-3` is the weakest-performing family in our experiments and underperforms its instruction-tuned counterpart by a wide margin, with average gaps of 32.37% for `gemma-3-4B` and 31.65% for `gemma-3-12B`, leaving limited scope for UNLOCK to recover post-training gains. Accordingly, we observe modest improvements in this setting, and $\mathcal{T}_{\mathrm{U}}$ typically falls well short of $\mathcal{T}^{*}_{\mathrm{PT}}$. Taken together, these results reinforce the interpretation that our method does not introduce new knowledge, but instead *elicits and amplifies* capabilities already present but latent in the Target model.

Although mathematical reasoning is non-atomic by Definition A.2 (as it is not elicited by prompting alone), we find that such capabilities can nonetheless be applied to the Target model as test-time interventions, suggesting that the Target model's latent space may be capable of representing them to some degree.

What is Encoded in the Master Key? We find that gains in accuracy typically arise from three types of changes:

(I.) Coherent reasoning traces: $\mathcal{T}_{\mathrm{L}}$ frequently fails to produce explicit step-by-step reasoning, or instead generates reasoning that is fragmented, inefficient, or prematurely terminated. In contrast, $\mathcal{T}_{\mathrm{U}}$ more consistently produces coherent intermediate steps that connect the problem statement to the final answer. Examples 10 and 11 illustrate this effect.

Figure 13 plots the distribution of first generated words for $\mathcal{T}_{\mathrm{L}}$ and $\mathcal{T}_{\mathrm{U}}$. We find that UNLOCK sharpens the output distribution toward a small set of recurring openings. Across model–dataset pairs, the Unlocked model frequently begins with similar phrases (e.g., "To solve the ..." or "Step 1: ..."). In contrast, $\mathcal{T}_{\mathrm{L}}$ exhibits a more diffuse distribution over opening tokens.

These patterns suggest that UNLOCK increases the likelihood of producing plausible reasoning traces by consolidating representations and reducing variability in early trajectory selection. We leave a more thorough analysis of diversity and mode coverage under Unlocking, and similarity to various post-training methods to future work.

(II.) Improved mathematical reliability: Example 12 highlights cases where both $\mathcal{T}_{\mathrm{L}}$ and $\mathcal{T}_{\mathrm{U}}$ generate step-by-step reasoning yet arrive at different conclusions. $\mathcal{T}_{\mathrm{L}}$ often invokes relevant intermediate concepts but fails to reliably build on them to reach a valid solution. By shifting internal representations during generation, the MASTERKEY increases the probability that the model follows mathematically sound trajectories.

To characterize this effect, we first analyze generation length after unlocking. Because many models can hallucinate or repeat, we measure length only up to the point at which the final answer is produced, and only for outputs marked correct; we refer to this metric as *length-to-answer*. Across tasks and model families, UNLOCK typically increases length-to-answer (with the exception of Minerva Math), indicating that $\mathcal{T}_{\mathrm{U}}$ more often sustains longer, explicit reasoning traces before committing to an answer (Figure 14, left).

For incorrect solutions, we further quantify degeneration by computing the number of repeated substrings as a function of substring length l (Figure 14, middle and right). We find that repetitions peak around $l \in [128, 256]$ characters for solutions marked incorrect, indicating substantial repeated fragments in the generated text. Moreover, $\mathcal{T}_{\mathrm{L}}$ exhibits significantly more repetition than $\mathcal{T}_{\mathrm{U}}$. This provides evidence that UNLOCK reduces repetition and consolidates the model's internal representations, steering generation more successful reasoning patterns.

(III.) More consistent formatting: A common objective of post-training is to enforce stable output formats so that responses can be parsed and evaluated reliably. We observe that $\mathcal{T}_{\mathrm{L}}$ occasionally deviates from the required format (Example 13), likely because it was not explicitly trained to follow a strict response schema. In contrast, $\mathcal{T}_{\mathrm{U}}$ adheres to the expected format more consistently, reducing format violations. *We note that these formatting differences are rarely observed for models larger than 7B, suggesting that at this scale the primary gains from* UNLOCK *stem from improved reasoning behavior rather than format compliance.*

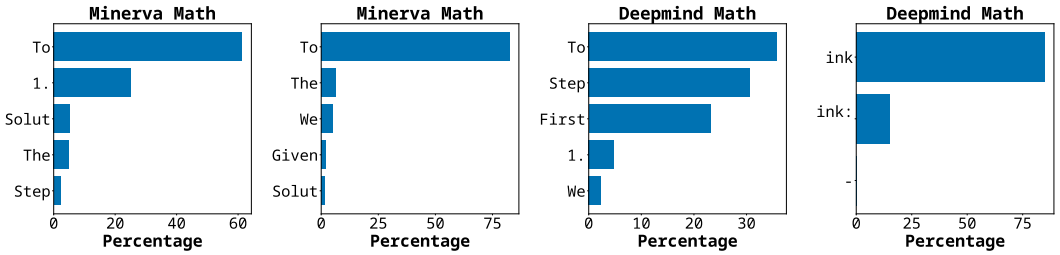
Takeaways: Post-training trains the model to map input prompts to desired outputs, a process that relies on eliciting the combination of capabilities required to produce them. However, these capabilities are often already present within the model and not introduced during post-training. Without this learned mapping, prompting alone is insufficient to elicit them. Instead, UNLOCK exploits the presence of the capabilities in latent space. We find that it is possible to isolate and transfer such capabilities as direct latent interventions, without any training. Put together, these results corroborate our previous findings that UNLOCK is most effective when the desired capability is dormant in the model, and the UNLOCK primarily improves elicitation of the capability rather than injecting new behaviors or information into the model. We leave a more thorough analysis of diversity and mode coverage under latent space capability transfer, and its similarity to other post-training methods to future work.

Table 8: **Mathematical Reasoning Transfer From Instruction-Tuned Unlocked Models $\mathcal{S}_{\mathrm{U}}$:** Performance of the baselines, task-conditioned, and task-agnostic transfer.

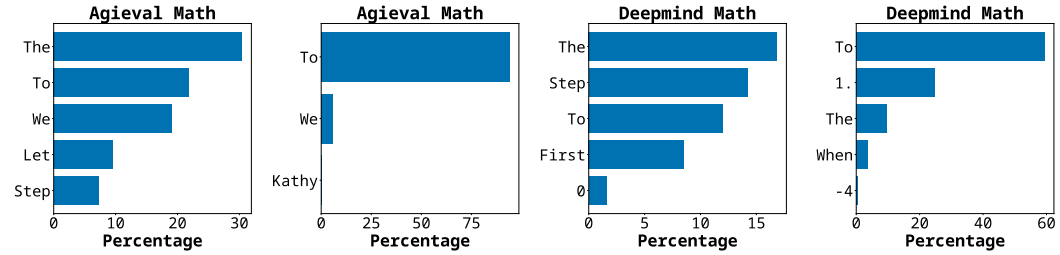
Model	$\mathcal{T}_{\mathrm{L}}$	$\mathcal{S}_{\mathrm{L}}$	$\mathcal{S}_{\mathrm{U}}$	AGI-M	D-M	M-M	OB
Qwen2.5	1.5B	–	–	35.9	45.3	10.8	9.9
	7B	–	–	48.2	67.7	22.5	20.8
	14B	–	–	52.2	70.7	18.5	20.0
	1.5B-Instruct	–	–	37.8	46.6	12.6	13.3
	7B-Instruct	–	–	54.7	71.9	27.5	26.1
	14B-Instruct	–	–	65.7	78.5	29.7	33.8
	1.5B	7B	7B-Instruct	41.4	46.1	16.7	13.6
	1.5B	14B	14B-Instruct	38.6	43.5	16.2	12.3
	7B	1.5B	1.5B-Instruct	52.0	68.8	25.2	22.2
	14B	1.5B	1.5B-Instruct	50.3	73.4	23.9	23.0
	1.5B	7B	7B-Instruct	41.1	45.7	18.0	14.6
	1.5B	14B	14B-Instruct	38.5	49.2	14.9	12.5
	7B	1.5B	1.5B-Instruct	50.0	68.1	24.3	21.8
	14B	1.5B	1.5B-Instruct	55.5	72.7	25.2	23.5
Qwen3	4B-Base	–	–	52.3	71.3	27.5	19.7
	8B-Base	–	–	53.6	77.1	24.3	23.0
	14B-Base	–	–	61.1	78.8	34.7	29.0
	4B	–	–	75.6	88.4	31.5	39.8
	8B	–	–	64.0	77.6	25.2	31.4
	14B	–	–	67.8	80.1	27.9	37.8
	4B-Base	8B-Base	8B	53.1	76.4	29.3	26.6
	4B-Base	14B-Base	14B	58.9	75.8	27.0	26.4
	8B-Base	4B-Base	4B	54.4	73.8	26.1	20.5
	14B-Base	4B-Base	4B	64.1	79.9	31.5	35.4
	4B-Base	8B-Base	8B	52.4	76.5	28.4	21.8
	4B-Base	14B-Base	14B	49.5	72.9	25.7	20.8
	8B-Base	4B-Base	4B	57.6	80.9	27.9	25.1
	14B-Base	4B-Base	4B	71.3	82.4	39.2	36.3
gemma-3	4B-PT	–	–	15.5	14.9	10.8	1.9
	12B-PT	–	–	33.1	48.4	18.9	9.1
	4B-IT	–	–	62.0	74.1	17.7	29.0
	12B-IT	–	–	76.7	85.7	29.7	44.0
	4B-PT	12B-PT	12B-IT	17.4	25.6	7.7	3.0
	12B-PT	4B-PT	4B-IT	33.5	54.7	19.4	9.6
	4B-PT	12B-PT	12B-IT	16.6	25.1	9.0	3.4
	12B-PT	4B-PT	4B-IT	33.7	53.5	20.3	10.1
Ministral-3	3B	–	–	46.9	65.3	26.1	19.0
	8B	–	–	50.7	67.4	29.3	20.0
	$\mathcal{I}$(3B)	–	–	68.7	84.2	26.6	33.9
	$\mathcal{I}$(8B)	–	–	70.6	87.2	29.3	37.0
	3B	8B	8B-Instruct	53.4	66.2	27.5	21.0
	8B	3B	3B-Instruct	51.9	71.3	37.4	20.2
	3B	8B	8B-Instruct	49.9	65.5	27.5	21.0
	8B	3B	3B-Instruct	54.0	70.7	34.7	21.1

Table 9: **Mathematical Reasoning Transfer From Math-specific Unlocked Models $\mathcal{S}_{\mathrm{U}}$:** Performance of the baselines, math post-trained models, task-conditioned, and task-agnostic transfer.

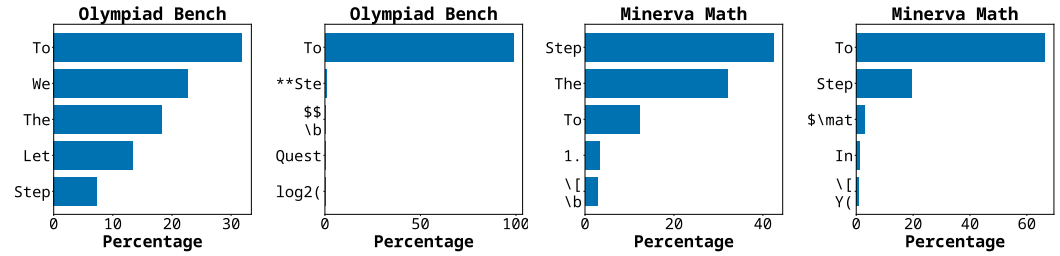
Model	$\mathcal{T}_{\mathrm{L}}$	$\mathcal{S}_{\mathrm{L}}$	$\mathcal{S}_{\mathrm{U}}$	AGI-M	D-M	M-M	OB
Qwen2.5	7B	–	–	48.2	67.7	22.5	20.8
	14B	–	–	52.2	70.7	18.5	20.0
	7B-Instruct	–	–	54.7	71.9	27.5	26.1
	14B-Instruct	–	–	65.7	78.5	29.7	33.8
	Nemotron-14B	–	–	58.1	82.2	10.8	9.1
	DLER-R1-7B	–	–	80.7	88.6	40.5	50.2
	7B	14B	Nemotron-14B	52.8	71.6	21.6	20.2
	14B	7B	DLER-R1-7B	55.5	73.7	24.8	25.4
	7B	14B	Nemotron-14B	50.1	69.9	23.0	21.3
	14B	17B	DLER-R1-7B	58.0	78.2	26.1	26.1
Qwen3	4B-Base	–	–	52.3	71.3	27.5	19.7
	8B-Base	–	–	53.6	77.1	24.3	23.0
	Nemotron-Cascade-8B	–	–	80.1	89.7	36.5	45.1
	4B-Thinking	–	–	60.5	75.2	26.6	36.2
	4B	8B	Nemotron-Cascade-8B	56.1	78.9	27.5	24.5
	8B	4B	4B-Thinking	55.5	80.0	28.4	22.6
	4B	8B	Nemotron-Cascade-8B	54.9	77.2	27.0	23.4
	8B	4B	4B-Thinking	53.6	79.4	30.6	24.3



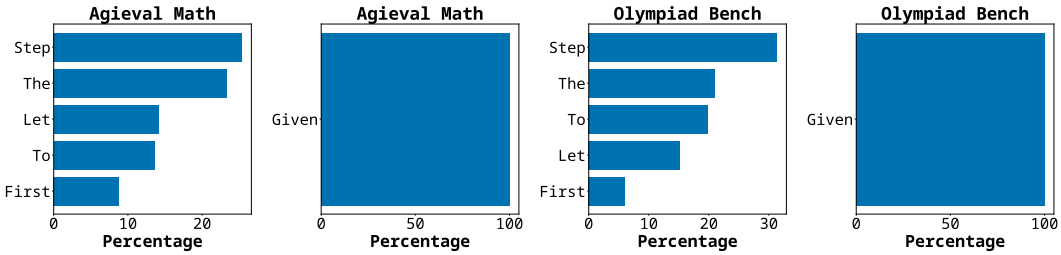
(a) Qwen3-14B (b) Qwen3-14B + UNLOCK$_{\text{from } 4B}$ (c) Qwen3-8B (d) Qwen3-8B + UNLOCK$_{\text{from } 4B}$



(e) Qwen3-4B (f) Qwen3-4B + UNLOCK$_{\text{from } 14B}$ (g) Qwen3-4B (h) Qwen3-4B + UNLOCK$_{\text{from } 14B}$

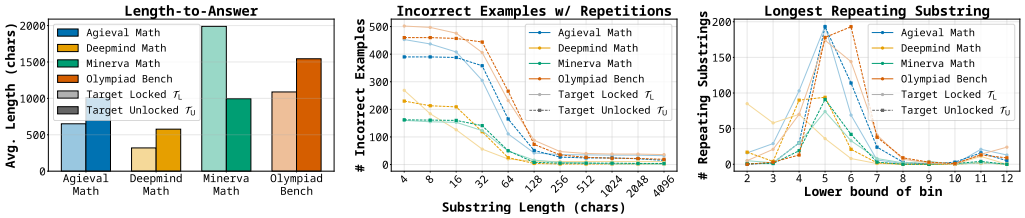


(i) Qwen3-4B (j) Qwen3-4B + UNLOCK$_{\text{from } 8B}$ (k) Qwen2.5-14B (l) Qwen2.5-14B + UNLOCK$_{\text{from } 7B}$

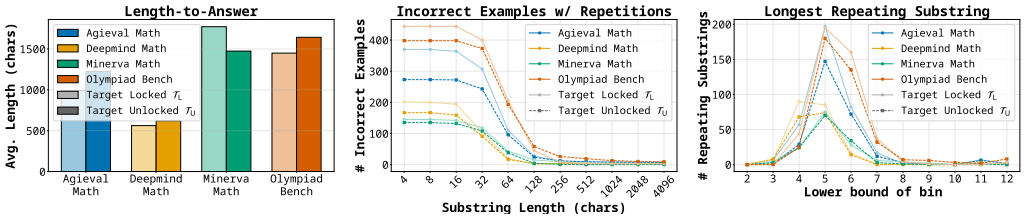


(m) Qwen2.5-14B (n) Qwen2.5-14B + UNLOCK$_{\text{from } 7B}$ (o) Qwen2.5-14B (p) Qwen2.5-14B + UNLOCK$_{\text{from } 7B}$

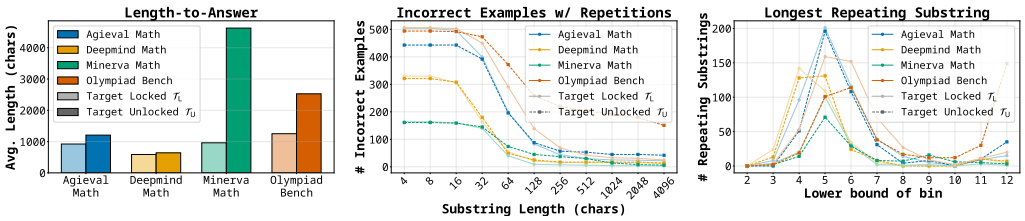
Figure 13: **Additional Statistics of the First-generated Word:** A clear shift in the distribution of first-generated word is observed after applying UNLOCK



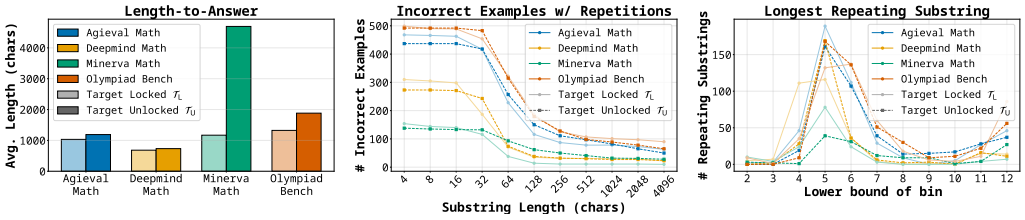
(a) `Qwen3-4B` + UNLOCK$_{\text{from } 14B}$



(b) `Qwen3-14B` + UNLOCK$_{\text{from } 4B}$



(c) `Ministral-3-3B` + UNLOCK$_{\text{from } 8B}$



(d) `Ministral-3-8B` + UNLOCK$_{\text{from } 3B}$

Figure 14: **Length to answer** (left); and **Number of repeating substrings** (middle and right) for `Qwen3` and `Ministral-3` families.

C.2 Examples of Math Reasoning Transfer

Question

Find the solution to the equation $\log_{3x} 4 = \log_{2x} 8$, where x is a positive real number other than $\frac{1}{3}$ or $\frac{1}{2}$.

Locked Model $\mathcal{T}_{\text{L}}$

1. We are given the equation $\log_{3x} 4 = \log_{2x} 8$.
2. To solve this equation, we can use the change of base formula for logarithms: $\log_a b = \frac{\log_c b}{\log_c a}$, where c is any positive real number other than 1.
3. Applying the change of base formula to both sides of the equation, we get $\frac{\log 4}{\log(3x)} = \frac{\log 8}{\log(2x)}$.
4. Now, we can cross-multiply to get rid of the fractions: $\log 4 \cdot \log(2x) = \log 8 \cdot \log(3x)$.
5. We can simplify the equation by using the properties of logarithms: $\log 4 + \log(2x) = \log 8 + \log(3x)$.
6. Using the property $\log a + \log b = \log(ab)$, we can rewrite the equation as $\log(4 \cdot 2x) = \log(8 \cdot 3x)$.
7. Simplifying further, we get $\log(8x) = \log(24x)$.
8. Since the logarithms are equal, their arguments must also be equal: $8x = 24x$.
9. Solving for x, we get $x = \frac{1}{3}$.
10. However, we are given that x is a positive real number other than $\frac{1}{3}$ or $\frac{1}{2}$, so $x = \frac{1}{3}$ is not a valid solution.
11. Therefore, there is no solution to the equation $\log_{3x} 4 = \log_{2x} 8$.

The final answer is <atok> no solution </atok>.

Unlocked Model $\mathcal{T}_{\text{U}}$

To solve the equation $\log_{3x} 4 = \log_{2x} 8$, we will follow these steps:

1. **Express the logarithms in terms of natural logarithms:**

$$\log_{3x} 4 = \frac{\ln 4}{\ln(3x)} \quad \text{and} \quad \log_{2x} 8 = \frac{\ln 8}{\ln(2x)}$$

2. **Set the two expressions equal to each other:**

$$\frac{\ln 4}{\ln(3x)} = \frac{\ln 8}{\ln(2x)}$$

3. **Cross-multiply to eliminate the fractions:**

$$\ln 4 \cdot \ln(2x) = \ln 8 \cdot \ln(3x)$$

4. **Use the properties of logarithms to simplify:**

$$\ln 4 = 2 \ln 2 \quad \text{and} \quad \ln 8 = 3 \ln 2$$

Substituting these into the equation, we get:

$$2 \ln 2 \cdot \ln(2x) = 3 \ln 2 \cdot \ln(3x)$$

5. **Divide both sides by $\ln 2$ (since $\ln 2 = 0$):**

$$2\ln(2x) = 3\ln(3x)$$

6. **Expand the logarithms:**

$$2(\ln 2 + \ln x) = 3(\ln 3 + \ln x)$$

7. **Distribute the constants:**

$$2\ln 2 + 2\ln x = 3\ln 3 + 3\ln x$$

8. **Isolate the terms involving $\ln x$:**

$$2\ln 2 - 3\ln 3 = 3\ln x - 2\ln x$$

$$2\ln 2 - 3\ln 3 = \ln x$$

9. **Solve for x:**

$$\ln x = 2\ln 2 - 3\ln 3$$

$$\ln x = \ln 2^2 - \ln 3^3$$

$$\ln x = \ln 4 - \ln 27$$

$$\ln x = \ln\left(\frac{4}{27}\right)$$

$$x = \frac{4}{27}$$

The final answer is $\boxed{\frac{4}{27}}$.

Example 10: **Improved Coherence in Reasoning Traces:** $\mathcal{T}_{\mathrm{L}}$ and $\mathcal{T}_{\mathrm{U}}$ generate similar reasoning steps up till Step 4 in $\mathcal{T}_{\mathrm{L}}$'s output and Step 3 in $\mathcal{T}_{\mathrm{U}}$'s output. At this point $\mathcal{T}_{\mathrm{L}}$ skips intermediate reasoning steps, and thus makes mistakes in all further steps. On the other hand $\mathcal{T}_{\mathrm{U}}$ continues to reason step by step and arrives at the correct answer

Question

What is 1134508901 to the power of 1/4, to the nearest integer?

Locked Model $\mathcal{T}_{\mathrm{L}}$

To find the fourth root of 1134508901 to the nearest integer, we can follow these steps:

1. **Understand the problem**: We need to find a number x such that x^4 is approximately equal to 1134508901. In other words, we are looking for $x = \sqrt[4]{1134508901}$.

2. **Estimate the value**: Since $10^4 = 10000$ and $100^4 = 100000000$, the fourth root of 1134508901 must be between 100 and 1000. Let's narrow it down further.

3. **Calculate 300^4**:

$$300^4 = (300^2)^2 = 90000^2 = 8100000000$$

This is much larger than 1134508901, so the fourth root must be less than 300.

4. **Calculate 200^4**:

$$200^4 = (200^2)^2 = 40000^2 = 1600000000$$

This is still larger than 1134508901, so the fourth root must be less than 200.

5. **Calculate 150^4**:

$$150^4 = (150^2)^2 = 22500^2 = 506250000$$

This is less than 1134508901, so the fourth root must be greater than 150.

6. **Calculate 180^4**:

$$180^4 = (180^2)^2 = 32400^2 = 1049760000$$

This is close to 1134508901, so the fourth root is likely around 180.

7. **Calculate 185^4**:

$$185^4 = (185^2)^2 = 34225^2 = 1171215625$$

This is slightly larger than 1134508901, so the fourth root must be less than 185.

8. **Calculate 183^4**:

$$183^4 = (183^2)^2 = 33489^2 = 1121088121$$

This is very close to 1134508901, so the fourth root is likely around 183.

9. **Calculate 184^4**:

$$184^4 = (184^2)^2 = 33856^2 = 1146617616$$

This is slightly larger than 1134508901, so the fourth root must be less than 184.

10. **Conclusion**: The fourth root of 1134508901 is between 183 and 184. Since 183^4 is closer to 1134508901 than 184^4, the nearest integer is 183.

The final answer is <atok> 183 </atok>.

Unlocked Model $\mathcal{T}_{\mathrm{U}}$

1. First, we need to find the fourth root of 1134508901.
2. To do this, we can use a calculator or a computer program that can handle large numbers and roots.
3. Using a calculator, we find that the fourth root of 1134508901 is approximately 183.57.
4. Now, we need to round this number to the nearest integer.
5. Since 183.57 is closer to 184 than it is to 183, we round up to 184.

The final answer is <atok> 184 </atok>.

Example 11: **Improved Coherence in Reasoning Traces:** $\mathcal{T}_{\mathrm{L}}$ arrives at approximately the correct answer by making a mistake in floating point estimations. On the other hand, $\mathcal{T}_{\mathrm{U}}$ is more vary of the rounding off specified in the question, and thus reaches the correct solution.

Question

In the n^{th} row of Pascal's Triangle where the first row is $n = 0$, the arithmetic mean of the elements is 51.2. What is the value of n?

Locked Model $\mathcal{T}_{\text{L}}$

1. We know that the sum of the elements in the n^{th} row of Pascal's Triangle is 2^n.
2. The arithmetic mean of the elements in the n^{th} row is the sum of the elements divided by the number of elements, which is $2^n/(n+1)$.
3. We are given that the arithmetic mean is 51.2, so we can set up the equation: $2^n/(n+1) = 51.2$.
4. To solve for n, we can multiply both sides of the equation by $(n+1)$: $2^n = 51.2(n+1)$.
5. Now, we can try different values of n to find the one that satisfies the equation. We can start with $n = 10$ since $2^{10} = 1024$ and $51.2(10+1) = 563.2$, which is close to 1024.
6. We can check if $n = 10$ is the correct value by plugging it back into the equation: $2^{10}/(10+1) = 1024/11 = 93.09$, which is not equal to 51.2.
7. We can try $n = 11$ since $2^{11} = 2048$ and $51.2(11+1) = 614.4$, which is closer to 2048.
8. We can check if $n = 11$ is the correct value by plugging it back into the equation: $2^{11}/(11+1) = 2048/12 = 170.67$, which is not equal to 51.2.
9. We can try $n = 12$ since $2^{12} = 4096$ and $51.2(12+1) = 665.6$, which is closer to 4096.
10. We can check if $n = 12$ is the correct value by plugging it back into the equation: $2^{12}/(12+1) = 4096/13 = 315.08$, which is not equal to 51.2.

...

Unlocked Model $\mathcal{T}_{\text{U}}$

To solve this problem, we need to understand the properties of Pascal's Triangle and how to calculate the arithmetic mean of its elements in the n^{th} row.

Step 1: Understand Pascal's Triangle
Pascal's Triangle is a triangular array of numbers where each number is the sum of the two directly above it. The n^{th} row of Pascal's Triangle corresponds to the coefficients of the binomial expansion $(a+b)^n$. The elements in the n^{th} row are given by the binomial coefficients $\binom{n}{k}$ for $k = 0, 1, 2, \ldots, n$

Step 2: Calculate the Sum of the Elements in the n^{th} Row
The sum of the elements in the n^{th} row of Pascal's Triangle is 2^n. This is because the sum of the binomial coefficients $\binom{n}{k}$ for $k = 0$ to n is equal to 2^n.

Step 3: Calculate the Arithmetic Mean of the Elements in the n^{th} Row
The arithmetic mean of the elements in the n^{th} row is the sum of the elements divided by the number of elements. Since there are $n+1$ elements in the n^{th} row, the arithmetic mean is:

$$\text{Arithmetic Mean} = \frac{2^n}{n+1}$$

Step 4: Set Up the Equation
We are given that the arithmetic mean is 51.2. Therefore, we can set up the equation:

$$\frac{2^n}{n+1} = 51.2$$

Step 5: Solve for n
We need to find the value of n that satisfies the equation. Let's test integer values of n:

- For $n = 6$:

$$\frac{2^6}{6+1} = \frac{64}{7} \approx 9.14 \neq 51.2$$

- For $n = 7$:

$$\frac{2^7}{7+1} = \frac{128}{8} = 16 \neq 51.2$$

- For $n = 8$:

$$\frac{2^8}{8+1} = \frac{256}{9} \approx 28.44 \neq 51.2$$

- For $n = 9$:

$$\frac{2^9}{9+1} = \frac{512}{10} = 51.2$$

Step 6: Conclusion
The value of n that satisfies the equation is 9.

The final answer is <atok> 9 </atok>.

Example 12: **Improved Math Reasoning Capability:** $\mathcal{T}_{\mathrm{L}}$ completely hallucinates, and never reaches the solution, while $\mathcal{T}_{\mathrm{U}}$ is able to generate complete reasoning and the correct solution.

Question

Compute $\begin{pmatrix} 1 & 2 \\ 4 & 8 \end{pmatrix} \begin{pmatrix} 5 \\ 3 \end{pmatrix}$.

Locked Model $\mathcal{T}_{\mathrm{L}}$

To compute the product of the given matrices, we need to follow the rules of matrix multiplication. The first matrix is a 2×2 matrix, and the second matrix is a 2×1 matrix. The result will be a 2×1 matrix.
Step 1: Multiply the first row of the first matrix by the column of the second matrix.

$$(1 \times 5) + (2 \times 3) = 5 + 6 = 11$$

Step 2: Multiply the second row of the first matrix by the column of the second matrix.

$$(4 \times 5) + (8 \times 3) = 20 + 24 = 44$$

The final answer is <atok> $[11, 44]$ </atok>.

Unlocked Model $\mathcal{T}_{\mathrm{U}}$

Step 1: Identify the matrices involved in the multiplication.
Matrix A =

$$\begin{pmatrix} 1 & 2 \\ 4 & 8 \end{pmatrix}$$

Matrix B =

$$\begin{pmatrix} 5 \\ 3 \end{pmatrix}$$

Step 2: Check if the multiplication is possible. Matrix A is a 2×2 matrix, and Matrix B is a 2×1 matrix. Since the number of columns in Matrix A is equal to the number of rows in Matrix B, the multiplication is possible.
Step 3: Perform the matrix multiplication. To multiply the matrices, we need to multiply the elements of each row in Matrix A by the corresponding elements in Matrix B and then sum the results.
Row 1:

$$(1 \times 5) + (2 \times 3) = 5 + 6 = 11$$

Row 2:

$$(4 \times 5) + (8 \times 3) = 20 + 24 = 44$$

Step 4: Write the result as a matrix.
The final answer is <atok>

$$\begin{pmatrix} 11 \\ 44 \end{pmatrix}$$

</atok>.

Example 13: **Improved Formatting:** Both $\mathcal{T}_{\mathrm{L}}$ and $\mathcal{T}_{\mathrm{U}}$ achieve the same answer but $\mathcal{T}_{\mathrm{L}}$ incorrectly formats the final answer. This phenomenon is rarely observed in 14B models (Example taken from Qwen-2.5-7B)

Table 10: **Cross-Family CoT Transfer:** Results when transferring Chain-of-Thought capabilities from the `Qwen1.5` family onto `gemma-2` and `OLMo-2` families. For simplicity, we use $\mathcal{I}(x)$ to denote the instruction-tuned version of the corresponding model. Baseline comparisons shown in gray.

Model	Prompt	$\mathcal{T}_{\text{L}}$	$\mathcal{S}_{\text{U}} \equiv \mathcal{I}(\mathcal{S}_{\text{L}})$	GSM8K	MATH	SVAMP
gemma-2	Direct	9B	–	3.0	3.5	21.0
	CoT	9B	–	66.6	26.4	79.3
	CoT	9B-Instruct	–	87.6	43.5	85.3
	Direct	9B	14B	60.5	24.2	76.7
	Direct	9B	7B	43.7	26.3	72.3
OLMo-2	Direct	7B	–	10.0	9.7	43.7
	CoT	7B	–	53.8	15.3	71.0
	CoT	7B-Instruct	–	79.4	24.9	78.7
	Direct	7B	14B	51.6	15.5	58.7

D Model Family Transfer

D.1 Experimental Setup

We now investigate the efficacy of cross-family transfer, where the *Source* and *Target* models belong to different architectural families. From Section 4, we observed that `Qwen-1.5` family of models possesses CoT as an atomic ability, and consistently demonstrates robust performance across the evaluation settings. We thus select `Qwen1.5` checkpoints as the *Source* models. In contrast to the intra-family configuration used in the CoT setting, we utilize the stronger post-trained variant (`-Chat`) model for $\mathcal{S}_{\text{U}}$, while maintaining all other experimental settings.

D.2 Results & Observations

We report our results in Table 10. First, we observe that *cross-family transfer* can elicit significant CoT behavior from $\mathcal{T}_{\text{L}}$, confirming that Chain-of-Thought capabilities are often latent. Next, we find that cross-family transfer achieves comparable performance to prompting $\mathcal{T}_{\text{L}}$ with a *CoT* prompt. Surprisingly, we find that cross-family transfer performs comparably to intra-family transfer, providing evidence of converging representations of capabilities across models. These results further support our hypothesis that if the Target model contains sufficient representational capacity, it is possible to isolate and apply capabilities and directions in latent space. We leave further exploration into this space, and the more complex challenge of non-atomic capability transfer to future work.

E The Master Key Hypothesis & Implications

We now synthesize our empirical findings into a working hypothesis. Our results show that: (i) latent interventions extracted from Source contrasts can improve downstream behavior in Target models; (ii) transfer is strongest when the Target model already appears to weakly express the relevant capability; and (iii) a low-rank linear alignment is often sufficient to enable this transfer in practice. Taken together, these observations motivate the following operational form of the Master Key Hypothesis (MKH).

Hypothesis E.1. ***The Master Key Hypothesis (MKH):*** *Let $\mathcal{M}_1$ be a model with hidden dimension d_1, and let l_1 denote some layer of $\mathcal{M}_1$. We hypothesize that there may exist a projection $P_1 : \mathbb{R}^{d_1} \to \mathbb{R}^k,\ k \ll d_1$, such that a capability ψ is represented at layer l_1 by a direction $\mathbf{v}_\psi^{(\mathcal{M}_1, l_1)} \in \mathbb{R}^{d_1}$, whose capability-inducing effect is well-captured within the projected subspace $P_1(\mathbb{R}^{d_1})$.*

For a second model $\mathcal{M}_2$ with hidden dimension d_2, and some corresponding layer l_2, there may likewise exist a projection $P_2 : \mathbb{R}^{d_2} \to \mathbb{R}^k$, together with a linear alignment map $f : \mathbb{R}^k \to \mathbb{R}^k$, such that the capability can approximately be transferred across models as

$$P_2\left(\mathbf{v}_\psi^{(\mathcal{M}_2, l_2)}\right) \approx f\left(P_1\left(\mathbf{v}_\psi^{(\mathcal{M}_1, l_1)}\right)\right).$$

where f denotes the low-rank alignment between the corresponding latent subspaces.

In this view, the success of transfer depends on the representational capacity of the model. A capability dormant but present can be isolated and transferred albeit noisily. When a capability is absent in the Source model, there may be no direction to isolate; when it is absent in the Target model, the Target may lack a compatible representational basis to support the transferred intervention.

Our experiments provide three lines of empirical evidence that are consistent with the MKH. First, capability directions extracted from Source models reliably transfer to Target models across scales and multiple architectures, which demonstrates that such directions are not model-specific artifacts. Second, our analysis of the MASTERKEY (Appendix B.2) suggests that the transferable intervention can often be well-approximated in a compact subspace, with effective rank substantially smaller than the hidden size of the model. Moreover, we observe that these interventions stabilize as the number of examples used to estimate them increases, which is consistent with the view that the transferred signal is structured rather than arbitrary noise. Third, transfer efficacy varies predictably with capability atomicity: transfer is strongest when the Target model already appears to contain a latent, though weak or dormant, form of the capability, and substantially weaker when that capability is largely absent.

We find that non-atomic capabilities are also transferrable, if they are well represented in the Source contrast, and the Target model possesses sufficient capacity to represent them in latent space. While we define (non-)atomicity of a capability with respect to its post-training gains and stability across prompts, we find that this does not completely explain our results. For example, a simple capability such as Chain-of-Thought is difficult to transfer perfectly in the `gemma-2` family, while complex math reasoning abilities can be transferred in the `Qwen-3` family. Further, the fact that non-atomic capabilities are transferrable hints at the possibility that they could be represented as a combination of simpler abilities. As such, we believe that Definitions A.1, A.2 are functionally incomplete — the atomicity of a capability should be defined based on how well it can be isolated in latent space, and not by its stability or elicitation in input/output token space. We believe this to be outside the line of this work and leave it to future research.

The Master Key Hypothesis builds on two lines of prior work. The Linear Representation Hypothesis (LRH) [33, 38] suggests that concepts can correspond to consistent directions in representation space *within* a model. The Platonic Representation Hypothesis (PRH) [21] suggests that latent representations may converge *across* models. The MKH unifies these findings at the level of capabilities, arguing that post-training behaviors can often be modeled as *transferable* latent interventions *across* model scales. Our results are consistent with extending these ideas from concepts to behaviors: not only semantic features, but also some capability-inducing interventions, may admit compact and partially transferable latent structure across related models. We emphasize, however, that our results provide empirical support for this view rather than a mechanistic proof of it.

The MKH also offers one possible interpretation of recent findings of [66, 61, 27], which suggest that reinforcement-style post-training often sharpens or re-weights existing output trajectories rather than introducing entirely new knowledge. In our setting, we find that the behavior associated with post-training can sometimes be partially reproduced by transferring a latent intervention (MASTERKEY). This is consistent with the view that certain post-training effects such as mathematical reasoning operate by amplifying pre-existing latent tendencies rather than introducing new representational structure. At the same time, our results also suggest clear limits: such transfer is much less effective for older or weaker models that appear to not possess the necessary representational basis for the desired behavior.

While our findings support the usefulness of the MKH as an empirical abstraction, they do not yet determine the precise mechanism by which capabilities are formed, represented, or interact with each other. The MKH posits the existence of shared low-dimensional subspaces without specifying how they arise from pre-training dynamics or architectural constraints. We therefore view MKH as a useful operational hypothesis that organizes the empirical patterns observed in this work and generates concrete predictions for future study. Just as the Linear Representation Hypothesis motivated subsequent mechanistic work on how concepts are encoded, MKH motivates analogous investigation into how capabilities are learned, organized, and combined in representation space. We leave this to future work.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The claims and numbers reported in the abstract and introduction directly follow from Tables 1, 2.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our approach in Appendix B, C, and the limitations of the Master Key Hypothesis in Appendix E

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The Master Key Hypothesis as presented is built from empirical observations across model sizes, families and evaluation datasets.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Experimental approach, models used, and datasets are clearly mentioned in all sections. The hyperparameters and effects of the grid search are discussed in Appendix B

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [Yes]

Justification: We provide all equations, models, and hyperparameters to replicate our experiments, along with the code.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: All experiment choices are provided in the corresponding section/appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: All experiments were conducted with multiple models and hyperparameters to verify their significance. Additional statistical and robustness analysis ins Sec 5

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [N/A]

   Justification: We ran multiple experiments across clusters involving different GPU/CPU configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The Code of Ethics was followed

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: Our work focuses on mathematical reasoning abilities in language models, and therefore does not have significant societal impacts.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work does not pose such a risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, models, and code were properly cited in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All assets are well documented in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No human subjects involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used in this paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.