

ReasonCACHE: Teaching LLMs To Reason Without Weight Updates

Sharut Gupta^{1,2,*}, Phillip Isola², Stefanie Jegelka^{2,3}, David Lopez-Paz¹, Kartik Ahuja¹, Mark Ibrahim¹, Mohammad Pezeshki¹

¹FAIR at Meta, ²MIT CSAIL, ³TU Munich

*Work done during an internship at Meta

Can Large language models (LLMs) learn to reason without any weight update and only through in-context learning (ICL)? ICL is strikingly sample-efficient, often learning from only a handful of demonstrations, but complex reasoning tasks typically demand many training examples to learn from. However, naively scaling ICL by adding more demonstrations breaks down at this scale: attention costs grow quadratically, performance saturates or degrades with longer contexts, and the approach remains a shallow form of learning. Due to these limitations, practitioners predominantly rely on in-weight learning (IWL) to induce reasoning. In this work, we show that by using *Prefix Tuning*, LLMs can learn to reason without overloading the context window and without any weight updates. We introduce REASONCACHE, an instantiation of this mechanism that distills demonstrations into a fixed key-value cache. Empirically, across challenging reasoning benchmarks, including GPQA-Diamond, REASONCACHE outperforms standard ICL and matches or surpasses IWL approaches. Further, it achieves this all while being more efficient across three key axes: data, inference cost, and trainable parameters. We also theoretically prove that REASONCACHE can be strictly more expressive than low-rank weight update since the latter ties expressivity to input rank, whereas REASONCACHE bypasses this constraint by directly injecting key-values into the attention mechanism. Together, our findings identify REASONCACHE as a middle path between in-context and in-weight learning, providing a scalable algorithm for learning reasoning skills beyond the context window without modifying parameters.

Date: February 3, 2026

Correspondence: sharut@mit.edu, mpezeshki@meta.com

Blogpost: <https://reasoncache.github.io/>



1 Introduction

In-context learning (ICL) represents one of the most remarkable capabilities of modern large language models. By placing demonstrations directly in the prompt, ICL elicits complex behaviors (shifting styles, formats, and task competencies) without a single gradient update (Brown et al., 2020; Min et al., 2022; Agarwal et al., 2024; Eyuboglu et al., 2025; Gupta et al., 2023; Petrov et al., 2023; Wang et al., 2025; Yin et al., 2024). In many regimes, especially when data is limited, this inference-time conditioning often matches or even exceeds task-specific fine-tuning on a range of tasks such as factual question answering, puzzles, summarization or grade school mathematics (Wei et al., 2022; Zhou et al., 2022; Akyürek et al., 2024; Eyuboglu et al., 2025; Agarwal et al., 2024; Lampinen et al., 2025; Yang et al., 2024b; Si et al., 2022; Awadalla et al., 2022).

ICL¹, however, struggles with demanding reasoning tasks; complex mathematical problem-solving and novel algorithmic skills rarely emerge just from a handful of demonstrations. While populating the prompt with more examples appears to be a natural workaround, standard ICL soon hits scaling limits, for three key reasons. 1) It is computationally burdensome and capacity-limited; the finite context window during training bounds how many demonstrations fit, and attention’s quadratic scaling causes inference latency and memory usage to grow rapidly; 2) It is unreliable at length; the advertised window for long context models often

¹Throughout, we use the term *in-context learning* broadly to denote any form of adaptation driven by information provided in the context (examples, instructions, or intermediate reasoning) without updating model parameters.

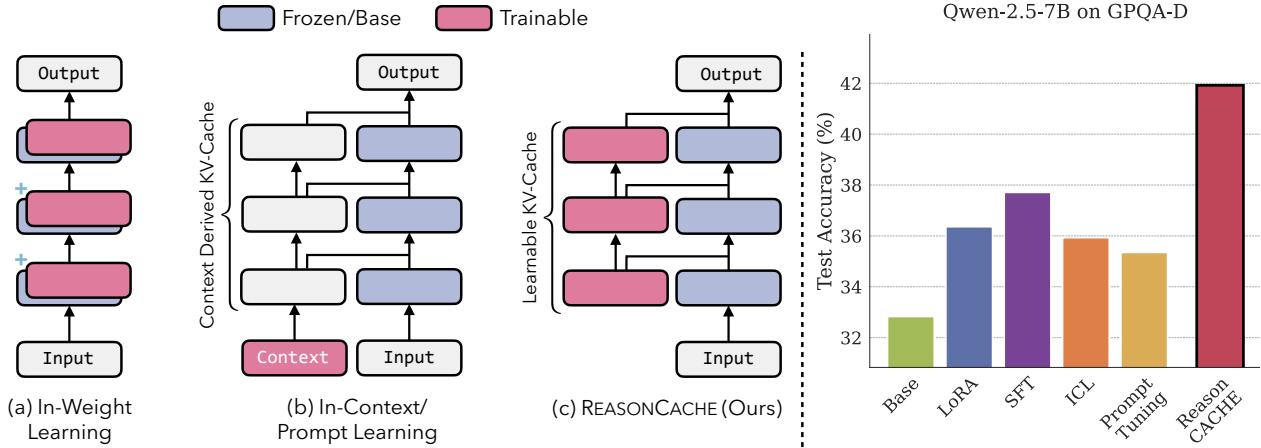


Figure 1 (Left) Comparison of adaptation mechanisms. (a) In-weight learning updates some or all model parameters. (b) In-context learning (ICL) adapts behavior by conditioning on exemplar tokens without parameter updates. (c) REASONCACHE (ours) learns an entire KV cache directly (trainable prefixes) at each attention layer, compressing demonstrations into a compact cached state while keeping the backbone frozen. (Right) Test accuracy (%) of in-context and in-weight baselines, also surpassing full supervised fine-tuning.

overstates how much context the model can use reliably, and performance often saturates and even degrades as relevant information is pushed farther away (Agarwal et al., 2024; Zhang et al., 2025; Zhang and Bottou, 2025; Liu et al., 2024); 3) Most critically, concatenating examples remains a shallow form of adaptation and fails to synthesize the novel reasoning pathways required for complex logic (Guo et al., 2025; Mosbach et al., 2023; de Wynter, 2025; Geng et al., 2024). For these reasons, practitioners typically relegate reasoning to parameter updates, typically through supervised fine-tuning or reinforcement learning, while using standard ICL primarily to steer model behavior rather than to acquire new reasoning capabilities (Muennighoff et al., 2025; Guha et al., 2025; Ye et al., 2025; Yuan et al., 2025; Yu et al., 2023; Yue et al., 2023; Su et al., 2025; Caccia et al., 2025; Kuratov et al., 2025). We revisit this prevailing paradigm and ask whether ICL’s observed limitations are truly fundamental, or instead reflect the particular way in-context supervision is currently represented and scaled. This leads us to the question

How can in-context learning be scaled into a mechanism for reasoning and what are its implications?

In this work, we demonstrate that prefix tuning (PT) (Li and Liang, 2021), an often-overlooked form of in-context adaptation, provides the ideal interface for scaling ICL and answering this question. We refer to this reasoning instantiation of PT as REASONCACHE. Specifically, REASONCACHE learns a small set of prefix key-value vectors at each attention layer while keeping all pretrained weights frozen, so that training compresses the effect of many demonstrations into a compact *cache*.

This scaling has four immediate implications that collectively position learning in-context as a viable route for reasoning. First, LLMs can reliably perform challenging reasoning tasks *without updating pretrained weights and without overloading the context*. Specifically, REASONCACHE improves GPQA-Diamond by up to 11% and outperforms SFT and LoRA. Second, REASONCACHE is markedly more *efficient*, achieving equal or better accuracy with 59% less data and 46% fewer trainable parameters than LoRA on GSM8K. Third, and most consequential for deployment, REASONCACHE yields faster inference with higher accuracy, generating 34% shorter reasoning chains while improving accuracy by 11% over SFT on GPQA-Diamond and translating directly into lower inference compute and monetary expense. Taken together, these results show that REASONCACHE overcomes the scaling limits of exemplar-based ICL, enabling reasoning with frozen models while simultaneously improving data efficiency, reducing the trainable footprint and inference-time cost.

To explain these gains, we theoretically prove that REASONCACHE can be *strictly more expressive* than LoRA.

Low-rank weight updates act through the input sequence, constraining the induced key-value cache by both the adapter rank and input rank, resulting in what we call a *carrier bottleneck*. REASONCACHE bypasses this by injecting learned key-value vectors directly into attention, enabling value-space directions that rank-limited updates cannot realize. We confirm this mechanism empirically, showing that, REASONCACHE increases the effective rank of learned representation by $\sim 20\%$ relative to SFT and LoRA.

To summarize, the key contributions of our work are:

- We show reasoning need not be learned through parameter updates. By distilling in-context data into a fixed key–value prefix, REASONCACHE overcomes the scaling limits of exemplar-based ICL and matches or surpasses in-weight adaptation methods such as LoRA and SFT on challenging reasoning benchmarks (GSM8K, MATH, AIME, GPQA).
- REASONCACHE improves efficiency along three axes without sacrificing accuracy: on GSM8K it needs 59% less *training data* and 46% fewer *trainable parameters* than LoRA, and at inference it boosts accuracy by 44% with 90% less compute than ICL; on GPQA it reduces *generation length* by 34% while improving accuracy by 11% over SFT.
- We prove REASONCACHE can be strictly more expressive than LoRA due to an input-dependent bottleneck in low-rank weight updates, and confirm this empirically by showing a $\sim 20\%$ increase in representation’s effective rank over SFT and LoRA.

2 Prefix Tuning

Our method, REASONCACHE, builds on prefix tuning (PT) (Li and Liang, 2021), which we formalize in this section. In-context learning (ICL) operates by constructing a key–value (KV) cache from the tokens provided in the context: the model processes these tokens through frozen weights, producing keys and values that subsequent tokens attend to. The expressiveness of ICL is therefore limited by what can be represented through the forward pass of raw text under pretrained weights. Moreover, the size of the KV cache is directly determined by the context length. In contrast, PT removes these limitations by making the KV cache itself a learnable object. Instead of deriving keys and values from input tokens, PT directly optimizes auxiliary KV vectors at each attention layer. This section formalizes the PT mechanism and establishes its relationship to ICL and prompt tuning.

2.1 Attention and the KV-Cache

Consider a transformer with L layers and dimension d . Given the representation $H^{(\ell-1)} \in \mathbb{R}^{n \times d}$ from the previous layer (with $H^{(0)}$ denoting the input embeddings), layer ℓ computes queries, keys, and values:

$$Q^{(\ell)} = H^{(\ell-1)}W_Q^{(\ell)}, \quad K^{(\ell)} = H^{(\ell-1)}W_K^{(\ell)}, \quad V^{(\ell)} = H^{(\ell-1)}W_V^{(\ell)}.$$

The attention output in each layer is $Y^{(\ell)} = \text{softmax}\left(Q^{(\ell)}(K^{(\ell)})^\top / \sqrt{d}\right)V^{(\ell)}$. During autoregressive generation, keys and values are cached so that each new token can attend to all previous tokens without recomputation. This *KV-cache* $\mathcal{C} = \{(K^{(\ell)}, V^{(\ell)})\}_{\ell=1}^L$ serves as the model’s working memory: it determines what information is accessible to subsequent tokens.

2.2 The Mechanism of Prefix Tuning

Prefix tuning introduces m trainable key–value pairs $(P_K^{(\ell)}, P_V^{(\ell)}) \in \mathbb{R}^{m \times d} \times \mathbb{R}^{m \times d}$ at each layer ℓ . The augmented keys and values are:

$$\tilde{K}^{(\ell)} = \begin{bmatrix} P_K^{(\ell)} \\ K^{(\ell)} \end{bmatrix}, \quad \tilde{V}^{(\ell)} = \begin{bmatrix} P_V^{(\ell)} \\ V^{(\ell)} \end{bmatrix}.$$

Each query $q_i^{(\ell)}$ attends jointly to prefix and token-derived keys, producing an output that is a convex combination over all values:

$$\tilde{y}_i^{(\ell)} = \sum_{p=1}^m \alpha_{ip} (P_V^{(\ell)})_p + \sum_{j=1}^n \alpha_{i,m+j} v_j^{(\ell)},$$

where $\alpha_{ip}, \alpha_{i,m+j} \geq 0$ are the attention weights (summing to one across each row). Throughout, all pretrained weights remain frozen; only the prefix parameters $\mathcal{P} = \{(P_K^{(\ell)}, P_V^{(\ell)})\}_{\ell=1}^L$ are optimized.

Remark 1. The prefix values $(P_V^{(\ell)})_p$ are free parameters that can point in *any* direction in \mathbb{R}^d . In contrast, token-derived values $v_j^{(\ell)}$ must lie in the rowspace of $H^{(\ell-1)}W_V^{(\ell)}$. This geometric freedom is the source of PT’s enhanced expressivity, which we formalize in [Section 4](#).

Remark 2. The prefix influences the model at two scales. Within each layer, prefix values contribute to the attention mixture. Across layers, the attention output (which includes contributions from the prefix) becomes the hidden representation from which subsequent keys and values are derived. Thus deeper layers see token representations that have already been shaped by prefix vectors at earlier layers.

2.3 Relationship to ICL and Prompt Tuning

Prefix tuning can be understood as a generalization of two related methods:

- **In-context learning.** In ICL, demonstration tokens X_{demo} are prepended to the input and processed through frozen weights, producing a demonstration cache $\mathcal{C}_{\text{demo}} = \{(K_{\text{demo}}^{(\ell)}, V_{\text{demo}}^{(\ell)})\}_{\ell=1}^L$. This is equivalent to PT with $P_K^{(\ell)} = K_{\text{demo}}^{(\ell)}$, $P_V^{(\ell)} = V_{\text{demo}}^{(\ell)}$, and no optimization.
- **Prompt tuning.** Prompt tuning ([Lester et al., 2021](#)) learns continuous embeddings $E \in \mathbb{R}^{m \times d}$ prepended at the input layer *only*, which then propagate through the frozen network to produce KV vectors at each layer. This is equivalent to PT where the prefix vectors are constrained to be outputs of the frozen model on E , rather than free parameters.

The key distinction is that PT optimizes KV vectors at each layer independently, bypassing the frozen projections. This gives PT strictly greater representational freedom as it can produce prefix vectors that no input embedding could generate.

2.4 Training and Inference

The prefix parameters $\mathcal{P} = \{(P_K^{(\ell)}, P_V^{(\ell)})\}_{\ell=1}^L$ are optimized to minimize the standard next-token prediction loss over a dataset $\mathcal{D} = \{D_1, \dots, D_M\}$ of M sequences:

$$\min_{\mathcal{P}} \sum_{j=1}^M \mathcal{L}(D_j \mid \mathcal{P}), \quad \mathcal{L}(D_j \mid \mathcal{P}) = - \sum_{t=1}^{|D_j|} \log p_{\theta}(x_t^{(j)} \mid x_{<t}^{(j)}, \mathcal{P}), \quad (1)$$

where θ denotes the frozen model parameters.

Initialization. Prefixes can be initialized randomly or from demonstration KV-caches, truncated or padded to the prefix length m . Alternatively, a trainable auxiliary network $f_{\phi} : \mathbb{R}^d \rightarrow \mathbb{R}^{2Ld}$ (e.g., an MLP) can generate the prefix for a fixed random input; this reparameterization often stabilizes optimization ([Li and Liang, 2021](#)). At inference, the prefix is computed once and cached, and f_{ϕ} can be discarded.

The formulation above presents PT as a general mechanism for injecting learned key–value vectors into the attention computation of a frozen transformer. In the rest of this paper, we use REASONCACHE to denote PT specialized to reasoning. In other words, REASONCACHE introduces no new adaptation primitive; rather, it studies and demonstrates how learned KV prefixes can scale in-context learning into a reliable and efficient mechanism for reasoning.

3 Experimental Results

Our experiments address the following key question: *Can reasoning capabilities traditionally learned via in-weight adaptation instead be acquired through in-context mechanisms, without modifying pretrained model parameters and without overloading the context?* We answer this question by evaluating REASONCACHE across two primary dimensions: (1) *accuracy* on challenging reasoning benchmarks, comparing against both in-context and in-weight baselines (Section 3.2); and (2) *efficiency* across three axes: data efficiency (Section 3.3), inference efficiency (Section 3.4), and parameter efficiency (Section 3.5). Ablation studies on design choices are deferred to Appendix C.1.

3.1 Experimental Setup and Datasets

Models and Datasets. We evaluate methods across both short- and long-form reasoning tasks. For short-reasoning tasks, we adapt a LLaMA-2 on MetaMathQA (Yu et al., 2023) and evaluate on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2020). For long-form reasoning, we adapt a Qwen-2.5-7B-Instruct (Yang et al., 2024a) on a filtered subset of OpenThoughts-3 (Guha et al., 2025) and evaluate on GPQA-Diamond (Rein et al., 2024) and AIME (MAA, 2024, 2025). We construct the OpenThoughts-3 subset by retaining only examples whose reasoning traces fit within a 4096-token budget, matching the controlled-context setting of prior work (Guha et al., 2025). At inference, we cap model “thinking” to 4096 tokens via the decoding-time intervention of Muennighoff et al. (Muennighoff et al., 2025). Concretely, we force termination of the reasoning phase by appending the end-of-thinking delimiter and “*Final Answer:*”, prompting the model to output its current best solution under the imposed constraint. For more details, refer to Appendix B.3.2. Across all datasets, we follow the standard train/eval splits provided by the Language Model Evaluation Harness (lm-evaluation-harness) (Gao et al., 2024).

Baselines. We compare REASONCACHE against standard *in-weight* baselines (supervised fine-tuning (SFT) and LoRA) and *in-context* baselines (exemplar-based in-context learning (ICL) and prompt tuning). For LoRA, we apply adapters to the attention key-value projections and sweep ranks $r \in \{2^k\}_{k=0}^7$. For REASONCACHE and prompt tuning, we sweep the number of virtual tokens $m \in \{2^k\}_{k=0}^{10}$.

Training and Evaluation Setup. We train all methods with AdamW and a cosine learning-rate schedule (warmup ratio 0.05), using zero weight decay. On OpenThoughts-3, we train for 13 epochs with batch size 32 and sequence length 8192; on MetaMathQA, we train for 3 epochs with batch size 128 and sequence length 2048. At inference, we evaluate with lm-evaluation-harness under greedy decoding (temperature 0) and report exact-match accuracy (pass@1). Unless stated otherwise, methods share the same base model, tokenizer, training examples, optimization budget, and decoding configuration, and we select hyperparameters by best validation performance before reporting final test accuracy.

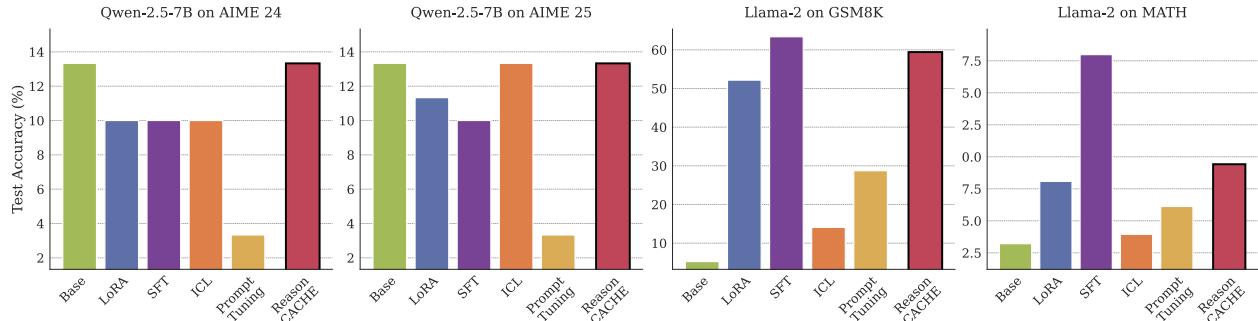


Figure 2 Test accuracy (%) of in-context and in-weight adaptation methods across AIME 24/25, GPQA-Diamond (shown in Figure 1), GSM8K and MATH. REASONCACHE consistently outperforms in-context baselines (ICL, Prompt Tuning) and exceeds LoRA while keeping pretrained weights frozen. On GPQA-Diamond, REASONCACHE surpasses full supervised fine-tuning (SFT).

3.2 ReasonCache Enables Effective Reasoning in LLMs Without Modifying Weights

ReasonCache Scales In-Context Learning: Figure 1(right) and Figure 2 summarize final test accuracy across all benchmarks. We first examine whether REASONCACHE meaningfully extends the capabilities of standard ICL. We observe a clear limitation of *shallow* contextual adaptation: both ICL and prompt tuning yield only modest gains, particularly on tasks requiring extended reasoning. In contrast, REASONCACHE consistently outperforms other in-context methods.

ReasonCache Outperforms In-Weight Adaptation on Reasoning Benchmarks. We next compare REASONCACHE against IWL methods. Even under matched parameter budgets, REASONCACHE matches or beats LoRA and even full SFT (on GPQA-Diamond). Specifically, when trained on OpenThoughts-3, REASONCACHE achieves the best performance on GPQA-Diamond (41.92%), exceeding both LoRA and full SFT.

This advantage, however, does not seem to extend to AIME 24 and 25: all methods remain near the baseline ($\sim 13\%$). IWL even hurts performance ($\sim 10\%$), while REASONCACHE preserves baseline accuracy. We attribute this to our 4096-token filtering, which likely underrepresents the extended reasoning that competition mathematics demands.

Having established that REASONCACHE matches or exceeds IWL methods in absolute accuracy, we next evaluate efficiency across three dimensions: how much *data* is required during learning (data efficiency), how much *compute* is needed at inference (inference efficiency), and how many *parameters* must be stored (parameter efficiency). Across all three, REASONCACHE exhibits superior accuracy-cost tradeoffs: it uses 59% less data than LoRA to reach the same accuracy, achieves 90% lower inference compute than ICL at higher accuracy, and requires 46% fewer parameters than LoRA while maintaining better performance.

3.3 ReasonCache is Data Efficient: Pushes the Accuracy vs Data Frontier

REASONCACHE combines ICL’s data efficiency with the scalability of IWL methods. In Figure 3, REASONCACHE traces the Pareto frontier of accuracy versus training dataset size across nearly four orders of magnitude. In the low-data regime, REASONCACHE matches or exceeds ICL. Initializing REASONCACHE from few-shot exemplars preserves the inductive bias of ICL, and subsequent optimization enables further gains beyond the initial few-shot performance. In contrast, however, ICL degrades as more examples are added (longer contexts dilute attention over exemplars and exacerbate position effects), consistent with reported long-context issues in the literature (Agarwal et al., 2024; Zhang et al., 2025; Zhang and Bottou, 2025; Liu et al., 2024).

IWL methods exhibit the opposite failure mode: LoRA and SFT tend to overfit with limited data (often mitigated by heavy augmentation (Yang et al., 2024b; Eyuboglu et al., 2025) or large-scale training). REASONCACHE is

$\sim 59\%$ more data-efficient than LoRA (to reach 50% accuracy), consistently outperforming it throughout this regime and closing the gap to high-data SFT, despite keeping the pretrained backbone frozen. This places REASONCACHE uniquely on the ICL–IWL spectrum: it inherits *ICL’s data efficiency* and inductive bias via few-shot initialization, while gaining the *data-scaling benefits of IWL methods* through gradient-based optimization. REASONCACHE therefore can be a bridge between in-context and in-weight adaptation.

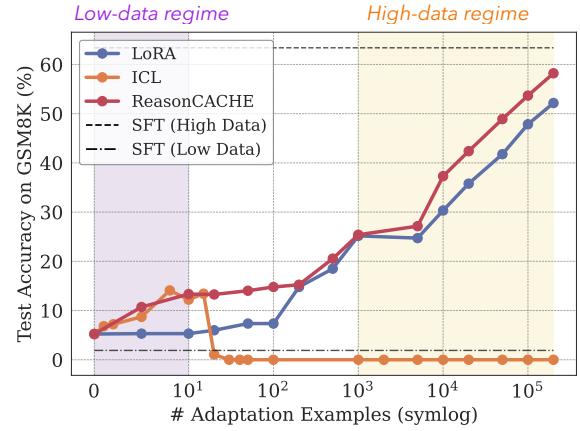


Figure 3 Accuracy as a function of training examples. REASONCACHE matches ICL’s data efficiency at low data and continues to scale with additional data like in-weight methods.

3.4 ReasonCache is Efficient At Inference: Pushes the Accuracy vs Compute Frontier

A fundamental advantage of REASONCACHE is that it decouples the learning budget (i.e., number of adaptation examples) from the inference cost. In ICL, the adaptation data must be included in the context at inference.

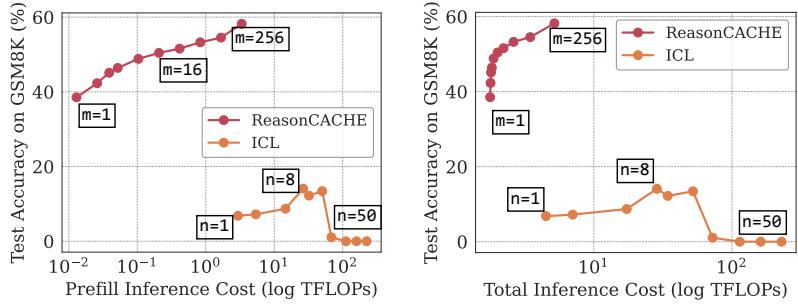


Figure 4 Accuracy as a function of inference cost measured in TFLOPs. *Left:* Prefill cost only. *Right:* Total inference cost (prefill + decoding). REASONCACHE consistently dominates in-context learning, achieving better accuracy at substantially lower inference compute across both metrics. n and m denote the number of in-context example, and the number of tokens allocated to for REASONCACHE , respectively.

Even a single long reasoning example (e.g., 10k tokens) directly increases the prefill cost quadratically and decoding cost linearly. REASONCACHE removes this coupling by learning from hundreds of thousands of tokens, while using only a short compact learned prefix (length m) at inference. Since prefill cost scales quadratically with context length, replacing long exemplar prompts with a short prefix yields substantial savings.²

ReasonCache Reduces Prefill Cost. We first consider GSM8K, where short reasoning chains make inference cost dominated by prefill. Figure 4 reports accuracy as a function of inference compute (measured in TFLOPs) for prefill (left) and prefill+decoding (right) on GSM8K. REASONCACHE dominates ICL, outperforming the best ICL configuration by 44.8 points while using 90% less total inference compute. This gap is driven by prefill costs, since ICL incurs quadratic overhead as demonstrations are added, whereas REASONCACHE replaces them with a short learned prefix. We note that IWL methods incur no additional prefill overhead at all since they modify weights rather than context. However, as shown next, REASONCACHE’s shorter generations offset its modest prefill cost, yielding better overall inference efficiency.

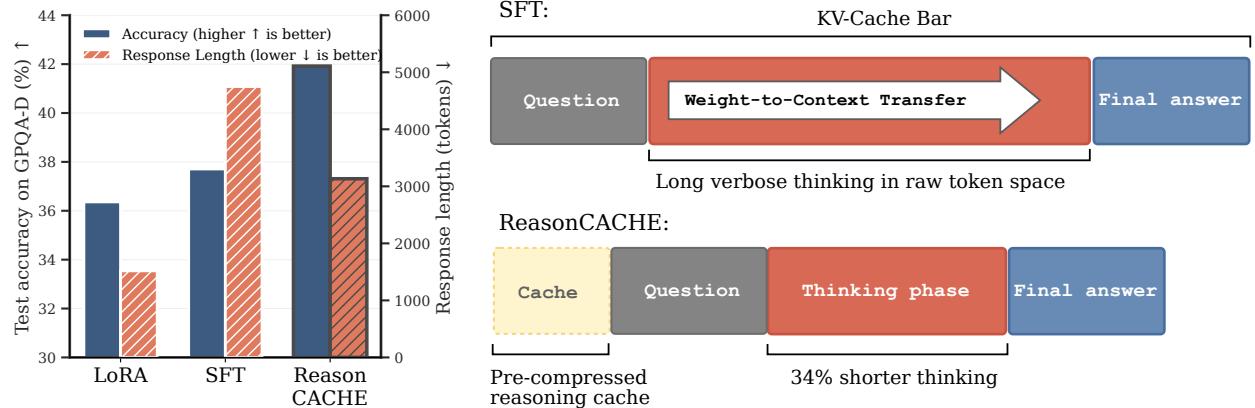


Figure 5 (Left): On GPQA-Diamond, REASONCACHE achieves 11% higher accuracy than SFT while generating 34% shorter responses. LoRA produces the shortest responses but at the cost of lower accuracy. (right): We hypothesize that SFT externalizes weight-encoded procedural knowledge into explicit context during generation. Although the model has internalized reasoning strategies through weight updates, it nonetheless regenerates these patterns as explicit token sequences at inference time, leading to unnecessary verbosity. Our conjecture is that in contrast, REASONCACHE stores procedural knowledge directly in the KV cache, eliminating the need for explicit externalization.

ReasonCache Reduces Generation Length. On long-form reasoning benchmarks such as GPQA-Diamond, models often generate thousands of reasoning tokens, causing decoding to dominate the inference cost. Interestingly, REASONCACHE attains higher accuracy while producing substantially shorter generations (Figure 5). Compared to SFT, REASONCACHE reduces generation length by 34% while improving accuracy by about 11%. LoRA produces shorter outputs but underperforms noticeably on this benchmark. We hypothesize that the learned prefix likely encodes task guidance; instead of spending tokens to re-derive the approach step-by-step, the model conditions on this information during the forward pass and can proceed more directly

²For prompt length S and generation length T , prefill cost is $O(S^2)$ and decoding cost is $O(T(S + T))$.

to the answer. In-weight methods, by contrast, must “unpack” their learned knowledge from weights to the context as shown in Figure 5 (right). As we formalize in Section 4, weight updates influence the KV-cache only through the input tokens, so exploiting learned knowledge requires sufficiently rich context in input.

3.5 ReasonCache is Parameter Efficient: Pushes Accuracy vs Parameter Frontier

Finally, we examine *parameter efficiency*, which captures the cost of *storing* and transferring an adaptation once it has been learned. Under matched parameter budgets, REASONCACHE consistently dominates LoRA across the entire accuracy-parameter curve on GSM8K (Figure 6). For example, to reach a target test accuracy of 50%, REASONCACHE requires 46% fewer parameters than LoRA. REASONCACHE is more parameter-efficient in the low-budget regime and continues to improve as the budget increases, while LoRA saturates early. This trend aligns with our theoretical analysis in Section 4: by injecting learned key-value prefixes at every layer, REASONCACHE can steer attention along directions that low-rank weight updates cannot realize even at a *higher* parameter budget.

A note on parameter counting: When REASONCACHE uses the MLP reparameterization (see Appendix C.1), the MLP serves only to stabilize training and is discarded at deployment, leaving only the generated prefix vectors; we therefore report only the deployed parameters (the prefix vectors) in Figure 6.

4 Theoretical Perspectives

The preceding experiments demonstrate that REASONCACHE can match or exceed its in-weight counterpart, LoRA, on challenging reasoning benchmarks (Section 3.2). What explains this? In this section, we show that REASONCACHE (and PT broadly) and LoRA are governed by fundamentally different expressivity bottlenecks: each method can, in certain regimes, access subspaces in value space that are unreachable by the other. Since this analysis is not specific to reasoning, we present it in the language of *Prefix Tuning* (PT) for clarity, using PT notation throughout. We characterize exactly when each method is more expressive and provide geometric intuition for the comparison. All proofs are deferred to Appendix A.

4.1 Setting and Notation

We adopt the notation of Section 2. Consider a single attention layer with frozen value projection $W_V \in \mathbb{R}^{d \times d}$. Given context embeddings $X \in \mathbb{R}^{n \times d}$, the base model produces value vectors v_1, \dots, v_n where $v_i = x_i W_V$. We define

$$S_X := \text{span}\{v_1, \dots, v_n\} \subseteq \mathbb{R}^d,$$

which captures all directions already expressible by the base model on this context. Let Π_X denote the orthogonal projector onto the complement S_X^\perp , and define:

$$t_X := \text{rank}(X), \quad \nu_X := \dim S_X^\perp = d - \dim S_X.$$

Here, t_X is the rank of the input sequence and ν_X as the *novelty capacity*, defined as the dimension of the subspace orthogonal to what the base model can already express. Intuitively, ν_X quantifies the remaining “headroom” in representation space for adding directions that are not already induced by the input context.

A LoRA update modifies the value projection to $W'_V = W_V + \Delta_V$ with $\text{rank}(\Delta_V) \leq r$, producing updated values $v'_i = x_i(W_V + \Delta_V)$. Prefix tuning instead contributes learned prefix values $P_V \in \mathbb{R}^{m \times d}$, which are concatenated with the token-derived values. Since the attention output is a convex combination of value

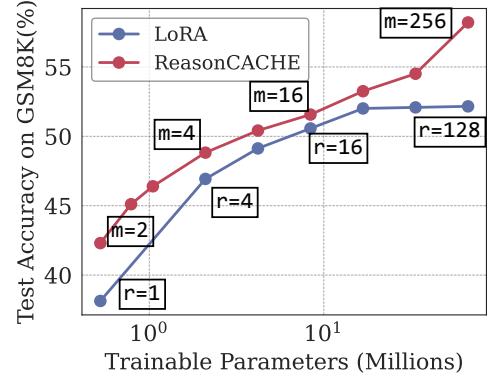


Figure 6 REASONCACHE achieves consistently higher test accuracy at matched parameter budgets and continues to improve where LoRA saturates.

vectors, it cannot produce directions outside the span of those values. The key comparison is therefore between the novelty subspaces, i.e., the new directions each method can add beyond S_X :

$$\Pi_X \text{span}\{v'_1, \dots, v'_n\} \quad (\text{LoRA}) \quad \text{vs.} \quad \Pi_X \text{span}\{(P_V)_1, \dots, (P_V)_m\} \quad (\text{PT}).$$

We refer to these subspaces as *novelty subspaces* since they capture directions that are not realizable by the base model under the given context.

4.2 Characterizing Achievable Novelty Subspaces

We first characterize exactly which novelty subspaces each method can realize.

Proposition 1. (LoRA Novelty Subspaces). At a fixed context X , a subspace $U \subseteq S_X^\perp$ is realizable as $\Pi_X \text{span}\{v'_1, \dots, v'_n\}$ for some value update Δ_V with $\text{rank}(\Delta_V) \leq r$ if and only if

$$\dim U \leq \min\{t_X, r\}.$$

Proposition 2. (Prefix Tuning Novelty Subspaces). At a fixed context X , a subspace $U \subseteq S_X^\perp$ is realizable as $\Pi_X \text{span}\{v'_1, \dots, v'_n\}$ for some value update Δ_V with $\text{rank}(\Delta_V) \leq r$ if and only if

$$\dim U \leq \min\{t_X, r\}.$$

Proposition 1 and **Proposition 2** state that both methods can realize *arbitrary* sets of new directions outside the base model's context-induced span; the difference is the bottleneck on how many such directions they can realize. For LoRA, the number of independent new directions is jointly limited by (i) how many independent directions the input context provides and (ii) the adapter's rank. For prefix tuning, the directions are likewise unconstrained, but their number is limited only by the number of prefix vectors.

Remark 3. **Proposition 1** reveals that the context X creates a bottleneck at $\min\{t_X, r\}$ for LoRA, resulting in what we call *carrier bottleneck*. Even with arbitrarily large adapter rank r , LoRA cannot produce novelty subspaces of dimension exceeding the context rank t_X . In contrast, **Proposition 2** shows that PT is *prefix-limited* only by m , the number of prefix vectors, which contribute directly without passing through the context.

4.3 Characterizing the Expressivity of LoRA and Prefix Tuning

We now compare the entire families of achievable novelty subspaces. Define:

$$\begin{aligned} \mathcal{L}(r) &:= \{\Pi_X \text{span}\{v'_1, \dots, v'_n\} : \text{rank}(\Delta_V) \leq r\}, \\ \mathcal{P}(m) &:= \{\Pi_X \text{span}\{(P_V)_1, \dots, (P_V)_m\} : P_V \in \mathbb{R}^{m \times d}\}. \end{aligned}$$

Theorem 1. (Expressivity Comparison). At a fixed context X , define $D_{\text{LoRA}} := \min\{t_X, r, \nu_X\}$ and $D_{\text{PT}} := \min\{m, \nu_X\}$. Then:

- (i) $\mathcal{L}(r) \subseteq \mathcal{P}(m)$ if and only if $D_{\text{LoRA}} \leq D_{\text{PT}}$.
- (ii) $\mathcal{P}(m) \subseteq \mathcal{L}(r)$ if and only if $D_{\text{PT}} \leq D_{\text{LoRA}}$.

Consequently, strict inclusion $\mathcal{L}(r) \subset \mathcal{P}(m)$ holds if and only if $D_{\text{LoRA}} < D_{\text{PT}}$ and equality $\mathcal{L}(r) = \mathcal{P}(m)$ holds if and only if $D_{\text{LoRA}} = D_{\text{PT}}$ and vice versa.

Interpretation. **Theorem 1** reveals that the expressive power of each method is governed merely by the prefix length m and the LoRA bottleneck $\min\{t_X, r\}$, subject to the novelty capacity ν_X .

PT is strictly more expressive when $m > \min\{t_X, r\}$. This occurs in two scenarios: (i) when a low-rank context carrier limits the LoRA update, regardless of the adapter rank r ; or (ii) when a low-rank adapter is chosen for parameter efficiency, and its rank is smaller than the prefix length. In these regimes, PT can realize novelty subspaces that no LoRA configuration can match.

Note that, in contemporary PEFT practice, prefix lengths are often chosen to be moderately large, ranging from tens to thousands of tokens (e.g., $m \in [16, 4096]$). LoRA however typically employs small adapter

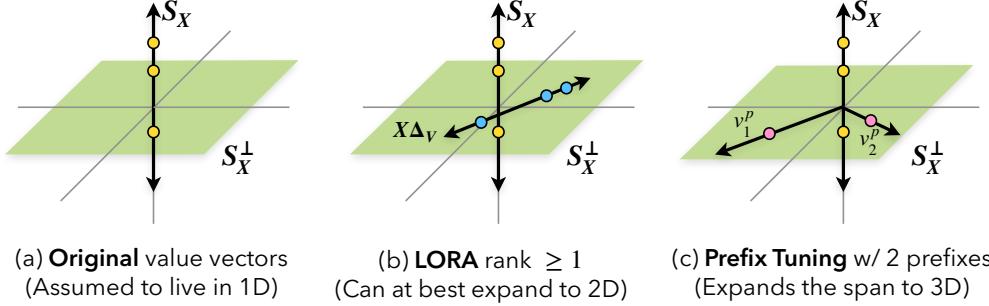


Figure 7 Prefix Tuning can span directions unreachable by LoRA. Assume the context X has rank one ($t_X = 1$). The original value vectors span a 1D subspace S_X (vertical axis); any new information must lie in the orthogonal novelty space S_X^\perp (horizontal plane). (a) The base model is confined to S_X . (b) A LoRA update introduces novelty via $X\Delta v$, but is carrier-limited by the context’s rank, adding at most one new dimension regardless of the adapter rank r . (c) Prefix tuning directly injects m learnable vectors. With $m = 2$, it can span the full 2D novelty space S_X^\perp , which LoRA cannot reach. This separation is not specific to the example: whenever $t_X < m$, PT accesses strictly more directions than LoRA.

ranks most commonly $r \in [4, 128]$). As a result, for sufficiently diverse and long inputs, the condition $m > \min\{t_X, r\}$ frequently holds, placing many practical deployments in the regime where prefix tuning enjoys a strict expressivity advantage.

LoRA is strictly more expressive when $\min\{t_X, r\} > m$, that is, when a high-rank context combined with a sufficiently large adapter rank provides a wider bottleneck than the prefix length.

Both methods are equivalent when the novelty capacity ν_X is the binding constraint. If $\nu_X < m$ and $\nu_X < \min\{t_X, r\}$, both methods saturate at $D_{\text{LoRA}} = D_{\text{PT}} = \nu_X$, and the choice reduces to other considerations such as computational efficiency.

Connection to Chain-of-Thought Reasoning. A useful implication of our analysis is that the parameter t_X depends only on the *context* available to the attention layer, regardless of whether that context comes from the original prompt or from tokens generated by the model itself. Producing a chain-of-thought expands this context with additional tokens, and can therefore increase t_X , enlarging the value span S_X and the set of directions available for subsequent computation. Through [Proposition 1–Theorem 1](#), this offers a plausible geometric view of why chain-of-thought often improves accuracy ([Wei et al., 2021](#)); by increasing t_X , it relaxes the input bottleneck that limits how many new directions LoRA can induce on a given input and expands the attainable output space.

Corollary 1 (PT Strictly More Expressive than QK-LoRA). *For LoRA applied only to query and key matrices, the value update rank is $r = 0$, giving $D_{\text{LoRA}} = 0$. If the novelty space is not saturated ($\nu_X > 0$) and at least one prefix token is used ($m \geq 1$), then $D_{\text{PT}} \geq 1 > D_{\text{LoRA}}$, implying $\mathcal{L}(0) = \{\{0\}\} \subset \mathcal{P}(m)$.*

This corollary highlights a fundamental difference in mechanism: LoRA on Q/K matrices can only *re-weight* existing value vectors within their original span S_X , whereas prefix tuning can *introduce entirely new vectors* into the attention mechanism’s value dictionary, enabling outputs in directions previously unavailable.

4.4 Empirical Evidence for Expressivity via Representation Geometry

We empirically validate our analysis above by probing the representation geometry (details in [Appendix C.2](#)). First, across attention heads, the pretrained model’s value vectors occupy a low-dimensional subspace, revealing substantial unused headroom in value space ([Figure 8](#)). REASONCACHE exploits this headroom since learned prefix values place most of their energy (typically 70–90%) outside the dominant value subspace, injecting directions the base computation scarcely uses rather than merely reweighting it ([Figure 9](#)). These directions further persist downstream: REASONCACHE’d models exhibit higher effective rank in last-layer representations than both supervised fine-tuning and LoRA, corroborating our layer-level expressivity mechanism despite the intervening nonlinearities across layers.

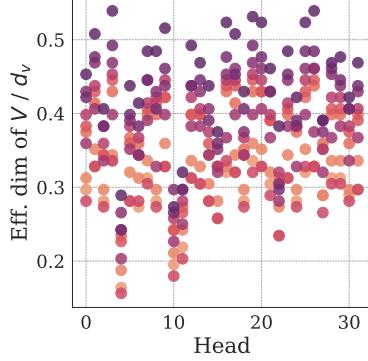


Figure 8 Effective dimension of value vectors across heads. Most heads use less than half of their available dimension, showing that the base model operates in a compact subspace.

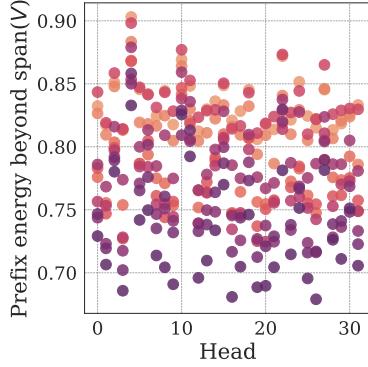


Figure 9 Fraction of prefix energy outside the value subspace of the base model. Prefixes place 80% of their mass beyond this span, exploiting directions unused by the base model.



Figure 10 Effective rank of last-layer token representations, defined as the rank at which the cumulative singular-value mass reaches 90%. REASONCACHE attains the highest effective rank among all methods.

4.5 An Illustrative Example

We now provide an example with a geometric interpretation comparing PT and LoRA, as shown in Figure 7. Consider a single attention layer, without nonlinearities or residual connections. Suppose the context X has rank one, so all value vectors lie on a line i.e. S_X is 1-dimensional.

Now consider a binary classification task: each token i belongs to one of two classes, and the target output y_i^* should be e_1 for class 1 and e_2 for class 2, where e_1, e_2 are orthogonal unit vectors in S_X^\perp . Since the targets span two dimensions in S_X^\perp , any method that can only produce one-dimensional outputs will fail.

Why QK-LoRA fails. Adapting only the query and key matrices changes the attention weights but not the values. The output remains a convex combination of the original values v_1, \dots, v_n , all of which lie in S_X . No matter how the attention is reweighted, the outputs cannot escape this 1D subspace. They thus have zero component in S_X^\perp , let alone the required two dimensions.

Why QKV-LoRA fails. Adding an update Δ_V to the value projection produces new values $v'_i = x_i(W_V + \Delta_V)$. Even if the LoRA rank r is arbitrarily large, the product $X\Delta_V$ has rank at most $\text{rank}(X) = 1$. This is because context X limits what the weight update can express (Proposition 1). The updated values span only a 1D subspace, which may have a component in S_X^\perp but cannot fully span the two dimensions.

Why ReasonCache succeeds. Prefix tuning with $m = 2$ can place two prefix values $(P_V)_1 = e_1$ and $(P_V)_2 = e_2$ pointing in the two target directions. The prefix keys $(P_K)_1$ and $(P_K)_2$ can then be learned to route each token to the appropriate prefix value based on its class. This achieves the desired two-class separation without any constraint from the context’s rank.

Theorem 2. (Loss Hierarchy). In the setting above, QK-LoRA and QKV-LoRA both incur strictly positive loss, while prefix tuning with $m = 2$ drives the loss to zero.

The formal proof, which constructs explicit solutions and lower bounds, is given in Appendix A.

5 Related Work

An extended discussion of related work appears in Appendix D; here we summarize the most relevant threads.

In-context learning (ICL) enables LLMs to adapt from demonstrations without gradient updates (Brown et al., 2020), but struggles to scale: performance saturates or degrades as context length increases (Agarwal et al., 2024; Liu et al., 2024; Hong et al., 2025), and ICL has been argued to be a shallow learner (de Wynter,

2025). This is particularly limiting for reasoning, where a single training example can exceed 10k tokens, making it infeasible to fit many demonstrations in context. The dominant approach for complex reasoning instead relies on weight updates via fine-tuning or reinforcement learning (Guo et al., 2025; OpenAI, 2024; Yu et al., 2023), but in-weight learning has its own limitations, such as the reversal curse (Berglund et al., 2023).

Prefix tuning (Li and Liang, 2021) offers an alternative by learning continuous vectors that steer the model without weight updates. While originally studied for classification and generation, we explore whether prefix tuning can acquire complex reasoning skills. Related work on context compression (Mu et al., 2023; Chevalier et al., 2023; Ge et al., 2023) and KV-cache distillation (Caccia et al., 2025) focuses on compressing contextual information rather than learning persistent skills. Cartridges (Eyüboğlu et al., 2025) compresses massive contexts into learnable KV-caches for knowledge retrieval; in contrast, we target skill acquisition (e.g., reasoning).

We view the KV-cache as a learnable *memory interface* spanning a spectrum from raw tokens (recovering ICL) to gradient-optimized vectors (approaching in-weight methods). This framing connects to work on memory in neural networks (Zhang et al., 2024; Lin et al., 2025) and transformers as implicit optimizers (Von Oswald et al., 2023; Mittal et al., 2025). Importantly, prefix tuning as studied here is *not* a test-time learner: the prefix is trained offline and frozen at deployment. Developing methods where the KV-cache remains plastic at test time, enabling true continual learning, remains an open problem.

6 Discussion

In this work, we introduce REASONCACHE, a mechanism that instantiates prefix tuning (PT) and learns a compact KV-prefix to distill and scale in-context data beyond the context window for efficient reasoning. Without modifying any existing weights, post-training with REASONCACHE can match or surpass in-weight adaptation methods such as LoRA and SFT on reasoning benchmarks (GSM8K, MATH, AIME, GPQA). REASONCACHE turns out to be more data-efficient (59% less data), produces more succinct reasoning chains (34% shorter generations), and is substantially more parameter-efficient (46% smaller adapted state) than in-weight methods. We theoretically ground the benefits of REASONCACHE by demonstrating that it can be more expressive by spanning new directions beyond what LoRA can cover. In all, REASONCACHE offers an efficient alternative for post-training LLMs to reason.

Limitations and future work While we focus on reasoning in this work, future work could also explore REASONCACHE as a post-training method to acquire skills other than reasoning. Given the plug-able nature of prefix caches, REASONCACHE opens the door for the composition of multiple skills with benefits including 1) avoiding catastrophic forgetting given prefix vectors leave existing weights intact, 2) inherently interpretable architecture via each skill’s prefix attention scores, 3) adaptable prefix sizes for more or less complex skills. REASONCACHE also offers possibilities for designing new flexible forms of memory. REASONCACHE provides a *spectrum of memory between short-term transient in-context and long-term persistent in-weight memory*. Additionally, the depth at which REASONCACHE is applied can control the flow of abstraction from finer-grained information closer to the raw inputs at earlier layers to more abstract information distilled through deeper layers. As REASONCACHE preserves compatibility with both ICL and in-weight methods, future work could also explore how REASONCACHE can be best combined with existing methods to further enhance steering and skill acquisition in LLMs. Finally, current inference engines and post-training frameworks do not natively support trainable KV-caches. For broad practical adoption, future work could also consider integrating trainable KV-caches into existing inference engines and post-training libraries using mechanisms similar to KV-caching.

7 Acknowledgements

This work was supported by a Packard Fellowship to P.I., by the MIT-IBM Watson AI Lab, and by ONR MURI grant N00014-22-1-2740. This work was also supported by the NSF AI Institute TILOS (NSF CCF2112665) and the Alexander von Humboldt Foundation. S.G. acknowledges the support of the MathWorks Engineering Fellowship. We thank Reyhane Askari, Divyat Mahajan, Sachin Goyal, Andrei Nicolicioiu and Sarthak Mittal for insightful discussions.

References

- Rishabh Agarwal, Avi Singh, Lei Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, et al. Many-shot in-context learning. *Advances in Neural Information Processing Systems*, 37:76930–76966, 2024.
- Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*, 2024.
- Anas Awadalla, Mitchell Wortsman, Gabriel Ilharco, Sewon Min, Ian Magnusson, Hannaneh Hajishirzi, and Ludwig Schmidt. Exploring the landscape of distributional robustness for question answering models. *arXiv preprint arXiv:2210.12517*, 2022.
- Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. The reversal curse: Llms trained on "a is b" fail to learn "b is a". *arXiv preprint arXiv:2309.12288*, 2023.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- Lucas Caccia, Alan Ansell, Edoardo Ponti, Ivan Vulic, and Alessandro Sordoni. Training plug-n-play knowledge modules with deep context distillation. *arXiv preprint arXiv:2503.08727*, 2025.
- Pan Chen, Shaohong Chen, Mark Wang, Shi Xuan Leong, Priscilla Fung, Varinia Bernales, and Alan Aspuru-Guzik. Schema for in-context learning. *arXiv preprint arXiv:2510.13905*, 2025.
- Tong Chen, Hao Fang, Patrick Xia, Xiaodong Liu, Benjamin Van Durme, Luke Zettlemoyer, Jianfeng Gao, and Hao Cheng. Generative adapter: Contextualizing language models in parameters with a single forward pass. *arXiv preprint arXiv:2411.05877*, 2024.
- Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*, 2023.
- Yu-Neng Chuang, Tianwei Xing, Chia-Yuan Chang, Zirui Liu, Xun Chen, and Xia Hu. Learning to compress prompt in natural language formats. *arXiv preprint arXiv:2402.18700*, 2024.
- 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.
- Adrian de Wynter. Is in-context learning learning? *arXiv preprint arXiv:2509.10414*, 2025.
- Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, et al. Cartridges: Lightweight and general-purpose long context representations via self-study. *arXiv preprint arXiv:2506.06266*, 2025.
- Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL <https://zenodo.org/records/12608602>.
- Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*, 2023.
- Tim Genewein, Li Kevin Wenliang, Jordi Grau-Moya, Anian Ruoss, Laurent Orseau, and Marcus Hutter. Understanding prompt tuning and in-context learning via meta-learning. *arXiv preprint arXiv:2505.17010*, 2025.
- Shangyi Geng, Wenting Zhao, and Alexander M Rush. Great memory, shallow reasoning: Limits of k nn-lms. *arXiv preprint arXiv:2408.11815*, 2024.
- Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, Zayne Sprague, et al. Openthoughts: Data recipes for reasoning models. *arXiv preprint arXiv:2506.04178*, 2025.
- Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

- Sharut Gupta, Stefanie Jegelka, David Lopez-Paz, and Kartik Ahuja. Context is environment. *arXiv preprint arXiv:2309.09888*, 2023.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- Kelly Hong, Troynikov Anton, and Jeff Huber. Context rot: How increasing input tokens impacts llm performance. Technical report, Chroma, 2025. URL <https://research.trychroma.com/context-rot>.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023.
- James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.
- Yuri Kuratov, Mikhail Arkhipov, Aydar Bulatov, and Mikhail Burtsev. Cramming 1568 tokens into a single vector and back again: Exploring the limits of embedding space capacity. *arXiv preprint arXiv:2502.13063*, 2025.
- Andrew K Lampinen, Arslan Chaudhry, Stephanie CY Chan, Cody Wild, Diane Wan, Alex Ku, Jörg Bornschein, Razvan Pascanu, Murray Shanahan, and James L McClelland. On the generalization of language models from in-context learning and finetuning: a controlled study. *arXiv preprint arXiv:2505.00661*, 2025.
- Minh Le, Chau Nguyen, Huy Nguyen, Quyen Tran, Trung Le, and Nhat Ho. Revisiting prefix-tuning: Statistical benefits of reparameterization among prompts. *arXiv preprint arXiv:2410.02200*, 2024.
- Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale: Parameter-efficient adaptation for pretrained language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.
- Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Association for Computational Linguistics (ACL)*, 2021.
- Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. Continual learning via sparse memory finetuning. *arXiv preprint arXiv:2510.15103*, 2025.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.
- Jack Lu, Ryan Teehan, Zhenbang Yang, and Mengye Ren. Context tuning for in-context optimization. *arXiv preprint arXiv:2507.04221*, 2025.
- MAA. Aime 2024 problems. https://artofproblemsolving.com/wiki/index.php/2024_AIME_I_Problems, 2024. Accessed: 2025-05-11.
- MAA. Aime 2025 problems. https://artofproblemsolving.com/wiki/index.php/2025_AIME_I_Problems, 2025. Accessed: 2025-05-11.
- Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.
- Sarthak Mittal, Divyat Mahajan, Guillaume Lajoie, and Mohammad Pezeshki. Iterative amortized inference: Unifying in-context learning and learned optimizers. *arXiv preprint arXiv:2510.11471*, 2025.
- Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. *arXiv preprint arXiv:2305.16938*, 2023.
- Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

OpenAI. Learning to reason with llms, 08 2024. URL <https://openai.com/index/learning-to-reason-with-llms/>.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, et al. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*, 2024.

Aleksandar Petrov, Philip HS Torr, and Adel Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. *arXiv preprint arXiv:2310.19698*, 2023.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*, 2022.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. Parametric retrieval augmented generation. *arXiv preprint arXiv:2501.15915*, 2025.

Eric Todd, Millicent L Li, Arnab Sen Sharma, Aaron Mueller, Byron C Wallace, and David Bau. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.

Johannes Von Oswald, Eyyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pages 35151–35174. PMLR, 2023.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. TRL: Transformer Reinforcement Learning. URL <https://github.com/huggingface/trl>.

Haonan Wang, Brian K Chen, Li Siquan, Liang Xinhe, Tianyang Hu, Hwee Kuan Lee, and Kenji Kawaguchi. Prefix-tuning+: Modernizing prefix-tuning by decoupling the prefix from attention. In *Second Workshop on Test-Time Adaptation: Putting Updates to the Test! at ICML 2025*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

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*, 2024a.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel Candes, and Tatsunori Hashimoto. Synthetic continued pretraining. *arXiv preprint arXiv:2409.07431*, 2024b.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning. *arXiv preprint arXiv:2502.03387*, 2025.

Qingyu Yin, Xuzheng He, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen, and Qiang Zhang. Deeper insights without updates: The power of in-context learning over fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4138–4151, 2024.

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

Weizhe Yuan, Jane Yu, Song Jiang, Karthik Padthe, Yang Li, Ilia Kulikov, Kyunghyun Cho, Dong Wang, Yuandong Tian, Jason E Weston, et al. Naturalreasoning: Reasoning in the wild with 2.8 m challenging questions. *arXiv preprint arXiv:2502.13124*, 2025.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Jianyu Zhang and Léon Bottou. Memory mosaics at scale. *arXiv preprint arXiv:2507.03285*, 2025.

Jianyu Zhang, Niklas Nolte, Ranajoy Sadhukhan, Beidi Chen, and Léon Bottou. Memory mosaics. *arXiv preprint arXiv:2405.06394*, 2024.

Xiaoqing Zhang, Ang Lv, Yuhan Liu, Flood Sung, Wei Liu, Jian Luan, Shuo Shang, Xiuying Chen, and Rui Yan. More is not always better? enhancing many-shot in-context learning with differentiated and reweighting objectives. *arXiv preprint arXiv:2501.04070*, 2025.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Appendix

A Supplementary Proofs	18
A.1 Proof of Proposition 1 (LoRA Novelty Subspaces)	18
A.2 Proof of Proposition 2 (Prefix Tuning Novelty Subspaces)	18
A.3 Proof of Theorem 1 (Comparison of Expressivity)	18
A.4 Proof of Theorem 2 (Loss Hierarchy)	19
A.4.1 Part (i): QK-LoRA Loss Floor	19
A.4.2 Part (ii): QKV-LoRA Loss Floor	20
A.4.3 Part (iii): Prefix Tuning Drive the Loss to Zero	20
B Supplementary Experimental Details and Assets Disclosure	20
B.1 Assets	20
B.2 Hardware and Setup	20
B.3 Experimental Setup and Training Protocol	20
B.3.1 Training Datasets	20
B.3.2 Evaluation Benchmarks	21
B.3.3 Training and Evaluation Protocol	21
C Additional Experiments	21
C.1 Ablating Design Choices for REASONCACHE	21
C.1.1 Effect of Prefix Parametrization	22
C.1.2 Effect of Prefix Initialization	22
C.2 Probing The Representation Geometry for Expressivity	22
C.2.1 Base Models Operate In a Low-Dimensional Value Subspace.	22
C.2.2 Prefix Values Inject Energy Outside The Base Subspace	23
C.2.3 Injected Novelty Propagates To Downstream Representations	23
C.3 In-Weight Adaptation Exhibits Catastrophic Forgetting Unlike REASONCACHE	24
D Extended Related Work	25

A Supplementary Proofs

This appendix provides complete proofs for all theoretical results in [Section 4](#).

A.1 Proof of [Proposition 1 \(LoRA Novelty Subspaces\)](#)

Proposition 1. (LoRA Novelty Subspaces). At a fixed context X , a subspace $U \subseteq S_X^\perp$ is realizable as $\Pi_X \text{span}\{v'_1, \dots, v'_n\}$ for some value update Δ_V with $\text{rank}(\Delta_V) \leq r$ if and only if

$$\dim U \leq \min\{t_X, r\}.$$

Proof. (\Rightarrow) Assume $U = \Pi_X \text{span}\{v'_1, \dots, v'_n\}$ for some Δ_V with $\text{rank}(\Delta_V) \leq r$, where $v'_i = x_i(W_V + \Delta_V)$. The dimension of a projected subspace cannot exceed the dimension of the original subspace:

$$\dim U = \dim(\Pi_X \text{row}(X\Delta_V)) \leq \dim(\text{row}(X\Delta_V)) = \text{rank}(X\Delta_V).$$

By properties of matrix rank, $\text{rank}(X\Delta_V) \leq \min\{\text{rank}(X), \text{rank}(\Delta_V)\}$. Given $\text{rank}(X) = t_X$ and $\text{rank}(\Delta_V) \leq r$, we have $\dim U \leq \min\{t_X, r\}$.

(\Leftarrow) Let $U \subseteq S_X^\perp$ be a subspace with dimension $s := \dim U \leq \min\{t_X, r\}$. We construct a matrix Δ_V with $\text{rank}(\Delta_V) \leq r$ such that $\Pi_X \text{span}\{v'_1, \dots, v'_n\} = U$.

Let $\{b_1, \dots, b_s\}$ be an orthonormal basis for U , and form the matrix $B \in \mathbb{R}^{d \times s}$ whose columns are these basis vectors, so $\text{col}(B) = U$. Since $s \leq t_X = \text{rank}(X)$, there exists a matrix $C \in \mathbb{R}^{d \times s}$ such that $XC \in \mathbb{R}^{n \times s}$ has rank s . To see this constructively, let the SVD of X be $U_X \Sigma_X V_X^\top$; choosing C to be the first s columns of V_X guarantees that XC has rank s .

Define $\Delta_V := CB^\top$. The rank satisfies $\text{rank}(\Delta_V) \leq \min\{\text{rank}(C), \text{rank}(B^\top)\} = s \leq r$. The resulting novelty matrix is $X\Delta_V = (XC)B^\top$. Since XC has full column rank s and B^\top has full row rank s , we have $\text{rank}(X\Delta_V) = s$ and $\text{span}\{(X\Delta_V)_1, \dots, (X\Delta_V)_n\} = U$.

Finally, since $U \subseteq S_X^\perp$, the projector Π_X acts as the identity on U :

$$\Pi_X \text{span}\{v'_1, \dots, v'_n\} = \Pi_X(U) = U.$$

□

A.2 Proof of [Proposition 2 \(Prefix Tuning Novelty Subspaces\)](#)

Proposition 2. (Prefix Tuning Novelty Subspaces). At a fixed context X , a subspace $U \subseteq S_X^\perp$ is realizable as $\Pi_X \text{span}\{v'_1, \dots, v'_n\}$ for some value update Δ_V with $\text{rank}(\Delta_V) \leq r$ if and only if

$$\dim U \leq \min\{t_X, r\}.$$

Proof. (\Rightarrow) Assume $U = \Pi_X \text{span}\{(P_V)_1, \dots, (P_V)_m\}$. Then $\dim U \leq \dim(\text{span}\{(P_V)_1, \dots, (P_V)_m\}) = \text{rank}(P_V)$. Since P_V has m rows, its rank is at most m , so $\dim U \leq m$.

(\Leftarrow) Let $U \subseteq S_X^\perp$ be a subspace with dimension $s := \dim U \leq m$. Let $\{u_1, \dots, u_s\}$ be a basis for U . Construct $P_V \in \mathbb{R}^{m \times d}$ by setting its first s rows to these basis vectors and the remaining $m - s$ rows to zero. Then $\text{span}\{(P_V)_1, \dots, (P_V)_m\} = U$. Since $U \subseteq S_X^\perp$, the projector Π_X acts as the identity on U :

$$\Pi_X \text{span}\{(P_V)_1, \dots, (P_V)_m\} = \Pi_X(U) = U.$$

□

A.3 Proof of [Theorem 1 \(Comparison of Expressivity\)](#)

Theorem 1. (Expressivity Comparison). At a fixed context X , define $D_{\text{LoRA}} := \min\{t_X, r, \nu_X\}$ and $D_{\text{PT}} := \min\{m, \nu_X\}$. Then:

- (i) $\mathcal{L}(r) \subseteq \mathcal{P}(m)$ if and only if $D_{\text{LoRA}} \leq D_{\text{PT}}$.
- (ii) $\mathcal{P}(m) \subseteq \mathcal{L}(r)$ if and only if $D_{\text{PT}} \leq D_{\text{LoRA}}$.

Consequently, strict inclusion $\mathcal{L}(r) \subset \mathcal{P}(m)$ holds if and only if $D_{\text{LoRA}} < D_{\text{PT}}$ and equality $\mathcal{L}(r) = \mathcal{P}(m)$ holds if and only if $D_{\text{LoRA}} = D_{\text{PT}}$ and vice versa.

Proof. By [Proposition 1](#) and [Proposition 2](#), the families $\mathcal{L}(r)$ and $\mathcal{P}(m)$ are precisely the sets of all subspaces of S_X^\perp whose dimensions are bounded by $\min\{t_X, r\}$ and m , respectively:

$$\mathcal{L}(r) = \{U \subseteq S_X^\perp : \dim U \leq \min\{t_X, r\}\}, \quad \mathcal{P}(m) = \{U \subseteq S_X^\perp : \dim U \leq m\}.$$

Since any realizable subspace U must reside in the ambient novelty space S_X^\perp , its dimension is inherently capped by $\nu_X = \dim S_X^\perp$. Thus, the maximum achievable dimensions are $D_{\text{LoRA}} = \min\{t_X, r, \nu_X\}$ and $D_{\text{PT}} = \min\{m, \nu_X\}$.

The set of all subspaces of a vector space up to a certain dimension forms a nested hierarchy: a family defined by cap D_1 is a subset of another with cap D_2 if and only if $D_1 \leq D_2$. This immediately implies:

$$\begin{aligned} \mathcal{L}(r) \subseteq \mathcal{P}(m) &\iff D_{\text{LoRA}} \leq D_{\text{PT}}, \\ \mathcal{P}(m) \subseteq \mathcal{L}(r) &\iff D_{\text{PT}} \leq D_{\text{LoRA}}. \end{aligned}$$

Strict inclusion $\mathcal{L}(r) \subset \mathcal{P}(m)$ holds if and only if $\mathcal{L}(r) \subseteq \mathcal{P}(m)$ and $\mathcal{L}(r) \neq \mathcal{P}(m)$, which is equivalent to $D_{\text{LoRA}} < D_{\text{PT}}$. In this case, there exists a subspace $U \subseteq S_X^\perp$ of dimension D_{PT} belonging to $\mathcal{P}(m)$ but not to $\mathcal{L}(r)$. \square

A.4 Proof of [Theorem 2](#) (Loss Hierarchy)

Theorem 2. (Loss Hierarchy). *In the setting above, QK-LoRA and QKV-LoRA both incur strictly positive loss, while prefix tuning with $m = 2$ drives the loss to zero.*

We formalize the setting of the illustrative example in [Section 4.5](#). Consider a single attention layer with frozen projections $W_Q, W_K, W_V \in \mathbb{R}^{d \times d}$. Suppose the context $X \in \mathbb{R}^{n \times d}$ has rank one, so $X = au^\top$ for some $a \in \mathbb{R}^n$ and unit vector $u \in \mathbb{R}^d$. Each token i belongs to one of two classes, with label $c_i \in \{1, 2\}$ determined by $c_i = 1$ if $a_i > 0$ and $c_i = 2$ if $a_i < 0$. Let $n_j := |\{i : c_i = j\}|$ denote the number of tokens in class j .

The target output for token i is $y_i^* := e_{c_i}$, where $e_1, e_2 \in \mathbb{R}^d$ are orthogonal unit vectors in S_X^\perp . The target matrix $Y^* \in \mathbb{R}^{n \times d}$ thus has rank 2. The loss is $\mathcal{L}(Y) := \frac{1}{2} \|Y - Y^*\|_F^2$.

We prove each part of the theorem separately.

A.4.1 Part (i): QK-LoRA Loss Floor

Lemma 1. *Under any variation $(\delta W_Q, \delta W_K)$, the output variation satisfies $\delta Y = (\delta A)V$, so each row $\delta y_i \in S_X$.*

Proof. The value matrix V does not depend on W_Q or W_K . Differentiating $Y = AV$ gives $\delta Y = (\delta A)V$; each row is a linear combination of rows of V , hence lies in $S_X = \text{span}\{v_1, \dots, v_n\}$. \square

Proof of Part (i). Let $E := Y - Y^*$ and decompose $E_i = E_{i,\parallel} + E_{i,\perp}$ with $E_{i,\parallel} \in S_X$ and $E_{i,\perp} \in S_X^\perp$. Let $\theta := (W_Q, W_K)$ and let $J := \frac{\partial \text{vec}(Y)}{\partial \theta}$ be the Jacobian.

By [Lemma 1](#), each row of any infinitesimal output change δY lies in S_X , so $\langle E_\perp, \delta Y \rangle_F = 0$ for all variations. Since $\text{vec}(\delta Y) = J\delta\theta$, we have $\text{vec}(E_\perp)^\top J\delta\theta = 0$ for all $\delta\theta$, implying $J^\top \text{vec}(E_\perp) = 0$.

The gradient of the loss is:

$$\nabla_\theta \mathcal{L} = J^\top \text{vec}(E) = J^\top \text{vec}(E_\parallel) + \underbrace{J^\top \text{vec}(E_\perp)}_{=0} = J^\top \text{vec}(E_\parallel).$$

Thus, the gradient has no component that can reduce the novelty error E_\perp . Since $y_i^* \in S_X^\perp$ and $y_i \in S_X$, we have $E_{i,\perp} = -y_i^*$, giving:

$$\mathcal{L}(Y) \geq \frac{1}{2} \|E_\perp\|_F^2 = \frac{1}{2} \sum_{i=1}^n \|y_i^*\|^2 > 0. \quad \square$$

A.4.2 Part (ii): QKV-LoRA Loss Floor

Proof. With $W'_V = W_V + \Delta_V$ trainable, we have $XW'_V = (au^\top)(W_V + \Delta_V) = a \cdot (u^\top W_V + u^\top \Delta_V)$. Since $u^\top W_V + u^\top \Delta_V$ is a $1 \times d$ row vector, all rows of XW'_V are scalar multiples of this vector. Thus $\text{rank}(XW'_V) \leq 1$, and consequently $\text{rank}(Y) \leq 1$ for all parameter settings.

The target Y^* has $\text{rank}(Y^*) = 2$, and its two nonzero singular values are $\sqrt{n_1}$ and $\sqrt{n_2}$ (equivalently, in an orthonormal basis containing e_1, e_2 , Y^* has exactly two nonzero columns, which are orthogonal indicator vectors of norms $\sqrt{n_1}$ and $\sqrt{n_2}$). By Eckart–Young–Mirsky, the best rank-1 approximation error in Frobenius norm is:

$$\min_{\text{rank}(Y) \leq 1} \frac{1}{2} \|Y - Y^*\|_F^2 = \frac{1}{2} \sigma_2(Y^*)^2 = \frac{1}{2} \min\{n_1, n_2\} > 0. \quad \square$$

A.4.3 Part (iii): Prefix Tuning Drive the Loss to Zero

Proof. The proof is constructive. First, we show the queries are linearly separable. The query vectors are $q_i = a_i(u^\top W_Q)$ where each q_i is a scalar multiple of the fixed direction $u^\top W_Q$. Define $\beta := (u^\top W_Q)^\top \in \mathbb{R}^d$. Assuming $\beta \neq 0$ (non-degeneracy), we have $\langle q_i, \beta \rangle = a_i \|\beta\|^2$, so $\text{sign}(\langle q_i, \beta \rangle) = \text{sign}(a_i)$. By the class definition, $c_i = 1$ iff $a_i > 0$ iff $\langle q_i, \beta \rangle > 0$, with margin $\gamma := \|\beta\|^2 \min_i |a_i| > 0$.

We construct a solution with prefix values $(P_V)_1 = e_1$, $(P_V)_2 = e_2$ and prefix keys $(P_K)_1 = \tau\beta$, $(P_K)_2 = -\tau\beta$ for large $\tau > 0$. For a class-1 token ($\langle q_i, \beta \rangle \geq \gamma$), the prefix logits satisfy $s'_{i1} = \frac{\tau}{\sqrt{d}} \langle q_i, \beta \rangle \rightarrow \infty$ and $s'_{i2} \rightarrow -\infty$ as $\tau \rightarrow \infty$. The attention weight on the first prefix converges to:

$$A'_{i,n+1} = \frac{e^{s'_{i1}}}{\sum_j e^{s'_{ij}}} \rightarrow 1 \quad \text{as } \tau \rightarrow \infty.$$

Symmetrically, class-2 tokens have $A'_{i,n+2} \rightarrow 1$. Thus, $y_i \rightarrow (P_V)_{c_i} = e_{c_i} = y_i^*$ for all i , and $\mathcal{L}(Y) \rightarrow 0$. \square

B Supplementary Experimental Details and Assets Disclosure

B.1 Assets

We do not introduce new data in the course of this work. Instead, we use publicly available, widely used image datasets for the purposes of benchmarking and comparison.

B.2 Hardware and Setup

Each model was trained with distributed training on 8 NVIDIA H200 GPUs (144 GB memory each) paired with Intel Xeon Platinum 8488C CPUs (48 cores). Training was implemented in PyTorch using the TRL library ([von Werra et al.](#)), and inference was run in parallel on 4 H200 GPUs. For evaluation, we used the Language Model Evaluation Harness ([lm-evaluation-harness](#)) library ([Gao et al., 2024](#)).

B.3 Experimental Setup and Training Protocol

B.3.1 Training Datasets

We rely on two large-scale training corpora that expose models to rich chains of mathematical reasoning.

MetaMathQA ([Yu et al., 2023](#)) is a dataset of roughly 400k math problems, each paired with detailed step-by-step solutions covering algebra, geometry, number theory, and calculus. We use it to provide broad coverage of high-school and undergraduate mathematics.

OpenThoughts-3 ([Guha et al., 2025](#)) contains approximately 3 million long-form reasoning traces across mathematics, physics, and logical puzzles. To make training feasible while retaining extended reasoning structure, we take a filtered subset of this dataset by discarding examples whose reasoning traces exceed 4096 tokens (as discussed in ([Guha et al., 2025](#))). This reduces overall size while preserving the majority of long, structured derivations.

B.3.2 Evaluation Benchmarks

Our evaluation spans two categories of reasoning tasks that capture distinct levels of difficulty: *high-school mathematics*, which tests structured multi-step problem solving, and *advanced competition or graduate-level problems*, which demand deeper abstraction and domain expertise.

High-school mathematics. GSM8K (Cobbe et al., 2021) consists of 8.5k grade-school word problems authored by human tutors. Each requires multi-step arithmetic reasoning and is paired with gold chain-of-thought solutions. MATH (Hendrycks et al., 2020) contains 12k competition-style problems spanning algebra, geometry, number theory, combinatorics, precalculus, and calculus. We follow prior work in evaluating on the standardized subset used in OpenAI’s benchmarks.

Advanced reasoning. GPQA-Diamond (Rein et al., 2024) contains 198 graduate-level questions in mathematics, physics, chemistry, and computer science. These items are sourced from qualifying exams and research-level material. Expert performance remains far from perfect (PhD holders achieve only 69.7%), making it a stringent test of advanced reasoning (OpenAI, 2024). When we write "GPQA" in the context of evaluation in this work, we always refer to the Diamond subset.

B.3.3 Training and Evaluation Protocol

Training Details. All models are trained with AdamW and a cosine learning-rate schedule with warmup ratio 0.05. Weight decay is set to zero throughout. On OpenThoughts-3, we train for 13 epochs with batch size 32 and a maximum sequence length of 8192. The learning rate is 1e-5 for supervised fine-tuning, and 1e-3 for prefix tuning, LoRA, and prompt tuning. On MetaMathQA, we train for 3 epochs with batch size 128 and a maximum sequence length of 2048, using a learning rate of 2e-5 for all methods.

For LoRA, we sweep over ranks $\{8, 16, 32, 64, 128\}$ and over different insertion points, including W_Q, W_K , full W_{QKV} , and W_{QKV} plus feedforward layers, and report the best-performing configuration. For prompt tuning and prefix tuning, we sweep over the number of learned tokens in $\{16, 32, 64, 128, 256, 512\}$ and report the best configuration. We additionally study two initialization techniques. First, *reparameterization*, where prefixes are generated from a smaller embedding projected through an MLP, improving training stability. Second, *initialization strategies*: prefixes are either initialized from a random token embedding in the base model’s vocabulary, or from a random example in the training dataset.

Evaluation Details. At inference, all experiments are evaluated using the lm-evaluation-harness framework (Gao et al., 2024), with temperature set to 0 (greedy decoding). Accuracy, equivalent to pass@1, is reported as the primary metric. For GSM8K and MATH, we rely on the standard evaluation configurations provided in the harness.

For models trained on the filtered OpenThoughts-3 dataset and evaluated on GPQA-Diamond, we adopt *budget forcing* at inference (Muennighoff et al., 2025). Prefix tuning is not yet supported in vLLM, and HuggingFace backends can become bottlenecked by very long generations. We believe this limitation has constrained large-scale testing and broader adoption of prefix tuning, confining most prior work to datasets with shorter or no reasoning traces. To address this, we follow the strategy proposed in Muennighoff et al. (2025): a decoding-time intervention that constrains the number of "thinking" tokens. Specifically, generation is terminated early by appending an end-of-thinking delimiter and "Final Answer:" once the budget is reached. In our experiments, we enforce a maximum reasoning budget of 4096 tokens for evaluations on GPQA.

C Additional Experiments

C.1 Ablating Design Choices for ReasonCache

We study two knobs: (1) how the prefix is parameterized; and (2) how it is initialized; and measure their effect under varying parameter budgets. In both ablations, we finetune using REASONCACHE on the complete MetaMathQA dataset and optimize to convergence, sweeping parameter budgets to quantify how each design choice impacts performance.

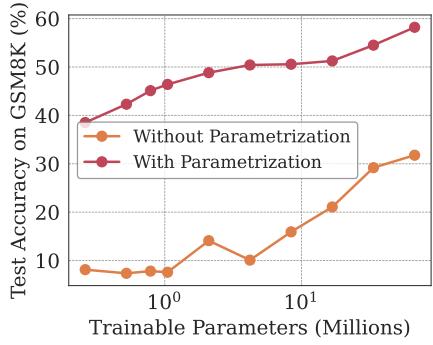


Figure 11 Test accuracy as a function of trainable parameters for REASONCACHE with direct optimization of prefix vectors and with MLP parametrization. Parametrization consistently improves accuracy, especially in the low-parameter regime.

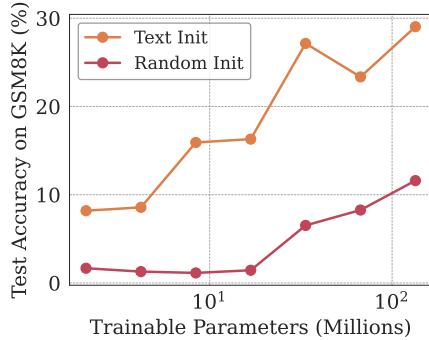


Figure 12 Test accuracy as a function of trainable parameters for REASONCACHE with random and text-based initialization. Initializing prefixes from text embeddings yields higher accuracy across all parameter budgets.

C.1.1 Effect of Prefix Parametrization

REASONCACHE learns a set of virtual tokens (key-value vectors) at each layer. One can optimize these prefixes directly as free parameters, or generate them with a small network (e.g., an MLP) trained end-to-end via backpropagation. As shown in Figure 11, MLP parametrization consistently improves test performance across all trainable-parameter budgets, with the largest gains in the low-budget regime. As we increase the parameter budget by increasing the prefix length, both variants improve, but parametrized prefixes retain a clear advantage throughout.

C.1.2 Effect of Prefix Initialization

Initialization also plays a central role in REASONCACHE. Prefixes can be learned from scratch with random initialization, or initialized from text embeddings of few-shot exemplars. As shown in Figure 12, text-based initialization consistently improves accuracy across all trainable-parameter budgets. As we increase the prefix length, optimization improves both variants, but the advantage of text initialization persists.

C.2 Probing The Representation Geometry for Expressivity

Our theoretical analysis shows that when REASONCACHE operates in regimes where in-weight updates are carrier-limited, it can realize its advantage by injecting value-space directions that lie outside the subspace used by the base model. We empirically validate this by probing representation geometry at three levels: (i) the dimensionality of value vectors used by the pretrained model, (ii) the directions of learned prefix values relative to this subspace, and (iii) the resulting dimensionality of downstream representations. All experiments in this section use LLaMA-2 7B as the base model, evaluated on GSM8K after adaptation on MetaMathQA.

C.2.1 Base Models Operate In a Low-Dimensional Value Subspace.

For each attention head in the final layer, we analyze the value matrix $V \in \mathbb{R}^{n \times d_v}$ induced by the pretrained model and compute its singular-value spectrum. Since σ_i^2 are the eigenvalues of $V^\top V$, they measure the variance explained along each direction. We define the *effective dimension* of V as the smallest K such that

$$\sum_{i=1}^K \sigma_i^2 \geq 0.9 \sum_{j=1}^{d_v} \sigma_j^2,$$

reported as K/d_v . Across heads and inputs, we find that the effective dimension of V , defined as the smallest K capturing 90% of the total variance, is consistently well below d_v (Figure 13). This indicates that the base

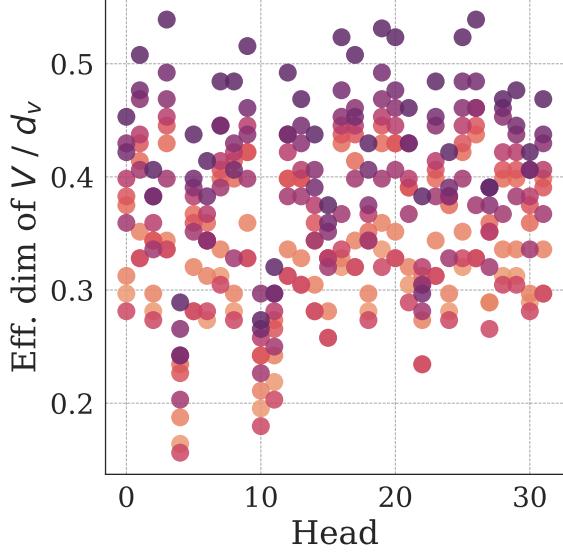


Figure 13 Effective dimension of value vectors across attention heads. Most heads use less than half of their available dimension, showing that the base model operates in a compact subspace.

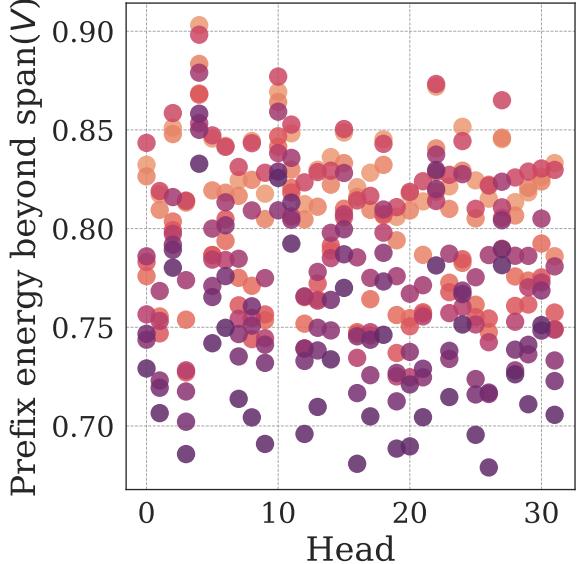


Figure 14 Fraction of prefix energy outside the dominant value subspace of the base model. Prefixes place 80% of their mass beyond this span, exploiting directions unused by the base model.

model concentrates computation in a compact value subspace, leaving substantial unused degrees of freedom for adaptation algorithms to exploit.

C.2.2 Prefix Values Inject Energy Outside The Base Subspace

We next examine whether REASONCACHE exploits this geometric headroom. Let $V_P \in \mathbb{R}^{m \times d_v}$ denote the learned prefix values. Using the K -dimensional subspace of V defined above, we project V_P onto $\text{span}(V)$ and measure the normalized residual energy

$$\frac{\|V_P - \Pi_{\text{span}(V)} V_P\|_F}{\|V_P\|_F}.$$

This quantity captures the fraction of prefix energy that lies outside the value subspace actively used by the base model. As shown in Figure 14, learned prefixes allocate the majority of their energy (typically 70–90%) outside $\text{span}(V)$, consistently across attention heads and input samples. Thus, REASONCACHE does not merely reweight existing value directions but injects novel directions that are largely unused by the pretrained computation.

C.2.3 Injected Novelty Propagates To Downstream Representations

We finally ask whether these newly injected directions persist beyond value space and reshapes the geometry of the model’s downstream representations. Concretely, we compute the singular-value spectrum of the last-layer token representation matrix F and normalize it to form a distribution over singular directions. Averaging this normalized spectrum across the evaluation set and taking its cumulative mass yields the curves in Figure 15. We summarize these curves via an *effective rank*, defined as the smallest k whose averaged cumulative mass reaches 90%.

As shown in Figure 16, REASONCACHE produces a markedly more gradual spectrum and a higher effective rank than both supervised fine-tuning and LoRA. This indicates that the directions injected at the attention level do not vanish downstream; instead, they propagate through the network and are expressed in the final representations as broader support across singular directions.

Together, these results corroborate our theoretical analysis. The pretrained model indeed operates within a

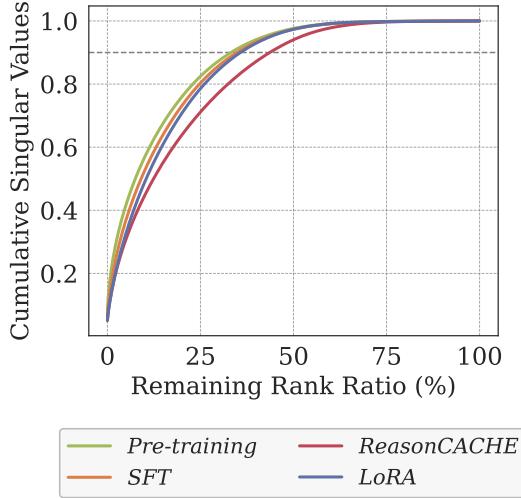


Figure 15 Cumulative singular-value spectra of last-layer token representations for GSM8K. REASONCACHE yields a more gradual spectrum than the base model, SFT, and LoRA, indicating broader support across singular directions.

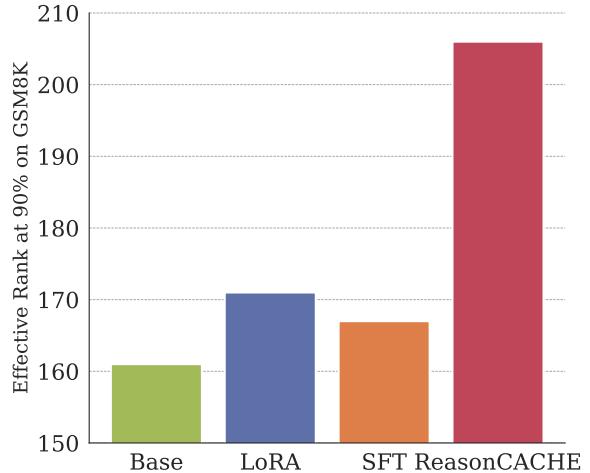


Figure 16 Effective rank of last-layer token representations, defined as the rank at which the cumulative singular-value mass reaches 90%. REASONCACHE attains the highest effective rank among all methods.

low-dimensional value subspace; REASONCACHE injects directions orthogonal to this span; and these directions propagate to yield higher-dimensional downstream representations.

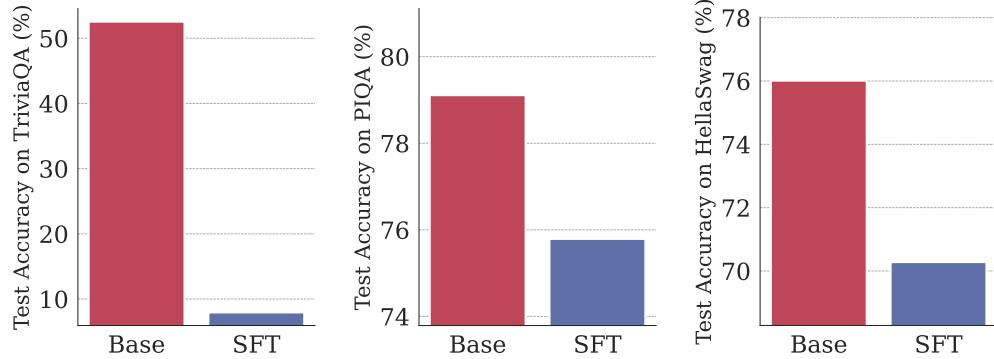


Figure 17 Accuracy on TriviaQA, PIQA, and HellaSwag for LLaMA-2 7B before and after supervised fine-tuning (SFT) on MetaMathQA, illustrating catastrophic forgetting under in-weight adaptation.

C.3 In-Weight Adaptation Exhibits Catastrophic Forgetting Unlike ReasonCache

Finally, we test whether in-weight adaptation degrades general-purpose capabilities outside the training distribution. Figure 17 reports accuracy on TriviaQA, PIQA, and HellaSwag for LLaMA-2 7B before and after supervised fine-tuning (SFT) on MetaMathQA. SFT substantially reduces performance on all three benchmarks relative to the base model, consistent with catastrophic forgetting from task-specific weight updates (Kirkpatrick et al., 2017).

By contrast, REASONCACHE and LoRA leave the pretrained weights unchanged and are reversible by construction: removing the learned prefixes or adapters exactly recovers the base model. We therefore omit them from Figure 17, since their post-removal performance matches the base model. Thus, while SFT can improve task performance but may overwrite broad capabilities, whereas parameter-isolated adaptation preserves the base model while enabling similar to better task-specific performance as shown in Figure 2.

D Extended Related Work

This section provides an extended discussion of related work, expanding on the summary in [Section 5](#).

In-Context Learning and Its Limitations. In-context learning (ICL) enables LLMs to adapt to new tasks from demonstrations without gradient updates ([Brown et al., 2020](#)). While remarkably sample-efficient for format control and simple tasks ([Min et al., 2022](#)), ICL faces fundamental limitations when scaling to tasks that require more demonstrations such as complex reasoning. These limitations manifest along several axes: *positional bias*, where models struggle to access information in the middle of long contexts ([Liu et al., 2024](#)); *stability degradation*, where increasing context length introduces attention noise and unreliable performance ([Hong et al., 2025](#)); *shallow learning*, where learning is primarily driven by deducing patterns from prompt regularities, resulting in limited generalization ([de Wynter, 2025](#)); and *diminishing returns*, where as the number of ICL demonstrations increases from a few to many, the performance of LLMs initially improves but then plateaus and can even decline ([Agarwal et al., 2024](#); [Zhang et al., 2025](#); [Zhang and Bottou, 2025](#)). These limitations motivate learning to compress demonstrations into a compact KV-cache that sidesteps the pathologies of long contexts.

Prompting and Eliciting Reasoning. Chain-of-thought prompting ([Wei et al., 2022](#)) showed that intermediate reasoning steps can elicit latent reasoning abilities. Extensions include least-to-most prompting ([Zhou et al., 2022](#)), which decomposes problems into subproblems, self-consistency ([Wang et al., 2022](#)), which improves reliability via multiple sampled chains, and zero-shot chain-of-thought ([Kojima et al., 2022](#)), which activates reasoning with simple prompt phrases. More recently, schema-activated ICL ([Chen et al., 2025](#)) has proposed retrieving abstract reasoning templates to bridge the gap between pattern priming (surface-level mimicry) and complex reasoning.

Prompt Tuning and Prefix Tuning. Prompt tuning ([Lester et al., 2021](#)) introduced learnable continuous embeddings prepended at the input layer, enabling task adaptation without modifying pretrained weights. Prefix tuning ([Li and Liang, 2021](#)) extended this by learning key-value vectors at every attention layer, providing greater representational flexibility. P-tuning v2 ([Liu et al., 2021](#)) showed that deep prompt tuning can match full fine-tuning across scales and tasks. Recent work has analyzed the statistical benefits of prefix reparameterization ([Le et al., 2024](#)), proposed architectural improvements such as decoupling the attention paid to the prefixes and regular context ([Wang et al., 2025](#)), and used Bayesian frameworks to demonstrate that soft prefixes can manipulate activations in ways inaccessible to discrete tokens ([Genewein et al., 2025](#)). While these methods were primarily evaluated on classification and generation tasks, we study whether prefix tuning can acquire complex reasoning skills—a capability typically associated with weight updates.

Context Compression and Distillation. Several methods aim to compress long contexts into compact representations. Gist tokens ([Mu et al., 2023](#)) train an LM to compress prompts into a small number of virtual tokens that can be cached and reused. AutoCompressor ([Chevalier et al., 2023](#)) adapts LMs to compress long contexts into compact summary vectors that are used as soft prompts. LLMLingua ([Jiang et al., 2023](#)) performs coarse-to-fine prompt compression by selecting/pruning tokens using LM-scoring signals (e.g., perplexity), while LLMLingua-2 ([Pan et al., 2024](#)) uses data distillation to learn task-agnostic compression policies. Natural language compression ([Chuang et al., 2024](#)) produces shorter textual “capsule prompts” for better transferability across LLMs. ICAE ([Ge et al., 2023](#)) compresses long context into compact memory slots via an in-context autoencoding objective. Cartridges ([Eyuboglu et al., 2025](#)) compress massive contexts into learnable KV-caches via self-study, allowing fast inference with frozen backbones. Deep context distillation ([Caccia et al., 2025](#)) trains document-level knowledge modules (LoRA modules) to simulate a teacher model’s hidden states and logits when the teacher has full document access. These methods primarily compress or modularize *contextual information*, rather than learning a persistent complex skill (e.g., reasoning) beyond what is expressed in the compressed representation.

Memory and Continual Learning. Our work relates to the broader question of how LLMs can acquire and retain knowledge. Standard in-weight learning comes with its own issues, such as the reversal curse ([Berglund et al., 2023](#)), where models fail to generalize bidirectional relationships. In contrast, Lampinen et al. ([Lampinen et al., 2025](#)) demonstrate that in-context learning naturally avoids these failures, generalizing flexibly where

in-weight fine-tuning breaks down. However, ICL is ephemeral and limited by context length. Continual learning via sparse memory fine-tuning (Lin et al., 2025) updates only the most relevant memory slots to prevent catastrophic forgetting. Memory Mosaics (Zhang et al., 2024) replace standard attention with networks of associative memories, demonstrating superior compositional generalization. Our perspective differs: we view the KV-cache as a *learnable memory interface* that spans a spectrum from raw tokens (recovering ICL) to gradient-optimized vectors (approaching in-weight learning). This framing opens a broader design space for adaptation, where learned prefixes serve as modular, pluggable “skill caches” that can be composed or swapped without modifying the base model. Prefix tuning, however, is *not* a true continual/test-time learner: the KV-cache is trained offline and then frozen. Developing methods where the KV-cache remains plastic at test time is an exciting direction for future work.

In-Context Optimization and Test-Time Learning. Recent work has framed transformers as performing implicit optimization over context (Von Oswald et al., 2023; Mittal et al., 2025). Context tuning (Lu et al., 2025) initializes learnable prefixes from ICL demonstrations, showing that optimization works best when grounded in context. Function vectors (Todd et al., 2023) identify that specific attention heads transport compact task representations, providing mechanistic insight into why prefix tuning works. Generative adapters (Chen et al., 2024) use hypernetworks to generate LoRA adapters based on input context.

Reasoning via Fine-Tuning and Reinforcement Learning. The dominant approach for instilling reasoning capabilities involves weight updates. DeepSeek-R1 (Guo et al., 2025) and OpenAI’s reasoning models (OpenAI, 2024) use reinforcement learning to incentivize reasoning. Data curation efforts (Guha et al., 2025; Yu et al., 2023; Ye et al., 2025) focus on constructing high-quality reasoning traces. Methods such as s1 (Muennighoff et al., 2025) and LIMO (Ye et al., 2025) show that a small number of high-quality examples (<1000) can instill reasoning skills, though individual examples can be extremely long (>10k tokens). Although prior theoretical work suggests that prefix tuning may be less expressive than full fine-tuning (Petrov et al., 2023), our work shows that prefix tuning offers an alternative post-training interface: learning reasoning skills into a portable KV-cache that can be plugged into a frozen model, combining the modularity of in-context methods with the depth of weight-based learning.