# PrimeTTT: Priming Test-Time Training for LLMs via Meta-Learned Initialisation

Anonymous ACL submission

## Abstract

Test-time training (TTT) adapts a frozen large language model (LLM) at inference time by updating prompt-specific parameters before generation. It is especially appealing for long-context reasoning. However, vanilla TTT yields only marginal gains on modern LLMs. We trace this bottleneck to two coupled failures: the inner loop is unprimed, because LLMs are trained as static endpoints rather than initialization points for test-time gradient adaptation; and the proxy objective is misaligned, because prompt-masked next-token prediction (PM-NTP) rewards prompt memorization rather than answer correctness. We propose **PrimeTTT**, a lightweight TTT method that casts this adaptation problem as meta-learning. A compact hypernetwork emits a prompt-specific LoRA initialization for the inner loop, trained with a MAML-style second-order objective: the inner loop descends the test-time-available PM-NTP loss, while the outer loss is answer-conditioned NTP on the gold answer and is backpropagated through the unrolled inner steps. Across out-of-distribution benchmarks disjoint from training, PrimeTTT consistently outperforms strong TTT baselines and transfers across model families and scales.

## 1 Introduction

Long-context reasoning has become a central capability of large language models (LLMs) (Grattafiori et al., 2024; Yang et al., 2025), yet frozen LLMs degrade sharply once inputs exceed the lengths they were trained on (Liu et al., 2024; Hsieh et al., 2024; Bai et al., 2024). Existing remedies recover this capability through prompting (Wei et al., 2022), retrieval augmentation (Robertson and Zaragoza, 2009; Lewis et al., 2020), parameter-efficient fine-tuning (Hu et al., 2022; Lester et al., 2021), or test-time training (TTT) (Sun et al., 2020; Hardt and Sun, 2024; Akyürek et al., 2024). Among these, TTT is designed as a mechanism that updates the model per prompt, writing prompt-specific information into the weights via a few SGD steps on an unsupervised objective, typically prompt-masked next-token prediction (PM-NTP), before generating. On modern LLMs, however, vanilla TTT extracts only marginal gains, and enlarging the inner-loop budget (more SGD steps and/or more PM-NTP tokens per prompt) does not close the gap (reported in §3; consistent with reports in Bansal et al., 2025), indicating that the bottleneck is structural rather than computational.

We take a different perspective. The two design choices vanilla TTT inherits without examination (a zero-initialised adapter (Hu et al., 2022) and an unsupervised proxy loss) are each defensible in isolation but jointly pathological. In §3, we make this concrete: **the loop is not primed**, because LLMs are trained as good static endpoints rather than good starting points for a downstream gradient (Finn et al., 2017; Nichol et al., 2018), so a zero-initialised adapter places the inner loop at a prompt-agnostic initialization; and **the proxy is misaligned**, because PM-NTP rewards prompt memorisation rather than answer correctness, so its gradient need not point toward better answers. Existing remedies attack only one side: one-shot hypernetworks such as T2L (Charakorn et al., 2025; Ha et al., 2016) fix the starting point but leave the gradient misaligned; reward-shaping TTT variants (Bansal et al., 2025) adjust the objective but still lack supervised answer feedback at test time. The inner-loop initialization and objective must therefore be *co-designed*.

We propose **PrimeTTT**, a lightweight TTT method that enforces this co-design via a compact hypernetwork trained with a MAML-style second-order objective (Finn et al., 2017; Antoniou et al., 2018): the inner loop descends PM-NTP, while the outer loss is answer-conditioned NTP (AC-NTP) on the gold answer. At test time, the hypernetwork emits a prompt-specific LoRA initialisation in a

single forward pass, replacing the zero-initialised adapter; the same $K$-step PM-NTP adaptation then refines it before generation. Because the outer loss is backpropagated through the unrolled inner steps, the resulting Hessian-vector terms provide the curvature signal needed to shape the PM-NTP update direction toward lower answer loss.

Our contributions are as summarized follows:

- We identify and empirically substantiate a coupled failure mode of vanilla TTT on LLMs: a train-test objective mismatch between PM-NTP (proxy) and AC-NTP (target) and an unprimed inner loop launched from a prompt-agnostic point (§3).
- We introduce PrimeTTT, a lightweight hypernetwork-initialised TTT method learns prompt-specific LoRA initializations with a MAML-style second-order objective, co-designing the warm start with the PM-NTP update geometry (§4).
- PrimeTTT is *efficient* and *transferable*. It adds only $\sim 2\%$ inference overhead beyond the same $K$-step PM-NTP adaptation. A single synthetic-trained checkpoint improves three backbones (Llama-3.1-8B, Qwen3-4B, GPT-oss-20B) on four out-of-distribution QA benchmarks (§5).

## 2 Related Work

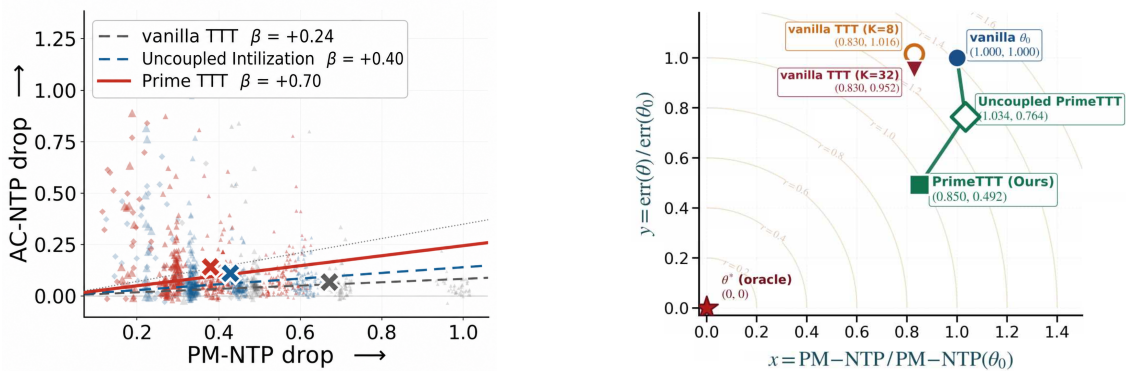
**Long-context limits and static adaptation.** Frozen LLMs degrade beyond pre-training lengths (Liu et al., 2024; Hsieh et al., 2024; Bai et al., 2024). Position interpolation (Peng et al., 2024; Chen et al., 2023) extends the addressable context but not long-range accuracy, while long-context finetuning (Chen et al., 2024) demands expensive supervision. Prompting (Wei et al., 2022; Kojima et al., 2022), retrieval (Robertson and Zaragoza, 2009; Lewis et al., 2020), and parameter-efficient finetuning (Hu et al., 2022; Lester et al., 2021) reshape inputs or freeze a single adapter, offering no per-prompt update at test time.

**Test-time training, hypernetworks, and meta-learning.** Test-time training (Sun et al., 2020; Wang et al., 2020) closes this gap by updating the model per input on a self-supervised loss. LLM variants retrieve neighbours (Hardt and Sun, 2024), actively select data (Hübotter et al., 2025), run unsupervised SGD on the prompt (Akyürek et al., 2024), restrict updates to query projections (Bansal et al., 2025), fold TTT into each layer (Sun et al., 2024), or apply MAML-style end-to-end training (Tandon et al., 2025). However, all these approaches start from a zero-initialised adapter or learn the inner loop without a per-prompt warm-start. Hypernetworks emit parameters from conditioning input (Ha et al., 2016); Text-to-LoRA (Charakorn et al., 2025) predicts a per-task LoRA in one forward pass but takes no subsequent gradient step. MAML (Finn et al., 2017) trains an initialisation for $K$ inner steps, with first-order (Nichol et al., 2018), stability (Antoniou et al., 2018), and implicit-gradient (Rajeswaran et al., 2019) variants. PrimeTTT brings these ideas into the TTT setting: a hypernetwork supplies the per-prompt warm-start, and MAML teaches it to land where the test-time unsupervised gradient descends toward answer correctness.

## 3 Why TTT Underperforms on LLMs: An Empirical Study

We ask why vanilla TTT, effective for smaller models (Sun et al., 2020; Hardt and Sun, 2024), recovers so little on modern LLMs. Table 2 reports the symptom: across three backbones, vanilla TTT improves over the frozen base by at most $0.8$ points on average and regresses on several cells, and quadrupling the inner-loop budget to $K{=}32$ does not help (e.g. Llama-3.1-8B drops from $80.6$ to $79.6$ overall). The bottleneck is not how many steps the inner loop takes; we trace instead to two coupled, measurable pathologies: *where* the inner loop starts and *what* it optimizes.

**Pathology 1: Train-test objective mismatch.** Vanilla TTT trains on $\mathcal{L}_{\text{PM}}$ (PM-NTP, prompt memorization and fitting) but is evaluated under $\mathcal{L}_{\text{AC}}$ (AC-NTP, answer tokens). We measure the objective geometry using the gradient cosine alignment $\rho(\phi) := \cos(\nabla_\phi \mathcal{L}_{\text{PM}}, \nabla_\phi \mathcal{L}_{\text{AC}})$ (computed offline as $\mathcal{L}_{\text{AC}}$ requires gold answers). To summarize how this geometry manifests along the actual inner-loop trajectory, we report the transfer slope $\beta$: the regression slope of AC-NTP decrease against PM-NTP decrease over the $K{=}8$ inner steps. Figure 1a shows $\beta{=}0.24$ for vanilla TTT on Qwen3-4B, meaning that a unit decrease in the prompt-memorization loss yields less than a quarter-unit decrease in the answer-conditioned loss. Oracle fits show the mismatch more starkly. A LoRA fit purely to $\mathcal{L}_{\text{AC}}$ drives AC-NTP nearly to zero




(a) Per-step PM-to-AC transfer (Qwen3-4B). Vanilla TTT shows weak transfer ($\beta$=0.24); a hypernetwork-only warm-start (no MAML coupling) improves it ($\beta$=0.40); PrimeTTT aligns best ($\beta$=0.70).

(b) Distance to the per-prompt oracle adapter ($\theta^*$). Vanilla TTT remains far from the oracle even at $K$=32; only PrimeTTT gets closer to the oracle and crosses inside the unit ring.

Figure 1: **Two empirical pathologies of vanilla TTT on LLMs.** (a) A train-test objective mismatch (weak PM-to-AC transfer). (b) An unprimed inner loop never approaches the per-prompt oracle.

nats/token but increases PM-NTP above the frozen baseline. Conversely, a Full-NTP oracle that fits both prompt and answer tokens plateaus at 2.28 nats/token on $\mathcal{L}_{\text{AC}}$, two orders of magnitude worse than the AC-only oracle (Appendix F, Table 10). Thus PM-NTP is not a relaxation of AC-NTP, but a different objective.

**Pathology 2: An unprimed inner loop.** Vanilla TTT starts every test-time update from $\phi_0$=0, the same prompt-agnostic adapter for all examples. To estimate where a useful prompt-specific update should move, we construct a per-prompt ORACLE adapter $\phi^\star$ by fitting $\mathcal{L}_{\text{AC}}$ offline; his gives the best answer-supervised adapter reachable within the LoRA space Figure 1b shows that vanilla TTT does not get close to this oracle: after $K$=8 steps it remains at distance 1.41, and even increasing the budget to $K$=32 only reduces the distance to 1.26 (10% improvement for 4× compute). Thus, the failure is not simply that vanilla TTT needs more inner-loop steps. Instead, the starting point is wrong. We therefore train a test-to-LoRA (T2L) initializer to predict an adapter close to the ORACLE before test-time updates begin. We further compare whether coupling this initialization with the inner-loop objective is necessary. The results show that T2L alone helps, but the coupled PrimeTTT design is necessary and gives the best performance.

**The pathologies are coupled.** The two issues are not fully separable. Matching the offline oracle gives a better warm start, but does not guarantee that the subsequent PM-NTP update is well aligned with AC-NTP (the uncoupled warm-start has slope $\beta$=0.40 in Figure 1a). Conversely, improving the inner-loop loss is limited if the update starts from a point with weak downstream gradient signal. We therefore optimise the warm start *through* the inner loop, so that $\phi_0$ is encouraged to land where PM-NTP descent also reduces AC-NTP. §4 introduces our PrimeTTT which implements this idea with asymmetric second-order MAML.

## 4 PrimeTTT

PrimeTTT combines a hypernetwork $H_\psi$ that emits a prompt-specific LoRA with a standard unsupervised inner loop. The two components share a single end-to-end objective: $H_\psi$ is trained so that $K$ steps of unsupervised TTT gradient steps, launched from $H_\psi(x)$, land in a configuration with low $\mathcal{L}_{\text{AC}}$ on the gold answer. Training runs in three offline phases. Stage 1 (§4.2) collects per-prompt oracle adapters as supervision targets. Stage 2 (§4.3) pretrains $H_\psi$ to regress these oracles. Stage 3 (§4.4) fine-tunes $H_\psi$ with a MAML-style objective that differentiates through the inner loop. The test-time protocol (§4.5) looks identical to the uncoupled version.

### 4.1 Setup

We adopt the notation of §3: $\theta_0$ are the frozen base weights, $\phi$ is the LoRA (Hu et al., 2022) that hypernetwork $H_\psi$ generated. Here $x = (x_1, \ldots, x_m)$ is the prompt and $y = (y_1, \ldots, y_n)$ is the gold answer. The test-time objective is prompt-masked NTP, $\mathcal{L}_{\text{PM}}(x; \theta_0, \phi) = -\frac{1}{m}\sum_{t=1}^{m} \log p_{\theta_0,\phi}(x_t \mid x_{<t})$, while the offline target is answer-conditioned NTP, $\mathcal{L}_{\text{AC}}(y \mid x; \theta_0, \phi) = -\frac{1}{n}\sum_{t=1}^{n} \log p_{\theta_0,\phi}(y_t \mid$

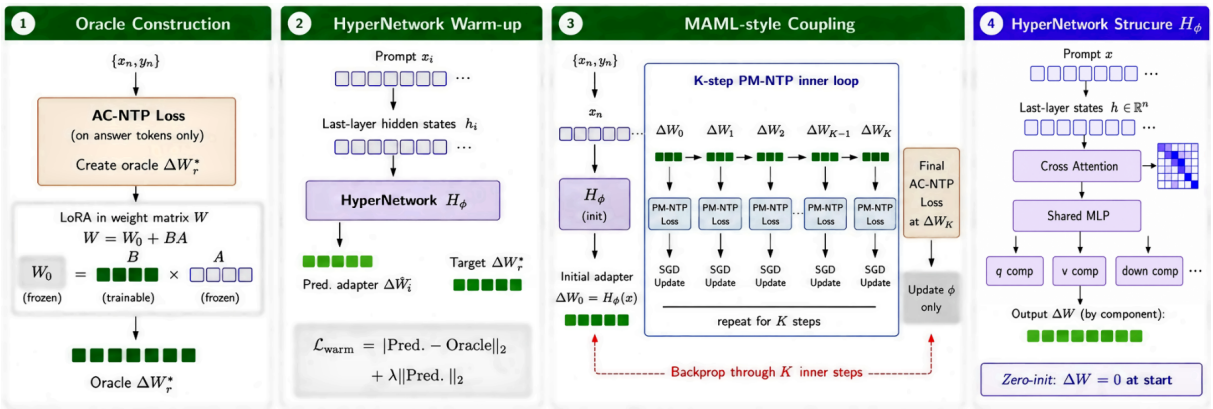



Figure 2: **The PrimeTTT pipeline.** Stage 1 constructs per-prompt oracle LoRA adapters $B_i^\star$ by optimizing AC-NTP on answer tokens only. Stage 2 warm-starts the hypernetwork $H_\psi$ to predict $\hat{B}_i$ matching $B_i^\star$ under an $\ell_1$ regression loss. Stage 3 couples $H_\psi$ to the inner loop: starting from $\phi_0 = H_\psi(x_i)$, $K$ unsupervised PM-NTP steps produce $\phi_K$, and the outer AC-NTP loss on the gold answer at $\phi_K$ is backpropagated through all $K$ inner updates (red dashed branch), retaining second-order terms. Oracles are used only as warm-up targets and never at inference.

$x, y_{<t})$. Thus the inner loop always uses $\mathcal{L}_{\text{PM}}$, and $\mathcal{L}_{\text{AC}}$ is used only for offline oracle construction and meta-training. We parameterize each LoRA update as $\Delta W = BA$, fixing $A$ at random initialisation and treating $B$ as the adapter parameter ($\phi$) updated by the inner loop:

$$\phi_k = \phi_{k-1} - \eta \nabla_\phi \mathcal{L}_{\text{PM}}(x;\, \theta_0, \phi_{k-1}), \qquad (1)$$

for $k{=}1, \ldots, K$ and inner learning rate $\eta$.

### 4.2 Stage 1: Collecting Oracles

For each training pair $(x_i, y_i)$ we fit an oracle LoRA by minimising $\mathcal{L}_{\text{AC}}$ with prompt tokens masked:

$$\phi_i^\star = \arg\min_{\phi} \mathcal{L}_{\text{AC}}(y_i \mid x_i;\, \theta_0, \phi). \qquad (2)$$

Masking the prompt tokens makes the oracle target answer prediction rather than prompt reconstruction. Including prompt tokens in the oracle loss yields a full-NTP oracle with substantially worse answer likelihood (Appendix F). We fit a rank-8 LoRA on $W_q, W_v, W_{\text{down}}$ in the top layers only, targeting modules that control long-context retrieval and readout: $W_q$ affects which positions attention retrieves, while $W_v$ and $W_{\text{down}}$ affect what information is read and written back for answer formation. This choice follows prior TTT studies (Tandon et al., 2025; Feng et al., 2026), and is supported by the cross-model transferability of oracle adapters in canonical KV space (§5.6). For each adapted matrix, only the matrix $B \in \mathbb{R}^{48\times\cdot}$ is learned. We define $B_{\text{base}} = \frac{1}{N}\sum_i B_i^\star$ and train hypernetwork $H_\psi$ to predict the residual $\Delta B_i = B_i^\star - B_{\text{base}}$. These oracle adapters serve only as warm-start training targets and are never used at inference.

### 4.3 Stage 2: Warming Up the Hypernetwork

Following the T2L hypernetwork objective (Charakorn et al., 2025), we train hypernetwork $H_\psi$ to regress the oracle residuals via an $\ell_1$ loss with $\ell_2$ shrinkage,

$$\mathcal{L}_{\text{warm}}(\psi) = \frac{1}{S}\sum_{s=1}^{S}\Big[\|H_\psi(x_i)_s - \Delta B_{i,s}\|_1 + \lambda\|H_\psi(x_i)_s\|_2\Big], \qquad (3)$$

where $s$ indexes LoRA slots. This stage is ordinary supervised regression, with no backpropagation through downstream SGD; it serves only as a cheap, stable initialization for the more expensive Stage 3 meta-training.

The architecture, layer dimensions, and all hyperparameters, are given in Appendix D. This provides a useful warm start, but it optimizes *oracle proximity* rather than *compatibility with the downstream SGD trajectory*. Stage 3 closes this gap.

### 4.4 Stage 3: Coupling via MAML

Stage 2 trains $H_\psi$ to match the offline oracle, but oracle proximity is not the same as TTT compatibility. The oracle minimises answer loss, whereas the test-time inner loop can only descend PM-NTP. In practice, the Stage 2 warm-start $\phi_0 = H_\psi(x)$ may lie at a point where PM-NTP descent stalls or drifts away from the answer, as shown in Figure 1a and §5.7. Stage 3 therefore follows MAML (Finn et al., 2017) and trains $H_\psi$ *through* the unrolled inner loop, so it learns a starting point from which $K$ PM-NTP steps actually reduce AC-NTP outer loss rather than merely resemble the oracle in isolation.

Let $\text{InnerLoop}(\phi_0, x; K, \eta)$ denote $K$ steps of PM-NTP SGD (Eq. 1) launched from $\phi_0$, produc-

ing the trajectory $\phi_0 \to \phi_1 \to \cdots \to \phi_K$. We fine-tune $H_\psi$ by minimising

$$\mathcal{J}(\psi) = \mathbb{E}_{(x,y)}\Big[\mathcal{L}_{\text{AC}}\big(y \,|\, x;\, \theta_0,\, \text{InnerLoop}(H_\psi(x), x; K, \eta)\big)\Big]. \tag{4}$$

The inner loop uses $\mathcal{L}_{\text{PM}}$ because that is all that is available at test time; the outer loss uses $\mathcal{L}_{\text{AC}}$ because gold answers are available offline. This asymmetry formalises the coupling: $H_\psi$ is judged by the answer loss at $\phi_K$, not at $\phi_0$, so it must produce a $\phi_0$ that is *useful for subsequent PM-NTP updates*, not one that already looks correct.

The mechanism behind why this works is straightforward. With $\rho(\phi)$ from §3, a single PM-NTP step gives $\mathcal{L}_{\text{AC}}(\phi_1) - \mathcal{L}_{\text{AC}}(\phi_0) = -\eta \,\|\nabla \mathcal{L}_{\text{AC}}(\phi_0)\| \,\|\nabla \mathcal{L}_{\text{PM}}(\phi_0)\| \,\rho(\phi_0) + \mathcal{O}(\eta^2)$, so as long as $\rho(\phi_0) > 0$ and $\eta$ is small, the unsupervised PM-NTP step lowers AC-NTP. Over $K$ steps, the first-order AC-NTP change accumulates along the trajectory, with each step helping when the PM-NTP and AC-NTP gradients remain aligned. Stage 3 meta-learns $H_\psi$ through these unrolled updates so that the initialization lands in regions where PM-NTP descent tends to reduce AC-NTP. Oracles drop out of the loss once Stage 3 begins.

Differentiating Eq. 4 makes this concrete. For one example, let $J_k = I - \eta \nabla^2_\phi \mathcal{L}_{\text{PM}}(x; \phi_k)$ be the Jacobian of the $k$-th PM-NTP update. Since $\phi_K$ depends on $\psi$ through the full chain $\phi_0 \to \phi_1 \to \cdots \to \phi_K$, the meta-gradient is

$$\nabla_\psi \mathcal{J} = \underbrace{\frac{\partial \mathcal{L}_{\text{AC}}}{\partial \phi_K}}_{\text{(i)}} \cdot \underbrace{J_{K-1} J_{K-2} \cdots J_0}_{\text{(ii)}} \cdot \underbrace{\frac{\partial H_\psi(x)}{\partial \psi}}_{\text{(iii)}}. \tag{5}$$

The three factors have distinct roles. Term (i) is the answer-direction signal at the endpoint: *how should $\phi_K$ move to lower AC-NTP?* Term (iii) is the hypernetwork Jacobian: *how does changing $\psi$ move the initialization $\phi_0$?* Term (ii) is the bridge between them: the Hessian factors transport the endpoint answer signal backward through the PM-NTP update landscape. This is the coupling between the proxy and target losses. If these Hessian terms are dropped, as in first-order MAML (Nichol et al., 2018; Antoniou et al., 2018), the meta-gradient no longer accounts for how PM-NTP curvature shapes the path from $\phi_0$ to $\phi_K$. In our ablation, this first-order variant captures only 59% of the Stage 3 gain (Table 4, row F), motivating the extra memory cost of backpropagating through the full unrolled chain. Implementation details are in Appendix D.

### 4.5 Test-Time Inference

At inference, PrimeTTT uses the same PM-NTP inner loop as vanilla TTT and changes only the initialization. only its starting point. Given a test prompt $x$, a single forward pass through $H_\psi$ produces the LoRA adapter $\phi_0 = H_\psi(x)$. We then run $K$=8 steps of PM-NTP SGD (Eq. 1) to refine $\phi_0$ into $\phi_K$, and generate with the model adapted by $\phi_K$. The LoRA is discarded afterwards. The hypernetwork forward pass adds roughly 2% to inference cost, and no gold answers enter the test-time pipeline.

## 5 Experiments

### 5.1 Setup

**Models and long-context extension.** We evaluate PrimeTTT with Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3-4B-Instruct (Yang et al., 2025), and GPT-oss-20B (Agarwal et al., 2025) (3-bit MLX quantization version). Base weights are frozen; all adaptation updates only a rank-8 LoRA on the top 16 layers, attached to `q_proj/v_proj/down_proj`. All long-context evaluations use YaRN-extended bases (Peng et al., 2024) with scale 4× and effective context length 128k. We additionally evaluate at 64k for stress testing. Every method in Table 2, including the frozen baseline, is evaluated on the same YaRN-extended weights, so position interpolation is a constant across rows.

**Benchmarks.** We employ two complementary long-context evaluation benchmarks: RULER (Hsieh et al., 2024), which gives a controlled length sweep (4k, 8k, 16k, 64k), and LongBench-v2 (Bai et al., 2025) (LBv2), which tests heterogeneous long-context QA. The covered task categories are listed in Table 1. We use an 80/20 train–test split. All results are reported as contains-rate (%), measuring whether the generated response contains the gold answer.

Table 1: Task categories covered by RULER and LongBench-v2.

| Benchmark | Categories |
|---|---|
| RULER | NIAH single, NIAH multi, NIAH distractor, multi-query, multi-hop, QA, variable tracking |
| LongBench-v2 | single-doc QA, multi-doc QA, long ICL, long dialogue, long structured, code repo |

Table 2: Main results on long-context reasoning benchmarks (contains-rate, %). Best result per model–column in **bold**. ⋆Vanilla TTT with $K{=}32$ inner-loop steps ($4\times$ the default $K{=}8$). The *Overall* column is contains-rate over the union of all evaluation prompts.

| Model | Method | r-4k | r-8k | r-16k | r-64k | LBv2 | Overall |
|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Ins. | Vanilla | 92.5 | 90.7 | 82.9 | 14.0 | 25.0 | 79.8 |
| | CoT | 91.0 | 85.8 | 77.1 | 11.0 | 22.9 | 75.8 |
| | RAG | 84.3 | 84.0 | 77.7 | 33.3 | 27.1 | 75.0 |
| | Prompt Tuning | 93.4 | 91.0 | 83.5 | 16.8 | 24.2 | 80.3 |
| | SFT | 92.5 | 91.4 | 84.0 | 13.0 | 18.8 | 79.8 |
| | qTTT | 91.8 | 88.9 | 82.3 | 12.0 | 25.0 | 78.8 |
| | Vanilla TTT | 93.3 | **92.6** | 84.0 | 15.0 | 20.8 | 80.6 |
| | Vanilla TTT ($K{=}32$)⋆ | 94.0 | 85.2 | 86.9 | 16.0 | 21.2 | 79.6 |
| | T2L | 92.7 | 91.5 | 86.4 | 22.1 | 26.4 | 81.7 |
| | TTT-NN | 91.2 | 90.4 | 84.7 | 35.7 | 24.6 | 80.8 |
| | SIFT | 90.8 | 89.1 | 83.6 | 28.4 | 22.3 | 79.4 |
| | PRIMETTT | **97.0** | 92.0 | **90.9** | **41.2** | **37.5** | **86.1** |
| Qwen3-4B-Ins.-0725 | Vanilla | 79.9 | 77.8 | 74.3 | 19.0 | 17.5 | 69.5 |
| | CoT | 17.2 | 13.6 | 10.3 | 8.0 | 14.6 | 13.3 |
| | RAG | 82.8 | 77.2 | 73.7 | 42.9 | 27.1 | 71.7 |
| | Prompt Tuning | 80.2 | 78.4 | 74.8 | 20.5 | 18.4 | 70.1 |
| | SFT | 80.6 | 77.8 | 74.3 | 18.0 | 17.0 | 69.6 |
| | qTTT | 80.6 | 77.8 | 74.9 | 18.5 | 17.0 | 69.8 |
| | Vanilla TTT | 79.9 | 77.8 | 74.9 | 20.0 | 19.5 | 70.0 |
| | Vanilla TTT ($K{=}32$)⋆ | 80.6 | 77.8 | 79.4 | 22.0 | **37.5** | 73.3 |
| | T2L | 82.5 | 79.6 | 77.2 | 26.6 | 21.3 | 72.3 |
| | TTT-NN | 83.4 | 80.8 | 78.4 | 40.2 | 23.7 | 74.0 |
| | SIFT | 82.6 | 79.7 | 77.0 | 32.8 | 21.9 | 72.6 |
| | PRIMETTT | **91.0** | **87.7** | **88.0** | **50.4** | 35.4 | **82.5** |
| GPT-oss-20B-tp3 | Vanilla | 91.0 | 76.5 | 70.3 | 28.6 | 37.5 | 72.8 |
| | CoT | 88.1 | 78.4 | 69.7 | 23.8 | 39.6 | 72.4 |
| | RAG | 85.8 | 70.4 | 68.6 | 52.4 | 33.3 | 69.6 |
| | Prompt Tuning | 90.3 | 77.8 | 71.4 | 20.5 | 22.0 | 71.6 |
| | SFT | 90.6 | 77.0 | 71.0 | 28.0 | 32.5 | 72.6 |
| | qTTT | 90.1 | 76.8 | 70.7 | 27.5 | 35.8 | 72.6 |
| | Vanilla TTT | 91.5 | 78.4 | 72.1 | 31.0 | 38.5 | 74.2 |
| | Vanilla TTT ($K{=}32$)⋆ | 92.3 | 80.1 | 74.6 | 34.5 | **45.2** | 76.5 |
| | T2L | 91.0 | 78.8 | 73.5 | 30.4 | 38.9 | 74.7 |
| | TTT-NN | 90.8 | 78.5 | 73.0 | 39.5 | 37.5 | 74.6 |
| | SIFT | 90.5 | 78.2 | 72.5 | 35.0 | 36.0 | 74.0 |
| | PRIMETTT | **94.8** | **82.5** | **78.4** | **56.0** | 44.7 | **79.8** |

**Baselines.** We group baselines by their adaptation mechanism.

*(i) Prompt-only.* **Vanilla** is the YaRN-extended base (Peng et al., 2024) evaluated as-is. **CoT** adds zero-shot chain-of-thought prompting (Wei et al., 2022; Kojima et al., 2022). **RAG** prepends BM25 top-3 chunks (Robertson and Zaragoza, 2009; Lewis et al., 2020) (512 tokens each, no reranking).

*(ii) Parameter-efficient finetuning on the same NIAH data.* **Prompt Tuning** (Lester et al., 2021) learns soft prompt tokens. **SFT** fine-tunes a LoRA (Hu et al., 2022) on the NIAH-style data (Kamradt, 2023; Hsieh et al., 2024) and freezes it at test time.

*(iii) Standard test-time training.* **qTTT** (Bansal et al., 2025) restricts the inner-loop update to the query projections $W_Q$ after a single KV-cache prefill. **Vanilla TTT** (Akyürek et al., 2024) (see also Sun et al., 2020; Hardt and Sun, 2024) runs $K{=}8$ steps of prompt-masked NTP-SGD on a zero-initialised LoRA; **Vanilla TTT** ($K{=}32$) gives it $4\times$ the inner-loop budget.

*(iv) Hypernetwork- and retrieval-based TTT.* **T2L** (Charakorn et al., 2025) emits a per-prompt LoRA from a text-conditioned hypernetwork with-

out an inner loop. **TTT-NN** (Hardt and Sun, 2024) works as in context learning of few-shot samples of the base on retrieved nearest neighbours of the test prompt. **SIFT** (Hübotter et al., 2025) actively selects training points at test time to fine-tune on.
*(v) Ours.* **PrimeTTT** is the full method (hypernet warm-start plus the coupled polish loop).

### 5.2 Main Results

Table 2 shows that PrimeTTT achieves the best overall score on all three backbones, with gains of $+4.4$, $+8.5$, and $+3.3$ points over the strongest baseline on Llama-3.1-8B, Qwen3-4B, and GPT-oss-20B, respectively. The gains are most pronounced on the controlled RULER sweep, especially at 64k context, where PrimeTTT is best for every backbone. LBv2 is more mixed: PrimeTTT is best on Llama-3.1-8B, while larger-budget vanilla TTT remains competitive on Qwen3-4B and GPT-oss-20B. This pattern is still consistent with our diagnosis: vanilla TTT delivers only marginal gains; PrimeTTT is most effective when the limiting factor is making the test-time PM-NTP updates informative, while benchmark/model-specific effects can change how much a larger inner-loop budget helps.

### 5.3 Out-of-Distribution Generalisation

The hypernetwork is trained on synthetic NIAH-style prompts, so a natural concern is that it only helps retrieval-shaped tasks. We test this on four held-out QA benchmarks that share no structure with the training data: HotpotQA and 2WikiMultihopQA (multi-hop retrieval, token-F1), MuSR (multi-step soft reasoning, MC accuracy), and QuALITY (long-document comprehension, MC accuracy), reusing the same hypernetwork checkpoints with no retraining (Table 3).

**The gains transfer.** PrimeTTT is the strongest method across all evaluated benchmarks and models. The hypernetwork-only warm start, which initializes the adapter from the Stage 2 but omits the MAML coupling in Stage 3, is consistently second. Over the frozen base, PrimeTTT gains $+3$ to $+6$ on multi-hop QA (HotpotQA, 2WikiMHQA) and $+4$ to $+6$ on MuSR and QuALITY, leading vanilla TTT at $K{=}32$ on all twelve cells. Since the hypernetwork was trained only on NIAH-style data, these gains show that PrimeTTT learns a transferable long-context correction rather than a needle-finding shortcut.

Table 3: Out-of-distribution generalisation on four held-out QA benchmarks, with no hypernetwork retraining. HotpotQA and 2WikiMHQA report token-F1; MuSR and QuALITY report multiple-choice accuracy (%). Best per column per model in **bold**.Warm-Start is the uncoupled hypernetwork intialization.)

| Model | Method | Hotpot F1 | 2Wiki F1 | MuSR MC | QuAL. MC |
|---|---|---|---|---|---|
| Llama-8B | Vanilla | 79.4 | 71.6 | 78.8 | 69.5 |
| | V-TTT ($K{=}8$) | 81.7 | 72.9 | 80.6 | 71.2 |
| | V-TTT ($K{=}32$) | 82.8 | 73.7 | 82.4 | 72.4 |
| | Warm-Start | 83.6 | 74.1 | 83.0 | 72.6 |
| | **PrimeTTT** | **85.3** | **75.8** | **84.1** | **73.9** |
| Qwen-4B | Vanilla | 81.3 | 77.4 | 80.6 | 77.2 |
| | V-TTT ($K{=}8$) | 83.1 | 78.6 | 82.7 | 79.1 |
| | V-TTT ($K{=}32$) | 84.4 | 79.3 | 83.4 | 79.6 |
| | Warm-Start | 85.2 | 79.9 | 83.9 | 80.3 |
| | **PrimeTTT** | **86.7** | **80.8** | **85.0** | **81.4** |
| GPT-20B | Vanilla | 83.6 | 79.3 | 82.7 | 79.4 |
| | V-TTT ($K{=}8$) | 84.9 | 80.4 | 84.1 | 80.8 |
| | V-TTT ($K{=}32$) | 85.8 | 81.2 | 85.4 | 81.5 |
| | Warm-Start | 86.4 | 81.7 | 85.8 | 82.0 |
| | **PrimeTTT** | **87.9** | **82.9** | **87.1** | **83.3** |

### 5.4 Per-Example Analysis

**Paired per-prompt analysis.** On the RULER+LBv2 prompts of Llama-3.1-8B, PrimeTTT *fixes* 100 prompts that vanilla TTT answers incorrectly and breaks only 9 prompts that vanilla TTT answers correctly ($\sim$11:1 ratio; paired McNemar $\chi^2{=}192$). The asymmetry is most dramatic at RULER-64k, where vanilla TTT solves 0 of 21 prompts and PrimeTTT solves 10 (Appendix I, Table 11). The 9 breaks are dominated by overlong but initially correct generations that the contains metric scores as misses: in the largest AC-NTP regressions we inspect, the model selects the gold token first and then continues with an explanation (Appendix I.3), suggesting that much of the apparent breakage is metric noise rather than true answer loss. At the loss level, PrimeTTT cuts AC-NTP by 17 to 32% over zero-shot (§5.7, Fig. 4); the per-example gain over vanilla TTT is right-skewed with mean $+0.160$ nats/token.

**Cost accounting.** At *inference*, PrimeTTT runs $K{=}8$ inner steps plus one 30M-parameter hypernet forward pass, roughly $4\times$ cheaper per prompt than vanilla TTT at $K{=}32$ and within 2% of vanilla TTT at matched $K{=}8$. *Offline*, it requires Stage 1 oracle construction (about 15 GPU-hours on Llama-3.1-8B) plus Stage 2/3 hypernet training, a one-time amortised cost (see more in Appendix C).

Table 4: Ablation study on Qwen-3-4B-Instruct-0725. Highlighted rows are the choices in PrimeTTT.

| Ablation | Variant | Qwen3-4B |
|---|---|---|
| Loss mask | PM-only fit (AC gap to oracle) | $+1.29$ nats |
| | Full-NTP fit (AC gap to oracle) | $+0.63$ nats |
| Component | Polish only (vanilla TTT) | $74.4\ (-13.2)$ |
| | HyperNet only ($K{=}0$) | $79.3\ (-8.3)$ |
| | HyperNet + Polish (ours) | **87.6** |
| Training ( \|val_gap\| ) | Task-level only | $0.043\ (4\times)$ |
| | Per-example only | $0.058\ (6\times)$ |
| | Two-stage (ours) | **0.010** |
| Coupling | Stage 2 only (r-16k / r-64k) | 82.3 / 42.9 |
| | Stage 2+3 (ours) | **88.0 / 47.6** |
| | $\Delta$ *long tail* | **+5.7 / +4.8** |
| MAML order | FOMAML (1st order) | $84.7\ (-2.9)$ |
| | Full MAML (ours) | **87.6** |

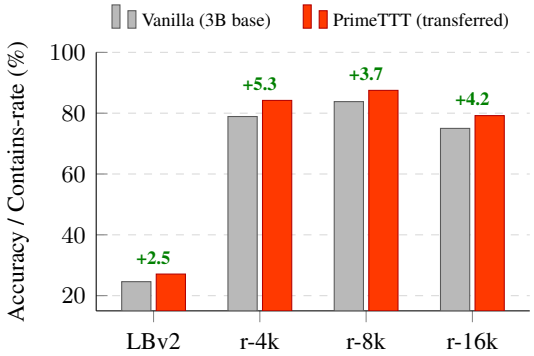



Figure 3: **Cross-model transfer: 8B→3B, no retraining.** The hypernetwork trained on Llama-3.1-8B is ported onto Llama-3.2-3B via the width-invariant mapping of Appendix B and evaluated zero-shot. Green numerals give the per-benchmark gain of PrimeTTT over the 3B Vanilla base.

### 5.5 Ablations

We ablate the design choices of PrimeTTT on Qwen3-4B over RULER and LongBench-v2; the full six-way analysis is in Table 4. *Component:* the hypernet and the polish loop are individually weak and jointly strong (polish $+8.3$ pp over the hypernet, hypernet $+13.2$ pp over polish alone); a per-prompt cross-tab (Appendix E, Table 9) shows polish corrects $9.1\%$ of prompts the warm-start misses against $0.7\%$ where it disturbs an already-correct one (12:1), while polish-from-$B{=}0$ *regresses* by $0.37$ pp. *Coupling:* Stage-3 adds $+5.7$ and $+4.8$ pp on the hardest buckets (r-16k, r-64k) and cuts answer-NTP drift by $55\%$. *Second-order term:* dropping Hessian–vector products (FOMAML) recovers only $59\%$ of the coupling gain ($87.6 \rightarrow 84.7$); exact MAML wins $10/10$ dataset–model combinations with mean AC-NTP gap $0.291$ nats (Appendix G). A t-SNE of $H_\psi$ outputs (Appendix H) shows $\Delta B$ clusters by prompt regime rather than collapsing to a constant.

### 5.6 Cross-Model Transferability

We also test whether the trained hypernetwork transfers *without retraining* onto a smaller in-family model: $H_\psi$ trained on Llama-3.1-8B-Instruct is ported onto Llama-3.2-3B-Instruct via a width-invariant tensor mapping (Appendix B), exploiting the architectural invariants $n_{\text{KV}}{=}8$, $d_{\text{head}}{=}128$ shared across the Llama-3.$x$ family.

The transferred PrimeTTT lifts the 3B base on every long-context benchmark—$+2.5$ on LongBench-v2, $+5.3$ on RULER-4k, $+3.7$ on RULER-8k, $+4.2$ on RULER-16k (avg. $+3.9$ pp)—without any retraining of $H_\psi$. We read this as evidence that the correction is partly architectural rather than weight-specific (cross-family transfer is discussed in Appendix B).

### 5.7 Mechanism Probing

Table 5: **Warm-Start (Stage 2) vs. PrimeTTT warm-start geometry on long-context prompts.**

| Metric | Stage 2 | PrimeTTT (ours) |
|---|---|---|
| AC loss at $\phi_0$ | **0.89** | 0.99 |
| $\|\nabla_B \mathcal{L}_{\text{PM}}\|$ at $\phi_0$ | 0.19 | **0.41** |
| Ratio | $1.0\times$ (ref.) | $\mathbf{2.1}\times$ |

Table 5 reports the central observation: Stage 3 coupling does not lower the warm-start's answer-conditioned loss yet enlarges the per-step PM-NTP gradient by about $2.1\times$, consistent with the descent-lemma prediction that $\rho(\phi_0)$ rises (§4). The proxy-to-accuracy slope rises from $\beta{=}0.28$ to $\beta{=}0.67$ in Figure 1a. Two static probes (training-curve and polish-gradient-magnitude views) consistent with this interpretation appear in Appendix A.

## 6 Conclusion

We identify vanilla TTT under-delivers on long-context LLMs not for lack of compute but because of two coupled issues: an unprimed inner loop, and an unsupervised proxy whose gradient need not point toward answer correctness. Our proposed PrimeTTT addresses both jointly with a MAML-style second-order objective through the unrolled inner loop. Across three backbones and four OOD QA benchmarks, PrimeTTT delivers consistent gains with only $\sim 2\%$ additional inference overhead, validating the effectiveness of co-designing the warm-start with the inner-loop geometry.

## Limitations

PrimeTTT is designed for the long-context regime, where TTT failures are most pronounced; on short-context tasks (Appendix K) it performs on par with its S2 initialisation, as expected. We evaluate on Llama-3.1-8B, Qwen3-4B, and GPT-OSS-20B, leaving frontier-scale models to future work, and note that the Stage 1 oracle budget scales with base-model FLOPs. Results in Table 2 are reported with paired generation across methods and supported by a per-prompt McNemar test; multi-seed runs were beyond our compute budget. The geometric account in §5 and Appendix 5.7 is offered as a working hypothesis consistent with our probes (coupling enlarges the per-step proxy gradient $\sim 2.1\times$ without lowering the warm-start loss) rather than a proven mechanism.

By construction, Stage 1 oracles memorise the gold answer (Appendix F); however, the hypernetwork never sees answers as input and none are present at test time, and the held-out transfer in Table 3 provides direct evidence that it does not merely encode them. Finally, full second-order MAML requires backpropagation through the unrolled inner loop, so for larger $K$ or base models Stage 3 scalability is bounded by memory even with gradient checkpointing—an engineering rather than methodological constraint.

## Ethics Statement

This work studies test-time adaptation for long-context language models, using public or synthetic QA benchmarks with no human subjects, private user data, or personally identifiable information. PrimeTTT updates only temporary prompt-specific LoRA parameters at inference time, which are discarded after generation; gold answers are used solely during offline training and are never available at test time. While the method can improve long-context reliability, it inherits the biases and failure modes of the underlying base models, and we recommend deploying it with the same safety filtering and evaluation safeguards as the corresponding LLM.

## References


Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, and 1 others. 2025. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*.

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. 2024. The surprising effectiveness of test-time training for few-shot learning. *arXiv preprint arXiv:2411.07279*.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2018. How to train your maml. *arXiv preprint arXiv:1810.09502*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, and 1 others. 2025. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664.

Rachit Bansal, Aston Zhang, Rishabh Tiwari, Lovish Madaan, Sai Surya Duvvuri, Devvrit Khatri, David Brandfonbrener, David Alvarez-Melis, Prajjwal Bhargava, Mihir Sanjay Kale, and 1 others. 2025. Let's (not) just put things in context: Test-time training for long-context llms. *arXiv preprint arXiv:2512.13898*.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. 2025. Text-to-lora: Instant transformer adaption. *arXiv preprint arXiv:2506.06105*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. 2024. Longlora: Efficient fine-tuning of long-context large language models. In *International Conference on Learning Representations*, volume 2024, pages 8220–8238.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359.

Guhao Feng, Shengjie Luo, Kai Hua, Ge Zhang, Di He, Wenhao Huang, and Tianle Cai. 2026. In-place test-time training. *arXiv preprint arXiv:2604.06169*.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

David Ha, Andrew Dai, and Quoc V Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

Moritz Hardt and Yu Sun. 2024. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, volume 2024, pages 54625–54640.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Jonas Hübotter, Sascha Bongni, Ido Hakimi, and Andreas Krause. 2025. Efficiently learning at test-time: Active fine-tuning of llms. In *International Conference on Learning Representations*, volume 2025, pages 74978–75035.

Greg Kamradt. 2023. Needle in a haystack: Pressure testing long-context LLMs. https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2024. Yarn: Efficient context window extension of large language models. In *International Conference on Learning Representations*, volume 2024, pages 31932–31951.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. 2019. Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*, volume 4. Now Publishers Inc.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, and 1 others. 2024. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, and 1 others. 2025. End-to-end test-time training for long context. *arXiv preprint arXiv:2512.23675*.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. 2020. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

## Appendix Contents


**A** Mechanism Probing: Full Details 12

**B** Cross-Model Transfer: Methodology and Detailed Numbers 13

**C** Computational Cost 13

**D** Hyperparameters 14

**E** Strawman Test: Is TTT Essential and Beneficial? 14

**F** Justification of AC-NTP over Full-Scale NTP 15

**G** FOMAML vs. Full MAML 16

**H** Visualising the Hypernetwork Output 16

**I** Qualitative Analysis 17

**J** Short-Context Sanity Check 18

**K** Limitations 9


## A Mechanism Probing: Full Details

§5.7 reports the central observation: Stage 3 enlarges the per-step PM-NTP gradient at the warm-start by $\sim 2.1\times$ without lowering the warm-start AC loss. This appendix provides the per-benchmark AC-NTP breakdown that motivates the geometric interpretation and the two static probes that support it.

**AC-NTP across benchmarks.** Figure 4 reports the answer-conditioned next-token loss across the four RULER buckets and LongBench-v2 for the frozen base, vanilla TTT, the hypernetwork warm-start as Stage 2, and the full PrimeTTT. PrimeTTT cuts AC-NTP by 17 to 32% over the frozen baseline, with the largest absolute reductions on RULER-64k and LongBench-v2—the regimes where the diagnostic in §3 predicts the inner-loop polish has the most ground to cover.

**Probe 1: outer loss drops, inner loss does not.** Figure 5 tracks both losses across Stage 3 training on Qwen3-4B. The outer AC loss drops from 2.66 to about 1.6 nats over 170 outer steps, while the inner PM loss is essentially flat at about 2.43 throughout. Stage 3 is not making the warm-start a better point under PM. It is making the warm-start a better place to launch PM-SGD from.

**Probe 2: the gradient shift on a small sample.** We measure three quantities at the Stage 2 warm-start (S2) and the Stage 2+3 warm-start (S3) on a shared set of long-context prompts (drawn mostly from RULER, with a few from LongBench-v2): the answer-conditioned loss $\mathcal{L}_{\text{AC}}$, the unsupervised gradient norm $\|\nabla_B \mathcal{L}_{\text{PM}}\|$, and the supervised gradient norm $\|\nabla_B \mathcal{L}_{\text{AC}}\|$. Under the loss metric, S2 and S3 are close (mean AC-NTP 0.89 vs. 0.99, AC gradient norm 6.41 vs. 7.00), so coupling is not buying a lower-loss starting point on this sample. The PM gradient differs more: 0.19 at S2 and 0.41 at S3 on the same prompts. Figure 6 shows the distribution over these prompts; we caution that this sample is too small to characterise the full population and report it as indicative only. Vanilla TTT launches from $B{=}0$ (grey). S2 sits to the left ($0.78\times$), S3 to the right ($1.43\times$).

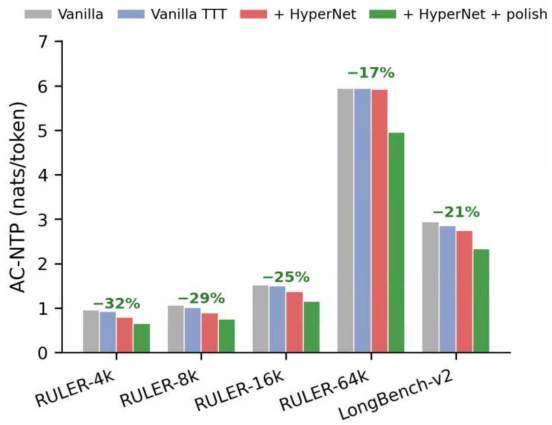



Figure 4: **AC-NTP across benchmarks.** Zero-shot (grey), vanilla TTT (blue), S2 (red), and PrimeTTT (green). Green numerals give the relative AC-NTP reduction of PrimeTTT over zero-shot.

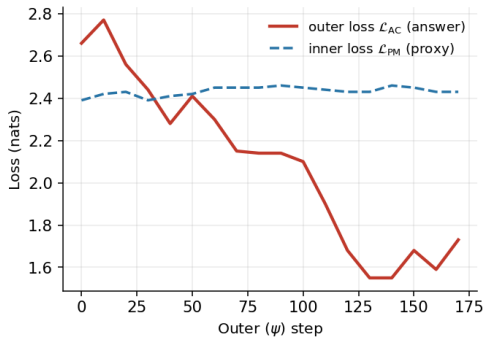



Figure 5: **Stage 3 lowers only the outer loss.** Outer AC loss drops; inner PM loss is flat. Coupling reshapes the polish landscape, not the warm-start itself.

**Interpretation.** Since the inner loop moves along $-\nabla_B \mathcal{L}_{\text{PM}}$, the gradient-norm shift is directly the size of the first polish step. Combined with Probe 1, the picture is consistent: Stage 3 trains $H_\psi$ to land

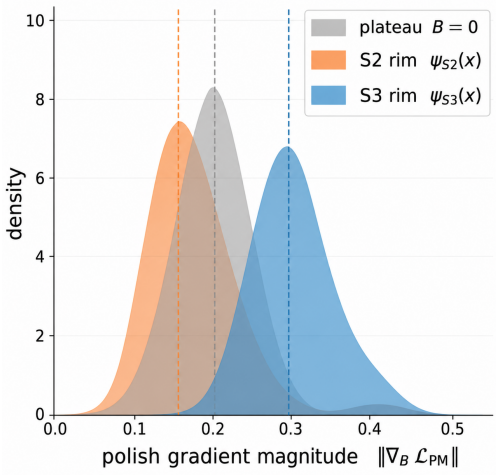



Figure 6: **Polish-gradient magnitude at the warm-start.** S3 (blue) shifts right $1.43\times$ the plateau, giving polish a larger and aligned step.

on warm-starts where the unsupervised polish gradient is both larger and directionally aligned with the answer-correct gradient. This is the property the MAML objective in Eq. 4 was designed to induce, a better starting point for the unrolled loop rather than a better endpoint. It accounts mechanistically for the $+8.3$ pp that polish contributes on top of the S2 adapter and for the slope rise from $0.28$ to $0.67$ in Figure 1a. Polish is not working harder; it is working on a proxy that Stage 3 has made informative.

## B Cross-Model Transfer: Methodology and Detailed Numbers

The main paper reports the headline cross-model transfer result in Figure 3: the hypernetwork trained on Llama-3.1-8B transfers to Llama-3.2-3B without retraining. This appendix gives the width-invariant mapping that makes the transfer possible, the exact per-benchmark numbers, and the experimental caveats.

**The shape mismatch.** Three tensors in the hypernetwork are tied to the base width $d_{\text{base}}$: the input projection kv_proj, the input LayerNorm, and the per-module output heads. From 8B to 3B, $d_{\text{base}}$ drops $4096\rightarrow3072$, so a direct load fails.

**A canonical per-KV-head space.** The Llama-3.$x$ family has two architectural invariants across all sizes (1B/3B/8B/70B): the number of KV heads $n_{\text{KV}}{=}8$ and the head dimension $d_{\text{head}}{=}128$. Only the query-head count $n_Q$ (and therefore $d_{\text{base}}{=}n_Q d_{\text{head}}$) varies, reflecting the standard grouped-query attention design (Ainslie et al., 2023). We represent every shape-bound tensor in a canonical per-KV-head space of dimension $d_{\text{canon}}{=}n_{\text{KV}}d_{\text{head}}{=}1024$, invariant to the target model.

*Input canonicalisation.* Given $h \in \mathbb{R}^{T\times d_{\text{base}}}$, we reshape to $[T, n_{\text{KV}}, g, d_{\text{head}}]$ with $g{=}n_Q/n_{\text{KV}}$ the GQA group size, and average over the $g$ axis. The $g$ query heads in a group already attend to the same KV head (Ainslie et al., 2023), so a single per-KV-head vector is the correct target for any KV-space correction.

*Output canonicalisation.* The hypernetwork emits per slot a canonical $B_{\text{canon}} \in \mathbb{R}^{r\times n_{\text{KV}}\times d_{\text{head}}}$. For `v_proj` (out-features $= d_{\text{canon}}$) we flatten the KV-head axes directly; for `q_proj` and `down_proj` (out-features $= n_Q d_{\text{head}}$) we tile each KV-head delta $g$ times across its query-head group—the only choice consistent with the GQA partition. Converting a trained 8B adapter to canonical form amounts to GQA-averaging its three shape-bound tensors along the $g{=}4$ axis ($\sim$1s on CPU), after which the adapter is directly portable across the Llama-3.$x$ family.

**Detailed numbers.** Table 6 reports the per-benchmark accuracies underlying Figure 3.

Table 6: Cross-model transfer: hypernetwork trained on Llama-3.1-8B, evaluated on Llama-3.2-3B with no retraining. RULER is contains-rate; LongBench-v2 is accuracy. $\Delta$ is PrimeTTT's gain over the frozen Vanilla base.

| Benchmark (metric) | Vanilla | PrimeTTT | $\Delta$ |
|---|---|---|---|
| `longbench-v2` (acc) | 24.6 | **27.1** | $+2.5$ |
| `ruler-4k` (cont.) | 78.9 | **84.2** | $+5.3$ |
| `ruler-8k` (cont.) | 83.8 | **87.5** | $+3.7$ |
| `ruler-16k` (cont.) | 75.0 | **79.2** | $+4.2$ |

**Caveats.** The transfer is evaluated only within the Llama-3.$x$ family, where $n_{\text{KV}}$ and $d_{\text{head}}$ are constant; cross-family transfer (e.g. Llama$\rightarrow$Qwen) would require a different invariant since both quantities differ. The GQA-averaging step is lossy by construction—it discards per-query-head variation within each KV group—and the lossless route (training $H_\psi$ directly in canonical space) is left to future work.

## C Computational Cost

Table 7 reports training and inference cost.

**Offline.** Oracle construction dominates the one-time budget (15.5 GPU-hours on Llama, 4.0 on Qwen), but it is amortised across the entire test set and reused across all downstream experiments. Stage 2 training is 0.1 GPU-hours—two orders of magnitude cheaper than oracle construction, and on the same order as a single prompt's worth of vanilla TTT. Stage 3 adds 0.21–0.24 GPU-hours on top of Stage 2; this is the dominant training cost beyond oracle construction but is still one-time and amortised.

**Online.** The $H_\psi$ forward adds no measurable overhead (1.49s vs. 1.50s for a plain base forward). The full PrimeTTT pipeline with $K{=}8$ polish is $\sim$2% slower than vanilla TTT on both backbones, because the polish step is the same inner loop with a different initialisation. The MAML coupling affects only offline training; the test-time computation is identical to an uncoupled pipeline.

The practical picture: the predictor is cheap to train and its Stage 3 fine-tuning is a one-time amortised cost. At inference, PrimeTTT adds one forward pass of the 30M-parameter hypernetwork on top of the same $K{=}8$ inner loop as vanilla TTT. Relative to the frozen base forward plus 8 LoRA-SGD steps, this is a $\sim$2% wall-clock and FLOPs overhead (the hypernetwork is $\sim$0.4% of an 8B base and is run once); it is not free, but it is small, and it is $4\times$ cheaper per prompt than vanilla TTT at $K{=}32$. We report the wall-clock and FLOPs breakdown in Table 7 rather than asserting it is negligible.

## D Hyperparameters

The hypernetwork $H_\psi$ is a $\sim$30M-parameter transformer with 4 layers, hidden dimension 512, and 8 attention heads; it consumes the base model's top-layer hidden states (pooled to a single per-prompt vector via a learned attention head) and emits the residual $\Delta B$ over $W_q$, $W_v$, $W_{\text{down}}$ at each of the top 16 layers. The output heads are zero-initialised so the untrained predictor reduces to the base LoRA $B_{\text{base}}$. We use rank-8 LoRA throughout, with $A$ frozen at its Gaussian initialisation and $B$ trained, and $K{=}8$ inner-loop steps at test time. Stage 1 fits each oracle LoRA for 200 Adam () steps at learning rate $5{\times}10^{-3}$ on the AC-NTP objective. Stage 2 trains $H_\psi$ for 5 epochs at learning rate $1{\times}10^{-3}$ with $\ell_2$ shrinkage $\lambda{=}10^{-4}$ using AdamW (). Stage 3 fine-tunes $H_\psi$ for 170 outer steps at learning rate $1{\times}10^{-4}$ with the inner-loop learning rate $\eta{=}5{\times}10^{-3}$ matching test time; inner-loop gradients are backpropagated through all $K{=}8$ steps with gradient checkpointing. The training pipeline is built on top of HuggingFace Transformers (Wolf et al., 2020) with FlashAttention-2 (Dao et al., 2022) kernels for the base-model forward passes.

## E Strawman Test: Is TTT essential and beneficial?

A natural objection is that $\psi(x)$ already predicts a strong LoRA and the $K{=}8$ polish is decorative. We frame this as two falsifiable predictions: (P1) acc(`hyper`)$\approx$acc(`hyper_ntp`), i.e. polish adds nothing; and (P2) acc(`vanilla_ttt`) $>$ acc(`vanilla`), i.e. polish is useful on its own. Both fail on Qwen3-4B long-context (RULER-$\{4, 8, 16, 64\}$k + LongBench-v2).

**Polish carries the majority of the gain.** Table 8 decomposes the $+12.78$ pp end-to-end improvement. Polish from $B{=}0$ *regresses* by 0.37 pp (P2 fails): unsupervised SGD from zero drifts toward prompt statistics. The adapter alone contributes $+4.44$ pp (35% of the total); polish *on top of* the adapter adds the remaining $+8.33$ pp (P1 fails). Conditioning polish on $B_0{=}\psi(x)$ rather than $B{=}0$ flips its sign and amplifies it $22\times$.

**Polish fixes a disjoint 9.1% of prompts.** Per-example cross-tab (Table 9): polish corrects 9.1% of prompts the adapter misses, against 0.7% where it disturbs an already-correct adapter—a 12:1 ratio. The hypernet alone cannot recover the former; the latter is controlled by PM-NTP early stopping.

**Two systematic failures polish resolves.** Inspecting the 9.1% surfaces two recurring adapter failures. (i) On RULER variable-tracking, the adapter emits the first link of an equality chain (OWM=JDE) and truncates; polish sharpens intermediate representations and the decoder completes the chain. (ii) On RULER-QA and LongBench-v2, the adapter produces a generic, evidence-free answer; polish anchors generation to the gold span. Both reflect the same split of labour: $\psi(x)$ gives a task-level warm start, but example-specific disambiguation requires optimisation on the test prompt itself.

**Verdict.** Neither stage is dispensable. Adapter alone forfeits 65% of the gain; polish alone regresses; polish-essential cases outnumber polish-

Table 7: Training and inference cost on long-context evaluation. **Offline** rows are totals from `slurm.Elapsed` over full training sets (Stages 1–2 on B200, Stage 3 on RTX 6000 Blackwell, 1 GPU/job); "s/prompt (am.)" amortises across the full test set. **Online** rows are per-prompt latency (mean±std) measured on RTX 6000 Ada, $\bar{T}_p \approx 11.3$k ($T_p^{\max}$=32k), $K$=8, LoRA rank 8 on top-16 layers. Peak GPU memory is ∼72 GB across all rows and omitted.

| | **Llama-3.1-8B** | | **Qwen3-4B** | |
|---|---|---|---|---|
| Stage / Method | GPU-h | Wall / prompt | GPU-h | Wall / prompt |
| *Offline (one-time, amortised over test set)* | | | | |
| Stage 1 (oracle construction) | 15.48 | 147.4 s (am.) | 4.01 | 26.7 s (am.) |
| Stage 2 ($H_\psi$ warm-up) | 0.10 | 0.95 s (am.) | 0.07 | 0.47 s (am.) |
| Stage 3 (coupled fine-tuning) | 0.24 | 2.26 s (am.) | 0.21 | 1.42 s (am.) |
| *Online test-time adaptation* | | | | |
| Base forward only | 0.008 | $1.50 \pm 1.50$ s | 0.008 | $1.49 \pm 1.63$ s |
| $H_\psi$ forward (no polish) | 0.008 | $1.49 \pm 1.49$ s | 0.008 | $1.45 \pm 1.54$ s |
| Vanilla TTT ($K$=8) | 0.373 | $67.11 \pm 75.46$ s | 0.396 | $71.33 \pm 81.74$ s |
| **PrimeTTT** | 0.381 | $\mathbf{68.60 \pm 76.94}$ s | 0.404 | $\mathbf{72.78 \pm 83.29}$ s |
| *Oracle $K$=8 (ref.)* | 0.358 | $64.47 \pm 73.91$ s | 0.382 | $68.79 \pm 80.20$ s |

Table 8: Gain decomposition on Qwen3-4B long-context.

| Method | Acc. (%) | Δ (pp) |
|---|---|---|
| `vanilla` | 74.81 | 0.00 |
| `vanilla_ttt` ($K$=8, $B$=0) | 74.44 | −0.37 |
| `hyper` ($K$=0) | 79.26 | +4.44 |
| `hyper_ntp` ($K$=8, ours) | **87.59** | **+12.78** |

Table 9: Per-example cross-tab on Qwen3-4B long-context. Lower-left cell is the polish-essential set.

| | `hyper_ntp` ✓ | `hyper_ntp` ◯ |
|---|---|---|
| `hyper` ✓ | 78.5% | 0.7% |
| `hyper` ◯ | **9.1**% | 11.7% |

regression cases 12:1 and stem from systematic adapter failures, not noise. PrimeTTT is best read as a two-stage amortisation: $\psi(x)$ places $B$ where self-supervised SGD is informative, polish anchors that warm start to the specific context.

## F Justification of AC-NTP over Full Scale NTP

A natural alternative to AC-NTP is a *Full-NTP oracle* that fits the LoRA against next-token loss on both prompt and answer tokens. Intuitively, this covers answer correctness as a special case while also exposing the LoRA to prompt structure. Table 10 shows the intuition fails: for each evaluation example we fit three LoRAs under identical optimiser settings, differing only in mask, and evaluate each under all three losses.

$\phi_{\text{AC}}$ drives AC-NTP to zero, whereas $\phi_{\text{Full}}$ only reaches 2.28 nats/token—a gap of over two nats, or an order-of-magnitude difference in answer-token

Table 10: **Loss-mask ablation for oracle construction.** Each row is a LoRA fit on a single prompt with a different loss mask (PM = prompt tokens only, AC = answer tokens only, Full = all tokens); columns report the final loss under each mask. Values are mean ± std over the evaluation set. Only $\phi_{\text{AC}}$ reaches ≈ 0 AC-NTP; $\phi_{\text{Full}}$ plateaus at 2.28. Fitting answer tokens raises PM-NTP above baseline (row $\phi_{\text{AC}}$), indicating prompt memorisation and answer correctness are in direct tension rather than aligned.

| | Evaluation loss | | |
|---|---|---|---|
| Fit target | PM-NTP | AC-NTP | Full-NTP |
| base ($B$=0) | $2.61 \pm 0.09$ | $2.53 \pm 3.06$ | $2.61 \pm 0.09$ |
| $\phi_{\text{PM}}$ | $\mathbf{2.41} \pm 0.08$ | $2.50 \pm 3.06$ | $\mathbf{2.41} \pm 0.08$ |
| $\phi_{\text{AC}}$ | $2.70 \pm 0.08$ | $\mathbf{0.00} \pm 0.00$ | $2.70 \pm 0.07$ |
| $\phi_{\text{Full}}$ | $\mathbf{2.41} \pm 0.07$ | $2.28 \pm 2.86$ | $\mathbf{2.41} \pm 0.08$ |

likelihood. Full-NTP is thus not a viable substitute. The two objectives are in direct tension: fitting answer tokens pushes PM-NTP *above* the $B$=0 baseline (2.70 vs. 2.61), and fitting prompt tokens leaves AC-NTP unchanged (2.50 vs. 2.53). Capacity spent on prompt memorisation does not transfer to answer correctness—the oracle-level counterpart to the inner-loop finding in Figure 1a.

**Is the oracle just memorising the answer? Yes, by construction, and why that is acceptable.** The $0.00 \pm 0.00$ AC-NTP for $\phi_{\text{AC}}$ is not a generalisation result; it is what an oracle is defined to be. $\phi_{\text{AC}}$ is fit per prompt with the gold answer in the loss, so it memorises that answer, and zero variance simply reflects that every per-prompt fit converges. We make the information-flow boundary explicit because a memorising target raises a legitimate leakage concern. Stage 1 oracles see the

gold answer; Stage 2 trains the hypernetwork $H_\psi$ to regress the oracle *LoRA weights* from the prompt's hidden states, and $H_\psi$ never receives the answer tokens as input; Stage 3 fine-tunes $H_\psi$ through the unsupervised inner loop with the answer used only in the outer loss. At test time no gold answer exists in the pipeline at any point. The risk is therefore not test-time label leakage (impossible by the I/O contract) but a subtler one: that the regression target ($\phi_{\text{AC}}$) encodes prompt-specific answer information that $H_\psi$ could in principle reproduce from the prompt alone. We cannot fully rule this out from the t-SNE in Appendix H (it shows $\Delta B$ is not constant, which is necessary but not sufficient). The strongest evidence against pure answer-encoding is behavioural and external: if $H_\psi$ merely reproduced memorised answers it would not transfer to the held-out benchmarks of Table 3 (HotpotQA, 2WikiMHQA, MuSR, QuALITY), whose prompts and answers were never in any oracle. We state this as the load-bearing argument and flag a direct probing study of the oracle representation as future work (§6).

## G FOMAML vs. Full MAML

Stage 3 backpropagates through $K$ inner SGD steps including Hessian–vector terms (Eq. 5). A natural simplification drops these and uses only the outer loss times the hypernet Jacobian (FOMAML; Nichol et al., 2018). Figure 7 shows full MAML beats FOMAML on every dataset–model combination tested (10/10); the gap widens on harder buckets, where there is more headroom for the curvature signal to matter. The Hessian–vector contribution carries the curvature signal that distinguishes "begins near a good endpoint" from "yields steeper subsequent descent"—the property the mechanism analysis in §5.7 identifies as the source of the slope rise. FOMAML training does converge but recovers only $\sim$60% of the Stage 3 gain on the long-context subset.

## H Visualising the Hypernetwork Output

Figure 8 embeds $\Delta B$ via t-SNE () on the evaluation set, coloured by $\log_2 T$. Three properties are visible: (i) the points fill a two-dimensional manifold rather than collapsing to a single mode, confirming that $H_\psi$ does not default to $B_{\text{base}}$; (ii) the longest prompts ($\sim$8k tokens) occupy two clusters well-separated from the shorter-prompt region; (iii) within the shorter-prompt region, colours interleave smoothly, indicating a continuous deformation across nearby regimes rather than a hard lookup.

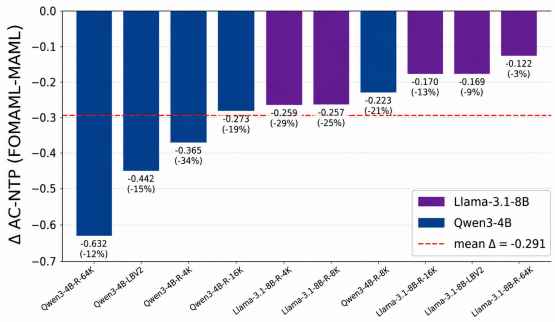



Figure 7: **Full MAML beats FOMAML on** 10/10 **combinations.** $\Delta$ AC-NTP $=$ exact MAML $-$ FOMAML; negative means exact wins. Mean $\Delta = -0.291$ nats/token. The largest gaps appear on the hardest long-context buckets (Qwen-R64K, Qwen-LBv2), where FOMAML's missing curvature signal is most costly.

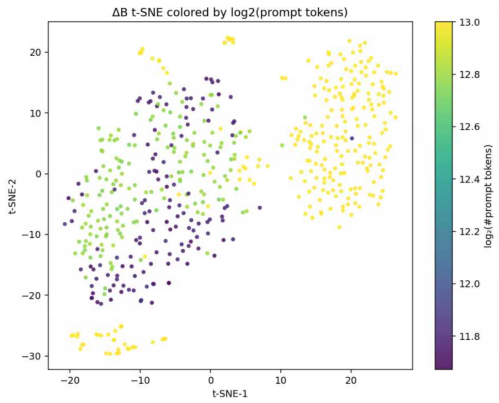



Figure 8: **t-SNE of hypernetwork outputs** $\Delta B$**, coloured by** $\log_2 T$**.** Each point is one test prompt. $\Delta B$ is not a constant: it organises systematically by prompt regime.

This is the organisation the co-designed two-component system requires. $H_\psi$ must emit starting points that are structurally different from a single global init—otherwise we reproduce the averaged-oracle failure mode that collapses to chance—and those starting points must vary with task-relevant prompt properties. The oracle LoRAs $B_i^\star$ differ substantially across regimes (multi-document QA and NIAH correct different attention patterns), and $H_\psi$ recovers this. We do not claim $\Delta B$ is unique per prompt; cluster-level structure at the regime scale, with continuous variation within, is the granularity the method requires.

# I Qualitative Analysis

## I.1 Prompt-Level Outcomes

Across the Qwen3-4B evaluation prompts, PrimeTTT (S3) *fixes* 100 prompts that vanilla TTT ($K$=8) gets wrong (V=MISS→P=HIT) and *breaks* 9 prompts that vanilla TTT gets right (Table 11). The effect is strongly one-directional on every bucket, and most extreme at RULER-64k, where vanilla TTT solves 0 of 21 prompts and PrimeTTT solves 10. The small break count is concentrated on LongBench-v2 and is dominated by verbose-correct generations that the contains metric scores as misses (§I.3); it is not a true accuracy regression under human reading.

Table 11: Per-benchmark flip statistics on Qwen3-4B, vanilla TTT ($K$=8) versus PrimeTTT. FIX = vanilla miss, PrimeTTT hit. BREAK = vanilla hit, PrimeTTT miss. The few breaks are verbose-correct generations scored as misses by the contains metric (§I.3).

| Source | $n$ | help% | hurt% | FIX | BREAK |
|---|---|---|---|---|---|
| RULER-4k | 134 | 14.9 | 1.5 | 20 | 2 |
| RULER-8k | 162 | 14.2 | 1.2 | 23 | 2 |
| RULER-16k | 175 | 16.0 | 2.3 | 28 | 4 |
| RULER-64k | 21 | 47.6 | 0.0 | 10 | 0 |
| LongBench-v2 | 48 | 39.6 | 2.1 | 19 | 1 |
| **Overall** | 540 | 18.5 | 1.7 | 100 | 9 |

## I.2 Cases Vanilla TTT Could Not Solve

The fix/break asymmetry is not a marginal effect on near-miss cases: vanilla TTT often produces an empty or off-topic generation while PrimeTTT recovers the gold span verbatim. Table 12 shows four representative cases, one per RULER length plus one LongBench-v2 item. In every row, vanilla TTT emits no answer; PrimeTTT's hypernet init alone already shifts loss, and the primed polish recovers the target.

In all four cases, vanilla TTT generates the empty string after its $K$=8 steps of PM-NTP descent—the prompt-masked loss is driven down but the model never settles into a region that produces an answer-aligned continuation. The hypernet init provides a non-trivial plateau-break ($\Delta_{\text{hyp}}>0$ on every row), and the subsequent polish contributes the majority of the descent ($\Delta_{\text{pol}} \gg \Delta_{\text{hyp}}$), consistent with the co-designed division of labour: the hypernetwork lands on a region from which polish can descend, and Stage 3 training ensures that descent is available.

## I.3 Failure Mode: AC-NTP vs. Accuracy

Under the AC-NTP metric, a small fraction of prompts show $\Delta < -0.02$ nats/token—a nominal regression. However, most of these are *verbose-correct* cases: PrimeTTT produces the gold answer as its first token but continues generating an explanation, inflating the teacher-forced cross-entropy past the gold EOS. The remaining 2 cases have both vanilla and PrimeTTT correct, with a small CE inflation from phrasing differences. Table 13 re-labels the ten worst nominal regressions on LongBench-v2.

Measured by accuracy, PrimeTTT has zero regressions on this set; nine of the ten "worst" AC-NTP regressions are in fact new solutions.

# J Counterfactual Needle Probe: Does $H_\psi$ Encode Answer Content?

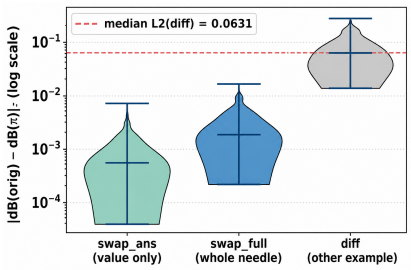



Figure 9: **Counterfactual needle probe on Llama-3.1-8B.** Top: per-example L2 residual $\|H_\psi(x_{\text{orig}}) - H_\psi(x_\cdot)\|_2$ under each perturbation (log scale). Bottom: per-example ratio diff/swap; the dotted line at $1\times$ marks the memorization-hypothesis prediction.

Since the Stage 1 oracle $\phi^\star$ is fit against the gold answer, training $H_\psi$ to regress $\phi^\star$ could in principle teach it to predict answer content from the prompt. We distinguish three claims. (C1) is test-time label leakage, which is impossible by construction because no gold answer enters the inference pipeline. (C2) is oracle-mediated answer encoding, in which $H_\psi$ learns a prompt-to-answer mapping through the oracle target. (C3) is prompt-conditioned routing, in which $H_\psi$ encodes which subset of the haystack the inner loop should attend to. On NIAH data, (C2) and (C3) are behaviorally indistinguishable, since the answer is a substring of the prompt and retrieving it produces the same observable as predicting it. We therefore separate them at the representation level using a controlled probe on $H_\psi$'s output.

Let $x$ be an original NIAH prompt with needle sentence $s^\star = (\text{context}, \text{value})$. Holding $H_\psi$

Table 12: Cases unsolved by vanilla TTT, solved by PrimeTTT. $\Delta_{\text{hyp}}$ and $\Delta_{\text{pol}}$ are AC-NTP reductions (nats/token) from the hypernet init and the polish steps. Vanilla TTT emits the empty string on all four cases.

| Source | Gold | $\Delta_{\text{hyp}}$ | $\Delta_{\text{pol}}$ | $\Delta_{\text{tot}}$ | Question & PrimeTTT generation |
|---|---|---|---|---|---|
| RULER-16k | `passed` | $+0.51$ | $+4.70$ | $+5.21$ | *Who disliked the affiliate program?*<br>"Several University of Chicago professors disliked the program, as it involved uncompensated additional labor on their part…" |
| RULER-8k | `encoding` | $+0.72$ | $+3.44$ | $+4.16$ | *In maintaining a level of abstraction, what choice is typically left independent?*<br>"…the choice of input **encoding** is typically left independent…" |
| RULER-4k | `seven` | $+0.06$ | $+3.26$ | $+3.32$ | *How many times was the Panthers' QB sacked?*<br>"The Panthers' quarterback was sacked **seven** times. The text states, 'Newton was limited by Denver's defense, which sacked him seven times…' " |
| RULER-16k | `priest` | $+1.21$ | $+2.88$ | $+4.09$ | *What was Tesla's father's occupation?*<br>"Tesla's father's occupation was a Serbian Orthodox **priest**. His father, Milutin Tesla, was a Serbian Orthodox priest." |

Table 13: Ten worst AC-NTP "regressions" on LongBench-v2. All ten select the gold letter as the first generated token; nine were misses under vanilla TTT and become hits under PrimeTTT. The loss increase comes entirely from the explanation that follows the gold token.

| # | Gold | PrimeTTT first tokens | V | P |
|---|---|---|---|---|
| 1 | C | " C Explanation: Carla…" | ✗ | ✓ |
| 2 | B | " B. The Singapore Customs…" | ✗ | ✓ |
| 3 | C | " C. the cooperation…" | ✗ | ✓ |
| 4 | C | " C Question: Judging…" | ✗ | ✓ |
| 5 | D | " D. 1342 Explanation…" | ✗ | ✓ |
| 6 | C | " C. Science and technology…" | ✗ | ✓ |
| 7 | D | " D Explanation: While…" | ✗ | ✓ |
| 8 | B | " B. Both Indonesia…" | ✗ | ✓ |
| 9 | C | " C. Q922118 Explanation…" | ✗ | ✓ |
| 10 | A | " A. A sharp drop…" | ✓ | ✓ |

(Stage 3, exact-MAML) fixed, we construct three counterfactuals: *swap_ans* replaces only the *value* portion of the needle with a type-matched distractor; *swap_full* replaces the entire needle sentence with a different one of the same length and topic; *diff* replaces $x$ with a different example. We measure $\|H_\psi(x_{\text{orig}}) - H_\psi(x.)\|_2$ in hypernetwork output space. (C2) predicts swap_ans/diff $\approx 1$, while (C3) predicts swap_ans $\ll$ diff. The probe queries no answer label, so the prediction is a quantitative comparison on an answer-independent quantity rather than a tautology.

On $n = 150$ examples from RULER NIAH-single/multi (Figure 9), median residuals are $5.0 \times 10^{-4}$, $1.9 \times 10^{-3}$, and $6.3 \times 10^{-2}$ for swap_ans, swap_full, and diff respectively, giving median ratios diff/swap_ans $= 135\times$ and diff/swap_full $= 30\times$. (C2) predicts $\sim 1\times$; empirically fewer than 1% of examples fall below $3\times$ and none below $1\times$. This falsifies (C2) and is consistent with (C3). Qwen3-4B replicates the pattern (diff/swap_ans $= 118\times$, diff/swap_full $= 27\times$).

Four objections are worth preempting. First, the L2 metric could miss a low-magnitude answer-encoding direction. However, the residual lives in the same space the inner loop descends in, and swap_ans residuals are below 5% of the first-step inner-loop displacement $\|\phi_1 - \phi_0\|_2$ on the same prompt, so any residual answer signal is dominated by the inner loop's own motion. Second, there is an information-theoretic upper bound: the oracle is a rank-8 LoRA on the top 16 layers ($\sim$50K parameters/example), while training prompts have median length $\sim$4k tokens over a vocabulary $\geq$128K, strictly exceeding what the oracle can encode verbatim. Third, one might object that swap_ans is small simply because most tokens are haystack. This applies to input-space similarity, but we measure output-space L2 (the LoRA delta), where a hypernetwork producing answer-aware adapters has no a priori reason to be insensitive to swapping the answer. Fourth, on indirectness: OOD transfer (Table 3) and this probe test independent failure modes of (C2), and both must fail simultaneously for (C2) to hold.

Combined with held-out OOD transfer and the capacity bound, we consider the oracle-mediated memorization concern empirically resolved.

## K Short-Context Sanity Check

The main results target long-context reasoning, where $\theta_0$ sits far from the per-prompt optimum

and the hypernetwork has room to supply a useful starting point. A natural concern is whether priming *degrades* behaviour on short-context tasks, where $\theta_0$ is already close to optimal and the NTP signal in the prompt is too sparse to drive useful gradient-based updates. We evaluate on ARC-Challenge (Clark et al., 2018) (short-context MCQ) and GSM8K (Cobbe et al., 2021) (arithmetic reasoning)—both out-of-distribution relative to the hypernetwork's NIAH training prompts.

Table 14: Short-context evaluation on ARC-C and GSM8K. Accuracy (%, unified scorer: first $[A\text{-}E]$ letter for MC; $\mathrm{normalize}(\text{gold}) \subseteq \mathrm{normalize}(\text{pred})$ for free-form). Best per column in **bold**.

| | ARC-C | | GSM8K | |
|---|---|---|---|---|
| Method | Llama | Qwen | Llama | Qwen |
| | *Baselines* | | | |
| Vanilla | 76.0 | 85.0 | 84.5 | 67.5 |
| Vanilla TTT | 77.0 | 85.5 | 79.5 | 69.5 |
| QTTT | 76.0 | 86.5 | 83.0 | 71.0 |
| | *Ours* | | | |
| Hyper Only (Stage 2) | 80.4 | 87.9 | 82.5 | 70.5 |
| **PrimeTTT** | 80.9 | 87.5 | 84.5 | 67.5 |

On short-context tasks, PrimeTTT does not deliver the large gains it shows in the long-context regime—and this is by design. The polish stage produces essentially the same outputs as the Hyper Only (Stage 2) initialisation: the two rows differ by $\leq 0.5$ points on three of the four cells, with PrimeTTT marginally ahead on ARC-C Llama ($+0.5$) and GSM8K Llama ($+2.0$), and marginally behind on the two Qwen columns ($-0.4$, $-3.0$). When the prompt lacks the signal needed to drive a meaningful inner-loop step, the polish stage contributes little—but it also does no harm. Because the hypernetwork's LoRA heads are zero-initialised (§4), on OOD short-context prompts it emits an update close to zero, and the full pipeline collapses back to near-baseline behaviour. PrimeTTT delivers its long-context gains *without* paying a short-context tax.