HETEROTELIC LEARNING: ONE TRAINING LOSS, MANY LEARNING ENDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Machine learning usually ties what a model learns to the loss optimized during training. Yet learning systems may need to adapt from simple and widely available objectives while serving purposes that are not directly expressed by those objectives, which is what current learning models cannot easily do. To alleviate this limitation, we propose Heterotelic Learning, a paradigm in which a mechanism learned in advance shapes updates from a shared online objective to serve different intended learning ends. We instantiate this idea with Learned Objective Transformation (LOT), which learns a temporary compiler from supervision specifying desired behaviors or target gradients. During adaptation, the compiler shapes the updates induced by a shared online objective and is then removed, leaving only the resulting parameter changes. Using the next-token prediction (NTP) as the shared objective in language modeling, we provide theoretical and empirical evidence that the same NTP objective can support substantially different adaptation effects, including changes in update direction, regularization, answer-oriented adaptation, and learning over longer streams. Evaluations on unseen contexts and multiple language-model backbones demonstrate that the proposed LOT can effectively serve different learning ends via a shared objective.

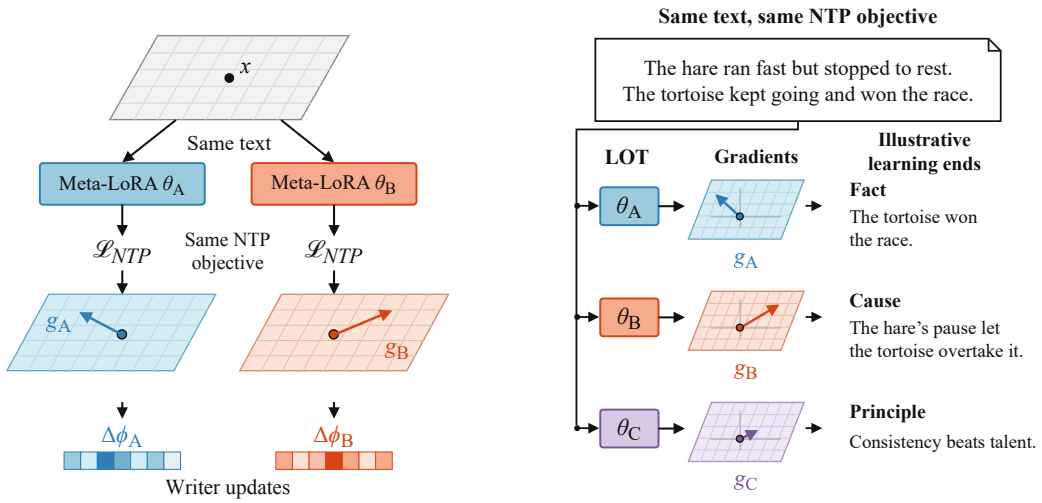
1 INTRODUCTION

Machine learning is usually organized around explicit training objectives: we choose a loss, optimize it, and expect the resulting updates to improve the behavior that the loss specified. Human learning is often less tightly coupled to an immediate objective, with the same experience later supporting recall, reasoning, adaptation, or other purposes not specified when it was acquired (Reber, 1967; Schapiro et al., 2013). Philosophy distinguishes autotelic activities, whose ends are internal to the activities themselves, from heterotelic activities, which are undertaken in service of separable ends (Mourelatos, 1993; Chrisman & Hubbs, 2025). Borrowing this means–end separation, we ask a question for machine learning: must the online objective used to compute parameter updates only specify the desired outcome of adaptation? Here, the online objective denotes the criterion optimized during adaptation to newly observed experience. To study this question, we introduce **Heterotelic Learning** as a framework for separating the online objective from the learning end it serves.

Pretraining supports many downstream tasks with one objective, while meta-learning can improve the objective itself over different learning tasks (Devlin et al., 2019; Bechtle et al., 2021; Baik et al., 2021). Heterotelic Learning studies how a mechanism learned in advance can shape parameter updates under a shared online loss rule to serve a chosen learning end. Given the same observed online input and loss rule, different learned mechanisms can induce different parameter updates.

This perspective matters most for models that learn from ongoing documents, interactions, or their own experience. Such experience often provides abundant self-supervision but little direct guidance about how the model should change in response. Language modeling makes this gap particularly clear. Nearly any text provides next-token prediction (NTP) targets, yet the value of learning from that text may only become apparent later. A useful update may help answer future questions, preserve unrelated knowledge, or avoid unnecessary change when similar information is encountered again. These goals can depend on information that is unavailable when the update is made. Heterotelic Learning addresses this gap by learning in advance how updates from a shared online objective should change the model.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Same NTP objective, different updates (b) Different learning ends

Figure 1: **Overview of Heterotelic Learning.** LOT compilers induce different gradients from the same text and NTP objective, enabling distinct possible learning ends after the compiler is removed.

We study this idea via Learned Objective Transformation (LOT), a concrete realization of Heterotelic Learning. As illustrated in Figure 1, LOT uses a temporary compiler to shape the update produced by the shared online objective, while a persistent writer stores the resulting parameter change. During offline meta-training, LOT uses a bilevel objective (Finn et al., 2017; Franceschi et al., 2018) in which the writer is updated using NTP, while the compiler is trained according to how well the resulting update matches a target gradient or produces a desired behavior. During later adaptation, the compiler is frozen: the model observes only the text and its ordinary next-token targets, updates the writer, and then removes the compiler before answering. We study several compiler parameterizations, ranging from additive adapters to a learned transformation inserted between the LoRA factors, and analyze how they reshape the NTP gradient and when they can reproduce a desired update.

Our experiments evaluate both the update itself and the behavior retained after the compiler is removed. LOT can substantially change the magnitude and direction of NTP updates, including transformations that cannot be reproduced by learning-rate changes alone. When meta-trained using question-answer supervision in the outer loop, it makes later passage-only NTP updates improve answer likelihood on unseen contexts and across multiple language-model backbones. The same framework also improves generated answers after the compiler is removed, and compilers trained over short update sequences retain gains over Direct NTP on substantially longer streams. LOT trains the compiler to shape future adaptation, with passage NTP supplying the online training objective. The resulting update rule determines how the model uses new text: the same supervision can lead to different changes in its parameters and behavior.

2 Related Work

Learning from new experience. Models can incorporate newly observed information through parameter updates. Test-time training uses self-supervised objectives to adapt models to new inputs (Sun et al., 2020). In language modeling, dynamic evaluation updates model parameters on recent text to improve subsequent predictions (Krause et al., 2018). Document-adaptation methods study how learning from new text supports later questions about its content. CaMeLS meta-learns token weights for language-modeling updates to improve subsequent question answering (Hu et al., 2023). PERK uses meta-training to learn how to encode new contexts into low-rank adapters and use the adapted model for reasoning (Chen et al., 2026). Model editing makes targeted changes from explicit facts or input–output specifications (Mitchell et al., 2022; Meng et al., 2022). Continual-learning methods manage sequential adaptation through replay, regularization, or gradient constraints (Jang et al., 2022; Lopez-Paz & Ranzato, 2017).

Learning how to update. Meta-learning can shape adaptation through several components of the learning process. MAML learns an initialization that supports rapid adaptation (Finn et al., 2017). Learned optimizers and meta-learned preconditioners learn how gradients are converted into parameter updates (Andrychowicz et al., 2016; Li et al., 2017; Flennerhag et al., 2020; Park & Oliva, 2019). Learned-loss methods learn the objective used during adaptation (Bechtle et al., 2021; Baik et al., 2021). Model parameterization also affects the update: linear adapter reparameterizations can induce gradient transformations (Hennigen et al., 2025). LOT uses meta-training to learn a temporary compiler that shapes the NTP gradient applied to a separate writer. It leaves the online supervision and NTP loss unchanged. After adaptation, the compiler is removed, and the writer retains the resulting parameter changes.

Relationship to established learning paradigms. Different learning paradigms emphasize different aspects of how models acquire and use knowledge. Transfer, multi-task, and continual learning study how knowledge is shared across tasks, domains, or sequences of experience (Pan & Yang, 2010; Zhang & Yang, 2022; Wang et al., 2024). Meta-learning studies how the adaptation process itself can be learned (Hospedales et al., 2022). Heterotelic Learning focuses on how the same online supervision can serve different learning ends. A learning end can specify what information to acquire, which existing behaviors to preserve, or how strongly to update.

3 Learned Objective Transformation

Learned Objective Transformation (LOT) learns how a shared NTP objective should update a model for a chosen learning end. We describe the online update and offline meta-training procedure, then introduce three compiler parameterizations. Their gradient transformations provide the basis for the expressivity analysis.

3.1 Online adaptation

LOT keeps the backbone frozen and separates adaptation into two components: a persistent writer with parameters ϕ and a temporary compiler with parameters θ. At each writer update, the compiler remains fixed and shapes the NTP gradient applied to the writer. The compiler participates in the NTP update and is removed before evaluation. The updated writer remains part of the model.

Let c denote the observed context, ϕ the writer parameters before an update, and ϕ^+ the writer parameters after the update. We write $f_{\phi,\theta}(c)$ for the model when the compiler is active and $f_\phi(c)$ for the model without the compiler. The writer is updated using the context NTP loss:

$$\ell_\theta(\phi; c) = \mathcal{L}_{\text{NTP}}(f_{\phi,\theta}(c)), \qquad \boldsymbol{g}_\theta(\phi; c) = \nabla_\phi \ell_\theta(\phi; c), \qquad \phi^+ = \phi - \eta \boldsymbol{g}_\theta(\phi; c). \tag{1}$$

After adaptation, evaluation uses the updated model f_{ϕ^+} with the compiler removed.

3.2 Meta-training the compiler

The compiler is meta-trained offline for each specified learning end. Each meta-training episode initializes the writer with $\Delta W_\phi = 0$ and differentiates through K NTP updates. Let $\phi_{0:K}(\theta)$ denote the resulting writer trajectory. For a chosen learning end p, an outer criterion $\mathcal{Q}_{p,e}$ evaluates the induced update or the behavior of the updated model after compiler removal on episode e:

$$\theta_p^\star = \arg\min_\theta \; \mathbb{E}_e \left[\mathcal{Q}_{p,e}(\phi_{0:K}(\theta)) \right]. \tag{2}$$

Gradient-control tasks match the NTP gradient to a requested target gradient. The target is evaluated at the current writer state and held fixed when differentiating the outer criterion. Behavioral tasks optimize the performance of the updated model after compiler removal. For example, *Acquire* minimizes hidden-QA NLL after a context-NTP update, optionally with an outer KL term that penalizes changes on unrelated prompts. Only the compiler parameters θ are updated when optimizing the outer criterion. During subsequent adaptation, the compiler is frozen and the writer learns from NTP on unseen contexts.

3.3 COMPILER PARAMETERIZATIONS

We instantiate the temporary compiler in three forms that differ in how they modify the model during adaptation. For an adapted projection, let W_0 denote the frozen backbone weight and $\Delta W_\phi = \gamma BA$ the persistent LoRA writer (Hu et al., 2022), where A is a fixed nonzero down-projection, B is a zero-initialized trainable up-projection, and γ is the LoRA scaling factor. We use W_{write} for the effective weight during an NTP update and W_{read} for the weight used during subsequent evaluation. All three parameterizations share $W_{\text{read}} = W_0 + \Delta W_\phi$.

Meta-LoRA adds a temporary perturbation ΔW_θ during adaptation,

$$W_{\text{write}} = W_0 + \Delta W_\phi + \Delta W_\theta. \tag{3}$$

Meta-HiRA applies a learned multiplicative modulation ΔH_θ (Huang et al., 2025),

$$W_{\text{write}} = (W_0 + \Delta W_\phi) \odot (1 + \Delta H_\theta). \tag{4}$$

BCA denotes a Meta-LoRA parameterization in which a learned linear transformation $C_\theta \in \mathbb{R}^{r\times r}$ is inserted between the LoRA factors B and A,

$$W_{\text{write}} = W_0 + \Delta W_\theta + \gamma BC_\theta A. \tag{5}$$

Meta-training jointly learns ΔW_θ and C_θ, with C_θ initialized to the identity. The former modifies the model during adaptation, while the latter directly transforms the gradient with respect to B. For analysis, we also consider a **C-only BCA** variant obtained by setting $\Delta W_\theta = 0$. At initialization, where $B = 0$, it leaves the logits and NTP loss unchanged while the gradient with respect to B is right-multiplied by $C_\theta^\top$. Hence $C_\theta = sI$ scales or reverses the update, while a nonscalar C_θ can change its direction. Appendix A.4 derives this result and extends it to nonzero writer states.

3.4 HOW THE COMPILER CHANGES THE NTP GRADIENT

We now examine how a temporary change to the forward path alters the writer gradient. We refer to the corresponding NTP update with the compiler removed as *Direct NTP*, and use the subscript off for its quantities. For one supervised token, let $\boldsymbol{p}_\theta$ denote the predicted probabilities, $\boldsymbol{e}_y$ the one-hot target, and $\boldsymbol{J}_\theta = \partial f_{\phi,\theta}/\partial\phi$ the logit Jacobian with respect to the writer parameters. Cross-entropy gives $\boldsymbol{g}_\theta = \boldsymbol{J}_\theta^\top(\boldsymbol{p}_\theta - \boldsymbol{e}_y)$. Comparing this gradient with the Direct NTP gradient yields

$$\begin{aligned}\boldsymbol{g}_\theta - \boldsymbol{g}_{\text{off}} &= \boldsymbol{J}_\theta^\top(\boldsymbol{p}_\theta - \boldsymbol{e}_y) - \boldsymbol{J}_{\text{off}}^\top(\boldsymbol{p}_{\text{off}} - \boldsymbol{e}_y) \\ &= \underbrace{\boldsymbol{J}_{\text{off}}^\top(\boldsymbol{p}_\theta - \boldsymbol{p}_{\text{off}})}_{\text{residual contribution}} + \underbrace{(\boldsymbol{J}_\theta - \boldsymbol{J}_{\text{off}})^\top(\boldsymbol{p}_\theta - \boldsymbol{e}_y)}_{\text{Jacobian contribution}}.\end{aligned} \tag{6}$$

The first term captures changes in the prediction residual, while the second captures changes in the Jacobian that maps this residual to the writer parameters. For sequence NTP, the same decomposition is averaged over supervised tokens. Meta-LoRA, Meta-HiRA, and BCA can affect both terms.

C-only BCA provides a useful special case. With $B = 0$ and $\Delta W_\theta = 0$, C_θ does not change the forward logits, so $\boldsymbol{p}_\theta = \boldsymbol{p}_{\text{off}}$ and the residual term vanishes. The compiler can therefore change the writer gradient through the Jacobian even when the logits and scalar NTP loss are unchanged.

3.5 BCA EXPRESSIVITY

We characterize BCA's gradient expressivity and its connection to the target objective evaluated without the compiler. Here, we write $M := \Delta W_\theta$ for the additive compiler perturbation and $C := C_\theta$ for the learned transformation. With A fixed and all trainable B factors initialized to zero, M determines the NTP weight gradient G_M, while C does not affect the forward model. The chain rule gives $\nabla_B \ell_{\text{NTP}} = \gamma G_M A^\top C^\top$, so C directly transforms the gradient with respect to B induced by M. For one adapted weight matrix, stacking $\gamma G_M A^\top$ across contexts gives X_M, while H stacks the corresponding fixed, compiler-independent target gradients with respect to B. Learning C therefore amounts to fitting $X_M C^\top$ to H.

Theorem 1 (BCA gradient expressivity) *For fixed M and unrestricted C, let $(C_M^\star)^\top = X_M^\dagger H$ and $P_M = X_M X_M^\dagger$, where $\dagger$ denotes the pseudoinverse. Then*

$$\|X_M C^\top - H\|_{\mathrm{F}}^2 = \underbrace{\|(I - P_M)H\|_{\mathrm{F}}^2}_{\textit{projection error}} + \underbrace{\|X_M (C - C_M^\star)^\top\|_{\mathrm{F}}^2}_{\textit{fitting error}}, \tag{7}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm.

For fixed M, $C_M^\star$ is the least-squares optimum and satisfies $X_M (C_M^\star)^\top = P_M H$. The projection error is determined by X_M, whereas the fitting error measures the deviation of C from $C_M^\star$. Thus, M determines the minimum error attainable over C, while C controls the fitting error for a fixed M. Meta-LoRA imposes the restriction $C = I$, whereas C-only BCA imposes $M = 0$. Full BCA optimizes both M and C, so its minimum $\|X_M C^\top - H\|_{\mathrm{F}}^2$ is no greater than that of either restricted variant under the same writer setup.

To connect this result to learning, consider n independent updates, each starting from the same zero writer state. Let $\mathcal{B} = [B_1^\top, \ldots, B_n^\top]^\top$ stack the independently updated B_i across n contexts, with other writer parameters held fixed. Define the target loss after compiler removal $\mathcal{L}_{\mathrm{tar}}(\mathcal{B}) = \sum_i \ell_{\mathrm{tar},i}(B_i)$. For the descent statement, assume $H = \nabla_{\mathcal{B}} \mathcal{L}_{\mathrm{tar}}(0)$ and that $\mathcal{L}_{\mathrm{tar}}$ is L-smooth in the Frobenius norm, with $L > 0$. The independent NTP SGD updates give $\mathcal{B}^+ = -\eta X_M C^\top$. For $0 < \eta \le 1/L$, Theorem 2 and Equation 7 yield

$$\begin{aligned} \mathcal{L}_{\mathrm{tar}}(\mathcal{B}^+) - \mathcal{L}_{\mathrm{tar}}(0) &\le -\frac{\eta}{2}\|H\|_{\mathrm{F}}^2 + \frac{\eta}{2}\|X_M C^\top - H\|_{\mathrm{F}}^2 \\ &= -\frac{\eta}{2}\|P_M H\|_{\mathrm{F}}^2 + \frac{\eta}{2}\|X_M (C - C_M^\star)^\top\|_{\mathrm{F}}^2. \end{aligned} \tag{8}$$

Thus the target loss decreases whenever the fitting error is smaller than $\|P_M H\|_{\mathrm{F}}^2$. At $C = C_M^\star$, any nonzero $P_M H$ guarantees strict descent. For joint updates of multiple projections, the corresponding terms are summed across projections. Appendix A.4 provides the proofs, further analyzes the projection and fitting errors, and discusses extensions to nonzero writer states.

The information available during adaptation also determines which updates can be learned. Appendix A separates target variation predictable from the observed context from variation requiring additional information.

4 Experiments

We first use controlled gradient and behavioral targets to test whether a shared context-NTP objective can induce different updates. We then evaluate answer-oriented adaptation, answer generation, and retention over longer streams. Finally, we examine how the learned compilers alter the NTP gradient, first through C in C-only BCA and then through its residual and Jacobian components.

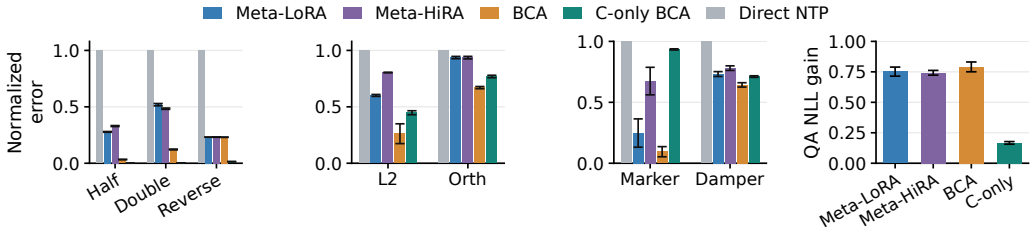
4.1 Experimental setup

The primary experiments use Qwen3-0.6B (Yang et al., 2025) and context-disjoint data from the MS MARCO-MQA block of SHINE's IFT-MQA collection (Liu et al., 2026). We additionally replicate the acquisition results on Qwen3.5-0.8B (Qwen Team, 2026) and Llama-3.2-1B (Meta, 2024). The persistent writer has rank 4, while Meta-LoRA and Meta-HiRA use rank 32 over attention and MLP projections. BCA additionally learns one C per adapted projection. Each meta-training episode starts from a zero-initialized writer. Evaluation is performed on unseen contexts, while continual experiments retain the same writer across the full stream. All reported model outputs are evaluated with the compiler removed. The matched architecture, generation, and continual studies use three meta-training seeds. Appendix B lists study-specific experimental settings.

For gradient analysis, we report the normalized squared error $E = \|\boldsymbol{g}_\theta - \boldsymbol{g}^\star\|_2^2 / \|\boldsymbol{g}_{\mathrm{off}} - \boldsymbol{g}^\star\|_2^2$, where zero denotes exact matching and one corresponds to Direct NTP. The metric captures both directional and magnitude errors. Task-specific metrics and averaging rules are given in Appendix D.

Table 1: **Behavior after adaptation in the matched comparison.** L2 and Orthogonality report penalty ratios relative to Direct NTP. Damper reports final QA NLL gain over Direct NTP. Acquire reports QA NLL gain over Direct NTP and unrelated-prompt KL.

Compiler	L2 penalty ↓	Orth. penalty ↓	Damper QA NLL gain ↑	Acquire QA NLL gain ↑	Acquire KL ↓
Meta-LoRA	0.334 ± 0.008	0.678 ± 0.053	-0.016 ± 0.015	0.752 ± 0.037	2.045 ± 0.564
Meta-HiRA	0.663 ± 0.003	0.831 ± 0.007	-0.020 ± 0.016	0.742 ± 0.019	1.663 ± 0.200
BCA	0.261 ± 0.105	0.815 ± 0.021	-0.039 ± 0.019	0.791 ± 0.040	2.443 ± 0.924
C-only BCA	0.161 ± 0.011	0.646 ± 0.025	-0.039 ± 0.011	0.167 ± 0.011	0.058 ± 0.024



(a) Gradient control (b) Regularization (c) Marker and Damper (d) Acquire

Figure 2: **Compiler architectures across learning ends.** The four panels evaluate gradient control, regularization, Marker and Damper, and Acquire under the same context-NTP objective. (a–c) Target error normalized to 1 for Direct NTP and 0 at the target, using classification error for Marker and squared gradient error for the other tasks. (d) Acquire QA NLL gain over Direct NTP after the compiler is removed (higher is better). Results are means ± sample SD across runs. Appendix D gives the full metric definitions and explains how each measure is computed for each learning end.

4.2 Learning ends under a shared NTP objective

We first test whether a shared context-NTP objective can serve different learning ends. We consider gradient control, regularization, Marker and Damper, and Acquire. Figure 2 reports target matching across these settings, while Table 1 reports the corresponding changes in behavior after adaptation.

The simplest targets rescale or reverse the Direct NTP gradient. For **Half**, **Double**, and **Reverse**, the target gradients are $0.5\boldsymbol{g}_{\text{off}}$, $2\boldsymbol{g}_{\text{off}}$, and $-\boldsymbol{g}_{\text{off}}$, respectively. BCA matches the Half and Double targets more accurately than Meta-LoRA across repeated experiments, while C-only BCA gives the lowest errors (Figure 2(a)). For Reverse, a zero update already gives $E = 0.25$. To measure the sign and magnitude along the Direct NTP gradient, we also report the projection coefficient $a = \langle \boldsymbol{g}_\theta, \boldsymbol{g}_{\text{off}} \rangle / \|\boldsymbol{g}_{\text{off}}\|_2^2$. C-only BCA reaches $a = -0.784$ with $E = 0.014$, whereas Meta-LoRA, Meta-HiRA, and BCA remain near $a = -0.1$ with errors around 0.23. C-only BCA also matches an equal-norm 90° rotated target on unseen contexts (Appendix C.3), showing that C can change both the magnitude and direction of the update rather than only rescaling it.

We next evaluate whether NTP updates can approximate gradients of regularized objectives. **L2** penalizes writer magnitude, while **Orthogonality** penalizes correlation between writer columns. Both are evaluated at the same nonzero writer state, so their regularization gradients are nonzero. As shown in Figure 2(b), BCA gives the lowest gradient error on both tasks among Meta-LoRA, Meta-HiRA, and BCA. The penalties after adaptation show a different ordering: BCA gives the lowest L2 penalty among the three, and Meta-LoRA gives a lower Orthogonality penalty. As shown in Table 1, C-only BCA yields the lowest penalties after adaptation.

A further challenge is that the desired update may depend on the current writer state. In **Marker**, the updated model predicts whether it has previously learned from a candidate passage. BCA reaches 95.2% balanced accuracy, compared with 87.2% for Meta-LoRA, and 65.5% for Meta-HiRA, while C-only BCA remains near chance. In **Damper**, the target update becomes progressively smaller over four repeated updates. As shown in Figure 2(c), BCA gives the lowest target error. The learned update scales decrease across repeated updates but do not fully match the target update, and all tested Damper compilers have lower final QA likelihood than Direct NTP, as shown in Table 1.

Finally, **Acquire** uses answer likelihood after adaptation as the meta-training objective, with the compiler removed when scoring answers. As shown in Figure 2(d), after a passage-only NTP update, Meta-LoRA, Meta-HiRA, and BCA all reduce held-out QA NLL relative to Direct NTP, while C-only BCA gives a smaller gain. Their effects on unrelated prompts differ substantially. As shown in Table 1, Meta-HiRA yields lower unrelated-prompt KL than Meta-LoRA and BCA, while C-only BCA changes unrelated predictions very little. This provides evidence that a fixed NTP objective can serve a downstream learning end without changing the online supervision.

The shared NTP objective produces different adaptation effects across these tasks. The compiler parameterization affects both gradient matching and the behavior retained after adaptation, with the relative performance varying across learning ends.

4.3 Answer-oriented adaptation

We next test whether stronger Direct NTP controls can reproduce the Meta-LoRA acquisition gain. Figure 3 compares Meta-LoRA with three increasingly flexible controls. **Best Direct** selects the best configuration from a sweep over Direct NTP learning rates and numbers of backward passes. **Exact norm** rescales the Direct NTP update to match Meta-LoRA's update norm, while **Module rate** fits a separate signed learning rate for each writer.

Relative to the pre-update model, Meta-LoRA attains a QA NLL reduction of 0.707, compared with 0.337 for Best Direct, 0.224 for Exact norm, and 0.126 for Module rate. Meta-LoRA retains a larger gain across the tested compute budgets, after matching update magnitude, and after allowing independent signed learning rates for each module. The lower gain of Module rate shows that module-wise rescaling alone does not reproduce the Meta-LoRA update, pointing to changes in update direction within modules. The same pattern holds on two additional backbones. Separately trained compilers improve QA NLL over Direct NTP by 0.578 on Qwen3.5-0.8B and 0.135 on Llama-3.2-1B. The corresponding gains over the Module rate controls are 0.485 and 0.131, respectively (Appendix C.1). Thus, the acquisition gain is consistent across all three tested backbones, including comparisons with independently fitted module-wise learning rates.

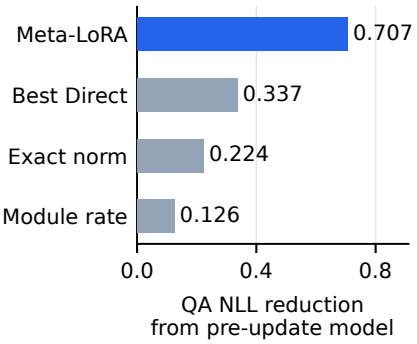


Figure 3: **Acquire controls.** Mean QA NLL reduction from the pre-update model.

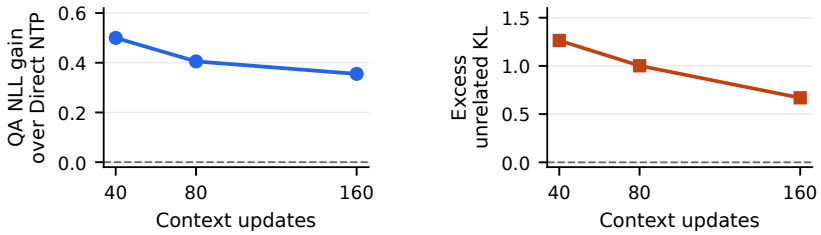
4.4 Answer generation after adaptation

The acquisition experiments above measure answer likelihood under teacher forcing. We also evaluate free-form answers, where both content and stopping affect performance. Meta-LoRA is meta-trained to predict the full answer and the end-of-sequence (EOS) token after a passage NTP update. A second variant adds an unrelated-prompt KL penalty to the outer objective. Both compilers then guide passage NTP adaptation on the test contexts.

Table 2 reports generation results after passage NTP adaptation, with the compiler removed before evaluation. Using answer and EOS supervision during meta-training increases answer F1 from 9.56 for Direct NTP to 18.71. The proportion of outputs reaching the 24-token cap falls from 100% to 5.86%, and the mean output length falls from 24.00 to 10.60 tokens. Adding KL regularization lowers unrelated-prompt KL from 5.82 to 3.54, with a small decrease in F1 to 18.44.

Table 2: **Answer generation after adaptation.** We use greedy decoding over 128 contexts with a 24-token cap and report means across repeated experiments.

Condition	Answer F1 (%) ↑	At 24-token cap (%) ↓	Length (tokens)	Unrelated KL ↓
Direct NTP	9.56	100.00	24.00	0.013
Answer+EOS	18.71	5.86	10.60	5.820
Answer+EOS+KL	18.44	10.03	12.50	3.540



(a) Retention (b) Unrelated-prompt KL

Figure 4: **Long-stream retention.** QA NLL gain and excess unrelated-prompt KL relative to Direct NTP through 160 updates. Curves show repeated-experiment means.

4.5 Long-stream retention

We next evaluate Meta-LoRA over streams substantially longer than those used during meta-training. Each meta-training episode contains eight updates. The compiler is then evaluated, without further meta-training, on streams of 40, 80, and 160 unseen contexts. At these three horizons, Meta-LoRA improves QA NLL over Direct NTP by 0.500, 0.405, and 0.355, respectively. After 160 updates, unrelated-prompt KL is higher than Direct NTP by 0.669, while a separate forgetting measure is lower by 0.012. Supplementary comparisons are reported in Appendix C.

To determine whether the performance stems from the adaptation process itself rather than a better starting point, we further compare the learned update rule with a MAML-style learned writer initialization (Finn et al., 2017). The learned initialization provides a 0.845 QA NLL reduction before any context update and gains a further 0.080 over the stream. Meta-LoRA has no pre-stream improvement and accumulates a 0.908 reduction through the context-NTP updates. Their final gains over Direct NTP are similar, at 0.372 and 0.355, respectively. The two methods obtain their gains at different stages. The learned initialization supplies most of its improvement before the stream begins, and Meta-LoRA accumulates its improvement through successive context updates.

Table 3: **Learned initialization versus learned update rule.** QA NLL reductions after 160 context updates. Mean ± sample SD across runs, each averaged over three shared streams.

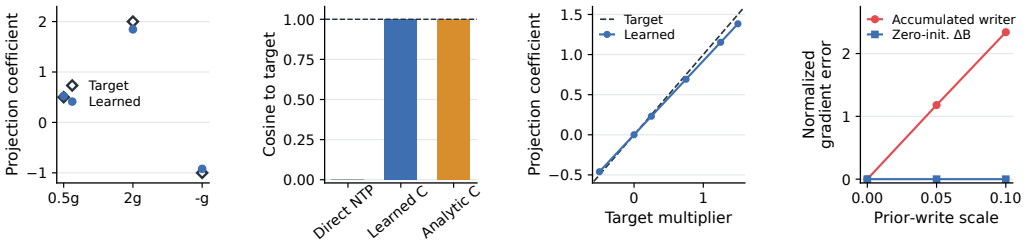
Meta-trained component	Before updates	From updates	Final vs. Direct NTP
Writer initialization	0.845 ± 0.012	0.080 ± 0.014	0.372 ± 0.005
Meta-LoRA compiler	0.000	0.908 ± 0.024	0.355 ± 0.016

We further evaluate the SHINE-trained compiler on LoCoMo (Maharana et al., 2024) without additional meta-training, using its annotated evidence as the incoming context stream. After 40 updates, Meta-LoRA achieves a QA NLL reduction of 0.725, compared with 0.311 for Direct NTP, while forgetting decreases from 0.135 to 0.068. This provides a cross-dataset evaluation of the learned update rule (Appendix C).

4.6 Reusing learned transformations

We examine C to understand how the compiler changes NTP updates. C-only BCA isolates C without changing the forward predictions at a zero-initialized writer. We use this setting to test whether the learned transformation generalizes to unseen contexts, whether learned transformations combine according to their linear action on the writer gradient, and whether the same transformation remains applicable after previous updates have accumulated in the writer.

The learned transformations retain their intended effect on unseen contexts (Figure 5a–b). In particular, for the equal-norm 90° rotation target, the learned transformation achieves a cosine similarity of 0.999 with the target and a norm ratio of 0.997, showing that C can change gradient direction while preserving its magnitude. Affine combinations of separately learned endpoint transformations also produce the corresponding intermediate transformations without additional meta-training (Figure 5c). Their projection coefficient varies almost linearly with the requested scale ($R^2 \approx 1.000$), as expected from its linear dependence on C at initialization.



(a) Scalar rules (b) Rotation (c) Composition (d) Nonzero writer state

Figure 5: **Reusing learned transformations.** (a) Learned scalar transformations. (b) Learned equal-norm $90°$ rotation. (c) Affine combinations of separately learned transformations. (d) After previous writer updates, applying C only to a zero-initialized ΔB preserves the current predictions, whereas applying it to the accumulated writer changes them. Plotted results are means.

Applying the same transformation after earlier updates requires separating the existing writer state from the new update. Applying C to the accumulated writer generally changes the current predictions. We therefore introduce a separate zero-initialized ΔB and apply C only through ΔBCA. At $\Delta B = 0$, the current predictions are unchanged, while the resulting update matches the transformed update to numerical precision (Figure 5d). The updated ΔB is then merged into the accumulated writer before the next context.

4.7 Residual and Jacobian Contributions

C-only BCA provides an explicit transformation of the writer gradient. We next use Equation 6 to quantify how the compiler changes the NTP gradient. For Acquire, 99.8% of the projected gradient change is assigned to the Jacobian term, with only 0.2% assigned to the change in the prediction residual. Half depends on both terms, but the Jacobian contribution remains larger at 74.8% (Table 4). The decomposition assigns most of Acquire's gradient change to the mapping from prediction errors to writer gradients. Its prediction residual contributes very little.

Table 4: **Signed contributions (%).** Each decomposition sums to 100. Acquire QA uses exact-norm matching.

Component	Acquire	Half
Gradient change		
Residual	0.21	25.17
Jacobian	99.79	74.83
Acquire QA alignment		Share
Residual		0.06
Jacobian		137.84
Norm correction		−37.90

The same distinction appears when the Acquire update is compared with the QA gradient of the model without the compiler under exact-norm matching. The Jacobian term contributes 137.8% of the net positive alignment, while the norm correction contributes -37.9% and offsets part of that contribution. The residual term is negligible. The norm correction arises from matching the update norm and is separate from the two terms in Equation 6. Across these comparisons, Acquire is dominated by the Jacobian term, whereas Half receives substantial contributions from both residual and Jacobian terms. These signed shares use the Direct NTP Jacobian as their reference, as specified in Equation 6.

5 Conclusion

Models adapting to incoming text can readily obtain next-token targets, even when direct supervision for the intended adaptation outcome is unavailable. Heterotelic Learning studies how supervision available offline can shape these online updates to serve a chosen learning end. We introduced LOT, which learns a temporary compiler from supervision specifying desired behaviors or target gradients to reshape updates under a shared NTP objective. The compiler is removed before evaluation, leaving only the resulting parameter changes. Across gradient control, regularization, answer-oriented adaptation, and longer update streams, LOT produces distinct adaptation effects from the same online supervision. Our analysis further shows that these effects can arise through changes in gradient geometry. These results provide a concrete framework for studying how widely available training objectives can be reused for different learning ends.

AI USE STATEMENT

The authors used generative AI tools for several auxiliary tasks, including improving the manuscript's language, organizing LaTeX content, checking numerical calculations, and formatting tables. Any material produced with AI assistance was examined and, where necessary, revised by the authors before inclusion. Responsibility for the final text, calculations, claims, and artifacts remains entirely with the authors.

REPRODUCIBILITY STATEMENT

Theoretical proofs are provided in Appendix A, and experimental protocols in Appendices B and D. The anonymized source package provides a minimal LOT implementation with runnable examples, dependency specifications, and validation tests. The full experimental pipelines, original data splits, and trained checkpoints are not included.

REFERENCES

Marcin Andrychowicz, Misha Denil, Sergio Gómez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016.

Sungyong Baik, Janghoon Choi, Heewon Kim, Dohee Cho, Jaesik Min, and Kyoung Mu Lee. Meta-learning with task-adaptive loss function for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9465–9474. IEEE, 2021.

Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta learning via learned loss. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pp. 4161–4168. IEEE, 2021.

Zeming Chen, Angelika Romanou, Gail Weiss, and Antoine Bosselut. PERK: Long-context reasoning as parameter-efficient test-time learning. In *International Conference on Learning Representations (ICLR)*, 2026.

Matthew Chrisman and Graham Hubbs. Epistemic normativity as autotelic: why suspension and judgment aren't like archery. *Philosophical Studies*, 2025. Published online 11 October 2025.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1: Long and Short Papers*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 2017.

Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *International Conference on Learning Representations (ICLR)*, 2020.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1568–1577. PMLR, 2018.

Lucas Torroba Hennigen, Hunter Lang, Han Guo, and Yoon Kim. On the duality between gradient transformations and adapters. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 23030–23046. PMLR, 2025.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Nathan Hu, Eric Mitchell, Christopher D. Manning, and Chelsea Finn. Meta-learning online adaptation of language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4418–4432. Association for Computational Linguistics, 2023.

Qiushi Huang, Tom Ko, Zhan Zhuang, Lilian Tang, and Yu Zhang. HiRA: Parameter-efficient Hadamard high-rank adaptation for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Ben Krause, Emmanuel Kahembwe, Iain Murray, and Steve Renals. Dynamic evaluation of neural sequence models. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2766–2775. PMLR, 2018.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Yewei Liu, Xiyuan Wang, Yansheng Mao, Yoav Gelbery, Haggai Maron, and Muhan Zhang. SHINE: A scalable in-context hypernetwork for mapping context to LoRA in a single pass. *arXiv preprint arXiv:2602.06358*, 2026.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Volume 1: Long Papers*, pp. 13851–13870. Association for Computational Linguistics, 2024.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 17359–17372, 2022.

Meta. Llama 3.2 model card. Official model documentation, 2024.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *International Conference on Learning Representations (ICLR)*, 2022.

Alexander P. D. Mourelatos. Aristotle's kinêsis/energeia distinction: A marginal note on Kathleen Gill's paper. *Canadian Journal of Philosophy*, 23(3):385–388, 1993.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Eunbyung Park and Junier B. Oliva. Meta-Curvature. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Qwen Team. Qwen3.5-0.8B. Hugging Face model card, 2026.

Arthur S. Reber. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 6 (6):855–863, 1967.

Anna C. Schapiro, Timothy T. Rogers, Natalia I. Cordova, Nicholas B. Turk-Browne, and Matthew M. Botvinick. Neural representations of events arise from temporal community structure. *Nature Neuroscience*, 16(4):486–492, 2013.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248. PMLR, 2020.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.

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

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2022.

A SUPPORTING THEORY AND PROOFS

We use the notation of Section 3: ϕ denotes the writer parameters, θ the compiler parameters held fixed during an online update, and $\boldsymbol{g}_\theta$ the resulting NTP gradient. Any target criterion F is evaluated with the compiler removed.

A.1 NTP GRADIENT DECOMPOSITION

For logits $f_{\phi,\theta}$ and label y, cross-entropy gives

$$\ell_\theta = -f_{\phi,\theta,y} + \log \sum_j e^{f_{\phi,\theta,j}}, \qquad \nabla_\phi \ell_\theta = \boldsymbol{J}_\theta^\top (\boldsymbol{p}_\theta - \boldsymbol{e}_y). \tag{9}$$

With fixed token weights $\sum_t w_t = 1$, adding and subtracting $\boldsymbol{J}_{\mathrm{off},t}^\top(\boldsymbol{p}_{\theta,t} - \boldsymbol{e}_{y_t})$ yields

$$\begin{aligned} \boldsymbol{g}_\theta &= \sum_t w_t \boldsymbol{J}_{\theta,t}^\top (\boldsymbol{p}_{\theta,t} - \boldsymbol{e}_{y_t}), \\ \boldsymbol{g}_\theta - \boldsymbol{g}_{\mathrm{off}} &= \sum_t w_t \left[\boldsymbol{J}_{\mathrm{off},t}^\top (\boldsymbol{p}_{\theta,t} - \boldsymbol{p}_{\mathrm{off},t}) + (\boldsymbol{J}_{\theta,t} - \boldsymbol{J}_{\mathrm{off},t})^\top (\boldsymbol{p}_{\theta,t} - \boldsymbol{e}_{y_t}) \right]. \end{aligned} \tag{10}$$

This decomposition is exact with Direct NTP as the reference, although the signed contributions of the two terms can cancel.

A.2 ONE-STEP OBJECTIVE TRANSFORMATION

Theorem 2 (One-step objective transformation) *Let $\boldsymbol{g}^\star = \nabla_\phi F(\phi)$ and $\phi^+ = \phi - \eta \boldsymbol{g}_\theta$. If F has an L-Lipschitz gradient along the update segment, $L > 0$, and $0 < \eta \le 1/L$, then*

$$\begin{aligned} \|\phi^+ - (\phi - \eta \boldsymbol{g}^\star)\|_2 &= \eta \|\boldsymbol{g}_\theta - \boldsymbol{g}^\star\|_2, \\ F(\phi) - F(\phi^+) &\ge \tfrac{\eta}{2} \left(\|\boldsymbol{g}^\star\|_2^2 - \|\boldsymbol{g}_\theta - \boldsymbol{g}^\star\|_2^2 \right). \end{aligned} \tag{11}$$

Proof. The state identity follows by subtraction. Smoothness gives

$$\begin{aligned} F(\phi) - F(\phi^+) &\ge \eta \langle \boldsymbol{g}^\star, \boldsymbol{g}_\theta \rangle - \tfrac{L\eta^2}{2} \|\boldsymbol{g}_\theta\|_2^2 \\ &= \tfrac{\eta}{2} \left(\|\boldsymbol{g}^\star\|_2^2 - \|\boldsymbol{g}_\theta - \boldsymbol{g}^\star\|_2^2 \right) + \tfrac{\eta}{2}(1 - L\eta) \|\boldsymbol{g}_\theta\|_2^2. \end{aligned} \tag{12}$$

The last term is inherently nonnegative. Consequently, a gradient error strictly bounded by $\|\boldsymbol{g}^\star\|_2$ guarantees descent of the objective function. For $F = \ell_{\mathrm{off}} + \lambda R$, the target becomes $\boldsymbol{g}^\star = \boldsymbol{g}_{\mathrm{off}} + \lambda \nabla_\phi R$. Thus, a decrease in F need not imply a decrease in R alone.

A.3 INFORMATION AVAILABLE DURING ADAPTATION

Fix a trained compiler and let O denote the information available during adaptation. Define $\bar{\boldsymbol{g}}(O) = \mathbb{E}[\boldsymbol{g}^\star \mid O]$. For square-integrable target gradients and any deterministic update rule $\boldsymbol{g}_\theta(O)$,

$$\begin{aligned} \mathbb{E}\langle \boldsymbol{g}_\theta - \bar{\boldsymbol{g}}, \bar{\boldsymbol{g}} - \boldsymbol{g}^\star \rangle &= \mathbb{E}\langle \boldsymbol{g}_\theta - \bar{\boldsymbol{g}}, \mathbb{E}[\bar{\boldsymbol{g}} - \boldsymbol{g}^\star \mid O] \rangle = 0, \\ \mathbb{E}\|\boldsymbol{g}_\theta - \boldsymbol{g}^\star\|_2^2 &= \mathbb{E}\|\boldsymbol{g}^\star - \bar{\boldsymbol{g}}\|_2^2 + \mathbb{E}\|\boldsymbol{g}_\theta - \bar{\boldsymbol{g}}\|_2^2. \end{aligned} \tag{13}$$

The first term is the irreducible error given O. For example, an independent equiprobable sign $Z \in \{-1, +1\}$ makes $Z\boldsymbol{g}_{\mathrm{off}}(O)$ have conditional mean zero; a fixed reversal remains predictable. For observations with $\sigma(O_0) \subseteq \sigma(O)$, where O_0 omits the current context, the same orthogonality gives

$$\begin{aligned} &\mathbb{E}\|\boldsymbol{g}^\star - \mathbb{E}[\boldsymbol{g}^\star \mid O_0]\|_2^2 - \mathbb{E}\|\boldsymbol{g}^\star - \mathbb{E}[\boldsymbol{g}^\star \mid O]\|_2^2 \\ &= \mathbb{E}\|\mathbb{E}[\boldsymbol{g}^\star \mid O] - \mathbb{E}[\boldsymbol{g}^\star \mid O_0]\|_2^2 \ge 0. \end{aligned} \tag{14}$$

A compiler must additionally approximate the predictable component.

A.4 Proof of BCA expressivity

For one adapted weight matrix, let $B \in \mathbb{R}^{d_{\text{out}} \times r}$, $A \in \mathbb{R}^{r \times d_{\text{in}}}$, and $C \in \mathbb{R}^{r \times r}$. Adaptation uses $W_0 + M + \gamma BCA$, and evaluation uses $W_0 + \gamma BA$. When all trainable B factors are zero, the weight gradient G_M is independent of every C. Holding the compiler fixed,

$$\begin{aligned} d\ell_{\text{NTP}} = \langle G_M, \gamma\, dBCA \rangle_{\text{F}} = \langle \gamma G_M A^\top C^\top, dB \rangle_{\text{F}}, \\ \nabla_B \ell_{\text{NTP}} = \gamma G_M A^\top C^\top. \end{aligned} \tag{15}$$

Let X_M and H denote the matrices obtained by stacking $\gamma G_M A^\top$ and the fixed target gradients across contexts, respectively. We then define $(C_M^\star)^\top = X_M^\dagger H$ and the orthogonal projector as $P_M = X_M X_M^\dagger$. Since P_M is an orthogonal projector,

$$\begin{aligned} X_M C^\top - H &= X_M (C - C_M^\star)^\top - (I - P_M) H, \\ X_M^\top (I - P_M) H &= 0, \\ \|X_M C^\top - H\|_F^2 &= \|(I - P_M)H\|_F^2 + \|X_M (C - C_M^\star)^\top\|_F^2 \\ &\le \|(I - P_M)H\|_F^2 + \|X_M\|_{\text{op}}^2 \|C - C_M^\star\|_F^2, \\ \inf_{M,C} \|X_M C^\top - H\|_F^2 &= \inf_M \|(I - P_M)H\|_F^2. \end{aligned} \tag{16}$$

This proves Theorem 1. A shared C fits exactly if and only if every column of H lies in $\text{col}(X_M)$. The restrictions $C = I$ (Meta-LoRA) and $M = 0$ (C-only) give

$$\inf_{M,C} \|X_M C^\top - H\|_F^2 \le \min \left\{ \inf_M \|X_M - H\|_F^2, \inf_C \|X_0 C^\top - H\|_F^2 \right\}. \tag{17}$$

The analysis above considers a single adapted weight matrix at $B = 0$, where G_M is independent of C and the fitting problem is linear in C. For multiple adapted weight matrices, the same argument applies by summing the corresponding error terms. The resulting bounds describe the optimal fit under a fixed rank, writer basis, and error metric; the error achieved in practice may additionally depend on optimization. When $B \neq 0$, however, G_M generally depends on C, so the fitting problem is no longer linear in C.

A.5 C-only gradient transformations

With $M = 0$, fixed A, and trainable B, one SGD update is

$$\nabla_B \ell_{\text{NTP}} = \gamma G A^\top C^\top, \qquad \Delta W_{\text{read}} = -\eta \gamma^2 G A^\top C^\top A. \tag{18}$$

At $B = 0$, logits and G are independent of C, so $C = sI$ exactly scales or reverses Direct NTP. For even rank, an orthogonal skew-symmetric C can induce an equal-norm 90° rotation. With $X = \gamma G A^\top$,

$$\langle X, XC^\top \rangle_F = \text{tr}(X^\top X C^\top) = 0, \qquad \|XC^\top\|_F^2 = \text{tr}(C X^\top X C^\top) = \|X\|_F^2. \tag{19}$$

The first trace is zero because $X^\top X$ is symmetric and $C^\top$ is skew-symmetric, while the norm equality follows from the orthogonality of C. The resulting gradient has equal norm and an orthogonal direction. The transformed gradients remain restricted to $\text{col}(X)$:

$$\min_C \|XC^\top - H\|_F^2 = \|(I - XX^\dagger)H\|_F^2. \tag{20}$$

Across contexts, X, H must be stacked before fitting the shared transformation.

A.6 Applying C after previous writer updates

For C-only BCA, let B_t denote the writer accumulated from previous updates. To apply the learned transformation to the next update without altering B_t, we introduce a separate variable ΔB, initialized at zero:

$$\begin{aligned} W_{\text{write}} = W_0 + \gamma(B_t A + \Delta B C A), \quad W_{\text{read}} = W_0 + \gamma(B_t + \Delta B)A, \\ \nabla_{\Delta B} \ell_{\text{NTP}} = X_t C^\top, \quad X_t = \gamma G_t A^\top, \qquad B_{t+1} = B_t - \eta X_t C^\top, \end{aligned} \tag{21}$$

where G_t is the NTP weight gradient at the current writer state B_t.

At $\Delta B = 0$, C does not affect the forward model, so X_t is independent of C. Nevertheless, C determines the update to B_t and therefore affects the target criterion evaluated after the update. Let $H^+ = \nabla_{B_{t+1}} F$. For one SGD step,

$$\begin{aligned} dF &= -\eta \langle H^+, X_t\, dC^\top \rangle_{\mathrm{F}} = -\eta \langle (H^+)^\top X_t, dC \rangle_{\mathrm{F}}, \\ \nabla_C F|_{B_t \text{ fixed}} &= -\eta (H^+)^\top X_t. \end{aligned} \tag{22}$$

The same independence from C holds when the additive compiler M is included, because the term ΔBCA vanishes at $\Delta B = 0$. In that case, M may change the adaptation predictions, while Theorem 1 applies to the update in ΔB with B_t fixed. The derivation above assumes one plain-SGD step from $\Delta B = 0$.

B Outer criteria and protocols

B.1 Writer state and outer criteria

Each episode samples a nonzero writer basis A and resets the trainable B factors to zero; the backbone remains frozen. The matched Damper comparison updates both A and B, whereas C-only BCA studies fix A. The compiler is fixed during updates and removed for evaluation. C-only BCA initializes $C = I$ and uses BCA for adaptation and BA for evaluation. For repeated updates, each updated ΔB is merged into B before the next context (Equation 21).

NTP sequences start with `Passage:` and supervise passage tokens only. Task-specific outer supervision, including questions, answers, regularization terms, and Marker labels, is used only during meta-training or final evaluation. Paired comparisons share tokenization, supervised positions, loss reduction, writer initialization, inner learning rate, update rule, and evaluation with the compiler removed. The compiler may change the NTP loss through the forward model, while the next-token targets and cross-entropy rule remain unchanged.

Acquire objective. The outer criterion, evaluated with the compiler removed, is $J_{\lambda,e} = L_e^{\mathrm{QA}} + \lambda R_e$, where R_e is reference KL on unrelated prompts. Prefix acquisition scores two answer tokens; the full-answer study scores 32 content tokens and the gold EOS target, with Answer+EOS and Answer+EOS+$0.1R_e$ variants.

Gradient scaling and regularization. For $a \in \{0.5, 2, -1\}$, the scaling criterion and achieved coefficient are

$$\mathcal{Q}_{\mathrm{scale}}(a) = \frac{\|\boldsymbol{g}_\theta - a\boldsymbol{g}_{\mathrm{off}}\|_2^2}{\|\boldsymbol{g}_{\mathrm{off}}\|_2^2 + \epsilon}, \qquad \widehat{a} = \frac{\langle \boldsymbol{g}_\theta, \boldsymbol{g}_{\mathrm{off}} \rangle}{\|\boldsymbol{g}_{\mathrm{off}}\|_2^2}, \tag{23}$$

where $\epsilon > 0$ is a numerical stabilizer. Regularization targets are $\boldsymbol{g}^\star = \boldsymbol{g}_{\mathrm{off}} + \lambda \nabla_\phi R$ for reference KL, factor L2, or column orthogonality.

Continual learning. For $K = 8$ meta-training updates, let $L_{j,t}$ be QA NLL after compiler removal for context j after update t. The outer criterion is

$$J_{\mathrm{cont}} = \frac{1}{K} \sum_{t=1}^{K} \left[L_{t,t} + 5 \frac{\sum_{j<t} [L_{j,t} - L_{j,j}]_+}{\max(1, t-1)} + 0.05\, \mathrm{KL}_t \right]. \tag{24}$$

Each adaptation step uses the NTP loss of the current context.

B.2 Data, models, and updates

Nonempty User/Assistant pairs from the training datasets are grouped by source document and ordered deterministically for the context-disjoint split below. Over 95% of answers exceed two tokens, so this metric measures answer-prefix likelihood.

Table 5: **Context-disjoint SHINE/MS MARCO-MQA split.**

	Train	Development	Test
Contexts	4,096	64	128
QA pairs	61,440	960	1,920
Median answer tokens	13	12	14
95th percentile	34	31	35

Qwen3-0.6B is the default frozen backbone. Its rank-4 writer covers 196 attention and MLP projections (1.38M trainable writer parameters); the temporary rank-32 compiler covers the same projections (20.19M parameters). Qwen3.5-0.8B and Llama-3.2-1B use separately trained compilers.

The gradient-matching experiments partition the writer into blocks b and use an additional compiler-off backward pass to compute the Direct NTP gradient. Its blockwise RMS magnitudes define a shared normalization across compared methods:

$$\sigma_b(s) = \max\left\{\frac{\|\boldsymbol{g}_{\text{off}}^{(b)}(s)\|_2}{\sqrt{d_b}}, 10^{-8}\right\}, \qquad \boldsymbol{D}_{\text{off}}(s) = \text{blockdiag}_b(\sigma_b(s)^{-1}\boldsymbol{I}_{d_b}). \tag{25}$$

The update is $\phi^+ = \phi - \rho \boldsymbol{D}_{\text{off}}(s)\boldsymbol{g}_\theta$, and matched Marker and Acquire comparisons use the same per-context normalization. For C-only BCA checks, each update instead uses plain SGD with one compiler-on NTP backward pass. The Direct NTP gradient is computed separately for measurement and is not used to construct that update.

B.3 Evaluation settings

Table 6: **Experimental settings.** Ctx denotes the maximum number of online passage tokens.

Study	Train ctxs	Meta steps	Seeds	Test ctxs	Ctx	Evaluation
Qwen3-0.6B Acquire	1,024	320	5	128	96	2 QA tokens
Qwen3.5 Acquire+KL	1,024	320	3	128	96	4 QA tokens
Llama-3.2 Acquire+KL	1,024	320	5	128	96	4 QA tokens
Acquire/Half study	1,024	320	3	128	96	QA/gradient
Full answer EOS / EOS+KL	1,024	960	3	128	96	32+EOS; greedy 24
Long-stream Meta-LoRA	1,024	256	3	3 streams	1,024	40/80/160 updates
Learned writer initialization	1,024	256	3	3 streams	1,024	40/80/160 updates
Gradient-matching suite	256 unique	960	1 or 3	128	96	gradient/task outcome
LoCoMo transfer	SHINE only	–	3	3 streams	1,536	10/20/40 updates

Single-context evaluation resets B per episode; continual evaluation resets once per stream. Native Acquire and the Acquire/Half study use disjoint 128-context tests; the latter also tests three unseen writer bases. Qwen3.5 and Llama use the same context split, a shared Direct NTP update, and outer criterion $L^{\text{QA}} + 0.2R_{\text{KL}}$. Marker pairs the adaptation passage with an unrelated candidate, reporting balanced accuracy from a fixed Yes/No prompt with the compiler removed.

For experiments incorporating EOS supervision, we evaluate the content and EOS probabilities using teacher forcing, followed by greedy decoding of a fixed prompt for a maximum of 24 tokens. Continual evaluation uses unseen 1,024-token passages; the 40/80/160-update results come from the same trajectory. The learned-initialization baseline shares the objective, basis, stream, and Direct NTP update. LoCoMo supplies annotated evidence directly.

The Direct NTP controls involve a hyperparameter sweep over learning rates and the number of NTP backward passes (ranging from one to sixteen). To control for the effective update magnitude $n_\theta = [\sum_m \|\Delta B_m A_m\|_{\text{F}}^2]^{1/2}$, the exact-norm Direct NTP baseline fits one positive scale per context; the Module rate oracle fits one signed coefficient per module. These controls and the QA, KL, target-gradient, and decomposition diagnostics are computed during evaluation. Independent VJPs verify the residual–Jacobian identity with relative error below 10^{-2}. Repeated experiments report sample standard deviations where available; training seed counts appear above.

C EXPERIMENTAL SUPPLEMENT

C.1 ACQUISITION CONTROLS AND REPLICATION

All evaluations remove the compiler after adaptation. Table 7 compares hidden-QA NLL on 128 unseen contexts. Best Direct NTP selects among 60 learning-rate and backward-pass configurations, exact-norm Direct NTP matches the Meta-LoRA update norm, and the Module rate control fits one signed Direct NTP scale per module to each Meta-LoRA update.

Table 7: **Acquire controls.** Mean hidden-QA NLL reduction from the pre-update model.

Condition	QA NLL reduction ↑
Meta-LoRA	0.707
Best Direct NTP (60 configurations)	0.337
Exact-norm Direct NTP	0.224
Module rate	0.126

The Module rate control recovers 17.8% of Meta-LoRA's gain and retains 14.2% of its update norm. Meta-LoRA has an excess unrelated-prompt KL of 0.786 relative to Direct NTP. The exact-norm matching has a relative error of 6.31×10^{-7}.

Cross-basis and cross-backbone replication. The shared compiler improves over basis-matched Direct NTP by 0.637 across three meta-training seeds and three unseen writer bases. Its update norm is $111.5\times$ the norm of Direct NTP, motivating the controls above. Table 8 reports independent training on two additional backbones. Both show QA gains over the Module rate controls, with larger update norms and unrelated KL. Each backbone uses its own trained compiler.

Table 8: **Replication across independent backbones.** Reported values are means over repeated runs. QA gain and excess KL are measured relative to Direct NTP. Gain over Module rate denotes the additional QA gain relative to the Module-rate control.

Backbone	QA gain	Gain over Module rate	Norm ratio	Excess KL
Qwen3.5-0.8B	0.578	0.485	86.97	0.177
Llama-3.2-1B	0.135	0.131	76.30	0.050

C.2 GENERATION AND LONG-STREAM EVALUATION

Generation. Across three meta-training seeds and 128 test contexts, Answer+EOS improves content NLL by 0.342 and greedy F1 by 0.091 over Direct NTP. Adding an outer KL weight of 0.1 reduces unrelated KL from 5.820 to 3.540 and retains content and F1 gains of 0.300 and 0.089.

Long streams and transfer. Continual evaluation pairs three compilers with three streams per seed. At 40/80/160 updates, final QA gains over Direct NTP are $0.500/0.405/0.355$. At 160 updates, forgetting decreases by 0.012, but unrelated KL increases by 0.669. The learned-initialization control matches episodes, outer criterion, and budget: its final gain is similar, while its improvement primarily precedes the stream, as shown in Table 3. Meta-LoRA has 0.012 less forgetting and 0.119 less excess KL than learned initialization.

Using LoCoMo's annotated evidence, the frozen SHINE-trained compilers gain 0.414 over Direct NTP after 40 updates and reduce forgetting by 0.066. We evaluate transfer on supplied evidence using two-token teacher-forced QA NLL and forgetting. Retrieval and the official LoCoMo score are outside this protocol.

C.3 C-ONLY BCA WITH UNCHANGED NTP LOSS

For each scalar target, we train three rank-4 C-only BCA compilers from $C = I$, select on development data, and freeze them before evaluation on 128 disjoint contexts with independent writer bases. Each compiler has 196 learned 4×4 matrices, compared with 1,376,256 trainable writer

parameters. The writer starts at zero, and the pre-update scalar NTP loss is numerically identical to that of Direct NTP.

Table 9: **Learned scalar transformations on unseen contexts.** Means across repeated experiments. Normalized error is the squared target-gradient error relative to the corresponding Direct NTP error.

Target	Scale	Projection	Cosine	Norm ratio	Normalized error $E \downarrow$
Half	0.5	0.523	0.995	0.526	0.014
Double	2	1.845	0.994	1.855	0.062
Reverse	-1	-0.921	-0.979	0.941	0.011

These gradients are obtained with a single compiler-on NTP backward pass; the Direct NTP gradient is computed afterward only for measurement. For an equal-norm 90° rotation target, C-only BCA reaches cosine 0.999, target-normalized MSE 0.003, and norm ratio 0.997. Analytic C in Figure 5b is the prescribed block-diagonal matrix with 2×2 rotation blocks $\left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$, not a learned result. In panel (d), prior-write scale s sets $B_{\text{ret}} = sB_{\text{Direct}}$, where B_{Direct} results from one module-normalized Direct NTP write. Normalized gradient error is the squared discrepancy from the learned map applied to the compiler-off gradient at this state, divided by that reference gradient's squared norm; both quantities are summed across writer modules. Applying the learned map to the accumulated writer raises this error to 1.18 and 2.34 at prior-write scales 0.05 and 0.10, respectively.

On a separate disjoint 128-context split, three C-only BCA compilers gain 0.217 QA NLL over Direct NTP, of which 0.063 remains beyond Module rate fitting, with unrelated KL 0.070. Matched Meta-LoRA and Meta-HiRA gain 0.857 and 0.911, with KL 2.007 and 2.211, respectively. C-only BCA's accuracy gain is 0.003, and target-probability change is near zero.

Regularization, Marker, and Damper. For these C-only BCA experiments, the accumulated writer term B_tA is held fixed, and a separate ΔB_t, initialized at zero, is used for the current update. Context NTP is differentiated through $\Delta B_t CA$. All evaluated updates preserve the pre-update scalar NTP value.

Table 10 normalizes target-gradient squared error by the error of Direct NTP; for Marker, classification error is normalized instead. R/R_D is the penalty ratio after adaptation.

Table 10: **C-only BCA with unchanged pre-update NTP loss.** Mean $\pm$ sample SD across repeated experiments on 128 disjoint test contexts.

Task	Train ctxs / outer steps	Normalized error $\downarrow$	Post-adaptation outcome
KL	256/120	0.997 ± 0.001	$R/R_D = 0.812$
L2	1,024/480	0.297 ± 0.006	$R/R_D = 0.313$
Orthogonality	1,024/480	0.649 ± 0.018	$R/R_D = 0.734$
Marker	1,024/600	0.997 ± 0.033	Acc. 0.505 (Direct NTP 0.503)
Damper	256/80	0.583 ± 0.004	Scales $\{0.639, 0.632, 0.629, 0.624\}$

L2 and Orthogonality reduce their normalized target-gradient errors. KL's target-gradient error remains close to that of Direct NTP despite a lower post-adaptation penalty, while Marker remains near chance. Damper produces nearly constant attenuation across the requested $(1, 0.5, 0.1, 0.02)$ schedule. For the Damper comparison, C-only BCA updates only B, whereas Meta-LoRA and Meta-HiRA update both A and B.

D MATCHED COMPILER ARCHITECTURES

Matched protocol. All matched experiments use three random seeds, 960 outer optimization steps, and 1,024/64/128 disjoint train/development/test contexts. Paired methods share the same writer bases and inner-update settings. Meta-HiRA uses an outer learning rate ten times that of Meta-LoRA. Both BCA variants initialize C to the identity and optimize it with a learning rate of 0.003. The Damper task updates both A and B, whereas all other tasks update only B.

Meta-training uses Adam, clips the outer-gradient norm at one, and applies a 10^{-6} penalty to the mean-squared compiler parameters. Scalar targets follow Equation 23. For Reverse, the outer ob-

jective additionally includes

$$10\left(1+\frac{\langle \boldsymbol{g}_\theta, \boldsymbol{g}_{\text{off}}\rangle}{\|\boldsymbol{g}_\theta\|_2\|\boldsymbol{g}_{\text{off}}\|_2}\right). \tag{26}$$

Model checkpoints are selected based on their performance on the development data. Online adaptation uses only context NTP, and all evaluations are performed with the compiler removed. We report mean $\pm$ sample SD across three independent runs, with each run averaged over 128 paired test contexts.

Gradient metrics. Table 11 reports normalized target-gradient error E, where $E = 1$ corresponds to Direct NTP and $E = 0$ to exact target matching. For scalar and regularization tasks,

$$E = \frac{\|\boldsymbol{g}_\theta - \boldsymbol{g}^\star\|_2^2}{\|\boldsymbol{g}_{\text{off}} - \boldsymbol{g}^\star\|_2^2}. \tag{27}$$

Scalar targets are $\boldsymbol{g}^\star = s\boldsymbol{g}_{\text{off}}$, while regularization targets are $\boldsymbol{g}^\star = \boldsymbol{g}_{\text{off}} + \lambda\nabla_\phi R$, evaluated at a shared nonzero writer state.

For Damper,

$$E = \frac{\sum_t w_t e_t}{\sum_t w_t(1-s_t)^2}, \qquad e_t = \frac{\|\boldsymbol{g}_{\theta,t} - s_t\boldsymbol{g}_{\text{off},t}\|_2^2}{\|\boldsymbol{g}_{\text{off},t}\|_2^2}, \tag{28}$$

with target scales $s = (1, 0.5, 0.1, 0.02)$ and weights $w = (16, 1, 4, 8)$. All gradient norms and inner products are computed over the concatenated trainable writer parameters.

Table 11: **Matched gradient errors.** Mean $\pm$ sample SD across runs. Normalized error is 1 for Direct NTP and 0 at the target.

Task	Meta-LoRA	Meta-HiRA	BCA	C-only BCA
Half	0.278 ± 0.003	0.330 ± 0.004	0.033 ± 0.002	$(0.3 \pm 0.1) \times 10^{-3}$
Double	0.519 ± 0.010	0.483 ± 0.006	0.122 ± 0.004	$(3.1 \pm 0.1) \times 10^{-3}$
Reverse	0.231 ± 0.001	0.232 ± 0.002	0.230 ± 0.001	0.014 ± 0.001
L2	0.601 ± 0.009	0.804 ± 0.002	0.262 ± 0.088	0.448 ± 0.018
Orth.	0.937 ± 0.010	0.936 ± 0.011	0.670 ± 0.010	0.769 ± 0.011
Damper	0.732 ± 0.020	0.781 ± 0.019	0.643 ± 0.018	0.711 ± 0.006

Behavior metrics. Table 12 reports model behavior after adaptation. The Reverse projection and Damper scale metrics are computed as

$$\frac{\langle \boldsymbol{g}_\theta, \boldsymbol{g}_{\text{off}}\rangle}{\|\boldsymbol{g}_{\text{off}}\|_2^2}. \tag{29}$$

For Marker, we report balanced accuracy. In Figure 2, its classification error is normalized by the corresponding Direct NTP error:

$$\frac{1 - \text{Acc}_{\text{method}}}{1 - \text{Acc}_{\text{Direct}}}. \tag{30}$$

Acquire QA NLL gain is the QA NLL after Direct NTP minus the QA NLL after the compared method, with both evaluated after compiler removal. QA NLL rebound is the difference between the final QA NLL and the minimum QA NLL observed during adaptation.

Table 12: **Behavior after adaptation.** Mean ± sample SD across runs. Positive QA NLL gain indicates improvement over Direct NTP.

Metric	Meta-LoRA	Meta-HiRA	BCA	C-only BCA
Reverse projection	-0.109 ± 0.014	-0.094 ± 0.010	-0.114 ± 0.017	-0.784 ± 0.010
Marker balanced accuracy	0.872 ± 0.062	0.655 ± 0.050	0.952 ± 0.020	0.521 ± 0.013
Acquire QA NLL gain	0.752 ± 0.037	0.742 ± 0.019	0.791 ± 0.040	0.167 ± 0.011
Acquire answer accuracy	0.424 ± 0.013	0.432 ± 0.008	0.429 ± 0.012	0.392 ± 0.001
Damper first-update scale	0.750 ± 0.017	0.778 ± 0.018	0.636 ± 0.013	0.653 ± 0.016
Damper fourth-update scale	0.595 ± 0.036	0.696 ± 0.047	0.460 ± 0.034	0.661 ± 0.016
Damper QA NLL rebound	0.083 ± 0.004	0.103 ± 0.008	0.070 ± 0.003	0.078 ± 0.006

E LIMITATIONS

Our experiments focus on a limited set of model scales, learning ends, and evaluation settings. Some metrics capture only particular aspects of post-adaptation behavior, and broader evaluation may reveal additional trade-offs across methods. Extending Heterotelic Learning to a wider range of models, tasks, and adaptation settings is an important direction for future work.