COMPILING LEARNING PROBLEMS INTO ADAPTATION PROGRAMS FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Model adaptation is typically governed by a fixed recipe, even though different update programs can produce substantially different behavioral outcomes. We introduce adaptation compilation, which reframes where, how, and to what extent a model should adapt as a joint prediction and decision problem. Rather than searching over candidate programs anew for each learning episode, a compiler learns from prior adaptations to predict a vector-valued counterfactual response surface over candidate programs—their expected effects on acquisition, transfer, boundedness, and preservation—and selects a program before adaptation begins. Because this predicted geometry captures multiple behavioral consequences rather than a single winner or scalar score, it can be reused under different downstream priorities without retraining. Across five learning types, preferred programs vary meaningfully across episodes, and this variation is predictable from pre-adaptation information. On Llama-3.1-8B, compiler-selected programs approach exhaustive search while outperforming global and objective-specific defaults. Replication on Gemma-2-9B preserves program heterogeneity and selection headroom, but shows that exploiting this headroom requires accounting for uncertainty when departing from strong defaults. Together, these results show that adaptation search can be amortized across related learning problems, turning prior adaptation experience into a basis for deciding how future learning should occur.

1 INTRODUCTION

Current parameter-efficient methods begin from the assumption that before learning starts, one must decide where the model is allowed to change. Low-rank adaptation (LoRA) modules are commonly inserted into a fixed set of attention or feed-forward projections, often uniformly across depth. The rank, placement, regularization of these modules may be tuned, but the underlying adaptation architecture is typically specified independently of the learning problem itself. Different forms of learning (such as acquiring a new fact, applying a behavioral rule, planning a multistep procedure, or inferring a causal relationship) may therefore be implemented through the same update mechanism, even though they impose fundamentally different demands on the model.

This assumption has become increasingly difficult to justify. Different forms of learning are not equally well supported by updates to the same regions of a network (Ramnauth & Scassellati, 2026). Some learning problems are acquired most effectively through highly localized updates, while others require broader or differently positioned changes. Moreover, the update location that maximizes immediate acquisition may not be the one that best promotes generalization—or, conversely, best prevents the learned behavior from extending beyond its intended scope.

Recent methods partially relax fixed adaptation architecture in that they dynamically reallocate rank, prune unnecessary parameters, select modules using local sensitivity measures, or route inputs among previously learned adapters. However, these approaches tend to optimize capacity within a predefined adaptation scheme or respond to optimization signals during a single training run. They do not yet treat the adaptation architecture itself as something to be learned. We introduce this gap as **adaptation compilation**. That is, given the nature of an update, what parts of the model should change, what form should those changes take, and how should the resulting updates be constrained?

A compiler receives a learning problem and produces an executable adaptation program. The program specifies where updates should occur, which modules should receive those updates, how much

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.

capacity each update should have, and how strongly unaffected behavior should be preserved. Adaptation is then performed only within the compiled program. Rather than adapting a model through a static configuration and evaluating its consequences afterward, the system predicts how a learning episode is expected to respond to alternative adaptation configurations and selects the configuration expected to produce the intended behavioral profile.

This matters for system reliability because such a compiler is not only efficient—avoiding exhaustive adaptation sweeps or costly diagnostics for every update—but also distinguishes whether an update was acquired, generalized appropriately, remained bounded, and preserved unrelated behavior. A lightweight system could therefore characterize an episode, predict the consequences of candidate programs, and reserve more expensive probing or explicit search for ambiguous cases. We develop this idea through six experiments: first establishing whether meaningful selection headroom exists; then testing whether adaptation outcomes can be predicted from pre-adaptation information and used to select better programs; identifying which episode–model signals matter; and finally evaluating generalization to unseen learning types and replication on a second model backbone.[1]

2 Background and Related Work

Adapting a pretrained model requires two core decisions: (1) how its parameters should change, and (2) which degrees of freedom should be available for adaptation. Our work connects prior research on parameter-efficient design, automated allocation, meta-learning, and task representation by treating adaptation structure as something that can be predicted from a new learning episode.

Parameter-efficient fine-tuning (PEFT) restricts adaptation to a small set of trainable degrees of freedom, including adapters, prompts, selected pretrained parameters, or low-rank updates (Houlsby et al., 2019; Li & Liang, 2021; Lester et al., 2021; Zaken et al., 2022; Hu et al., 2022a). These methods therefore require an adaptation configuration specifying where and how learning may occur. Such configurations are commonly fixed before optimization, even though effective update locations can vary across learning objectives and need not optimize acquisition, transfer, and boundedness simultaneously (Ramnauth & Scassellati, 2026). This motivates treating adaptation structure as part of the learning problem rather than solely as a manually specified hyperparameter.

Several approaches already make PEFT structure adaptive. Dynamic rank allocation, architecture search, and model-derived diagnostics can redistribute capacity or identify promising layers and modules (Zhang et al., 2023; Valipour et al., 2023; Mao et al., 2024; Hu et al., 2022b; Lawton et al., 2023; Zhou et al., 2024; Xu et al., 2026; Saket, 2026; Zhang et al., 2026). These methods establish that adaptation configuration is consequential, but generally optimize structure within the target training problem, search anew for each task, or translate a predefined diagnostic into an allocation rule. Our setting instead asks whether outcomes from *previous* adaptations can be reused to predict the consequences of alternative configurations for a new episode before adaptation begins.

This perspective connects adaptation compilation to meta-learning and per-instance algorithm selection. Meta-learning learns initializations, optimization rules, or adaptable parameter sets across tasks (Finn et al., 2017; Li et al., 2017; Andrychowicz et al., 2016; Ravi & Larochelle, 2017; Von Oswald et al., 2021), while Task2Vec and related approaches represent tasks through their interaction with a model rather than nominal task identity (Achille et al., 2019; Vu et al., 2020; Wang et al., 2021). Classical algorithm selection similarly uses features of a new problem instance to predict which candidate procedure will perform best (Rice, 1976). To our knowledge, these ideas have not been combined to predict the multidimensional behavioral consequences of each candidate adaptation program so that program selection can depend on the behavioral tradeoff of interest rather than directly predicting a single winner.

3 A Framework for Adaptation Compilation

We formulate this missing capability as **adaptation compilation**, which predicts the consequences of alternative update programs for a new learning episode and selects among them before adaptation begins. Ramnauth & Scassellati (2026) introduced *adaptation geometry* and characterized it through

[1] We provide the code, dataset, experimental outputs, and supporting artifacts for reproducibility, along with a Python package exposing adaptation compilation as a reusable interface for future applications and extensions.

empirical search over adaptation configurations; here, we amortize that search by learning to predict geometry from prior adaptation episodes.

Let M_θ denote the pretrained model and D the adaptation examples defining a desired model change. Let $V(M)$ be the set of modules that could be adapted. An adaptation configuration

$$g = \{(v, r_v) : v \in V_g \subseteq V(M)\}$$

specifies the adaptable modules v and their update capacities r_v; all other modules outside V_g are frozen. We restrict candidate programs to those satisfying a cost budget $C(g) \leq C_{\max}$.

Executing program g on episode D produces behavioral outcomes

$$Y_{D,M}(g; \xi) = [A, T, B, P],$$

where ξ captures optimization randomness and A, T, B, and P measure acquisition, transfer, boundedness, and preservation.[2] The expected outcomes across adaptation randomness define the episode's *adaptation geometry*,

$$\Gamma_{D,M}(g) = \mathbb{E}_\xi[Y_{D,M}(g; \xi)].$$

Thus, $\Gamma_{D,M}$ describes how the same learning episode is expected to behave under alternative adaptation programs.

Before adaptation, we characterize the episode and its interaction with the frozen model by

$$s_{D,M} = [e_D, b_{D,M}, q_{D,M}],$$

where e_D represents the adaptation examples, $b_{D,M}$ summarizes the frozen model's behavior on them, and $q_{D,M}$ contains module-level diagnostics. A learned predictor

$$F_\psi(s_{D,M}, g) = \widehat{\Gamma}_{D,M}(g)$$

estimates the behavioral geometry of each candidate program from records collected on previous adaptation episodes.

Given a utility specification τ describing the desired tradeoff among behavioral outcomes, the compiler selects

$$g^* = \arg \max_{g \in \mathcal{G}(M),\, C(g) \leq C_{\max}} U_\tau\left(\widehat{\Gamma}_{D,M}(g)\right).$$

The selected g^* is the *adaptation program*, the executable specification of where and how adaptation will occur. Compilation happens before adaptation, after which the selected program is trained through ordinary parameter-efficient optimization.

4 Experimental Setup

We evaluate adaptation compilation through five questions that follow the logic of the compiler itself: whether alternative programs create meaningful selection headroom (RQ1), whether their behavioral consequences can be predicted before adaptation (RQ2), whether those predictions support better program selection (RQ3), which pre-adaptation signals enable that prediction (RQ4), and whether the learned relationships generalize beyond represented learning families (RQ5). The following setup is shared across the five corresponding experiments (Sec. 5–9).

Learning Episodes. We build on the benchmark introduced by Ramnauth & Scassellati (2026), using the same latent-specification framework, example-generation procedure, and five learning objectives: lexical binding, factual association, behavioral policy learning, causal mapping, and procedural reasoning. Each latent specification defines an episode with an adaptation set and held-out evaluations for the behavioral outcomes defined in Sec. 3; preservation is evaluated on examples unrelated to the target adaptation. Episodes are partitioned at the latent-specification level into 400

[2]Following Ramnauth & Scassellati (2026), *acquisition* measures whether the intended change is learned, *transfer* whether it extends to appropriate held-out contexts, and *boundedness* whether it is withheld where it should not apply. We add *preservation* to measure whether unrelated pre-existing behavior remains unchanged. Cost is treated separately because it describes the adaptation procedure rather than its behavioral consequences.

meta-training, 100 validation, and 100 test episodes, balanced across objectives. Objective identity is not provided to the geometry predictor.

Model backbone. All experiments use Llama-3.1-8B-Instruct (Grattafiori et al., 2024), the primary backbone used by Ramnauth & Scassellati (2026). We retain it for continuity with the benchmark and computational tractability under repeated LoRA interventions.

Configuration space. We instantiate $\mathcal{G}(M)$ using four approximately budget-matched LoRA programs: early-, middle-, and late-depth rank-16 adaptation, and full-stack rank-4 adaptation. Layer position is represented by normalized model depth.

Utility. We use $U_\tau = \mathbf{w}_\tau^\top [A, T, B, P] / \|\mathbf{w}_\tau\|_1$, with $\mathbf{w}_{\text{bal}} = (1, 1, 1, 1)$ and transfer-, boundedness-, and preservation-heavy variants $(1, 2, 1, 1)$, $(1, 1, 2, 1)$, and $(1, 1, 1, 2)$, respectively. Because candidate programs are approximately budget matched, cost does not enter the experimental utility.

Optimization schedule calibration. We calibrate the optimization schedule on meta-training episodes using full-stack rank-16 LoRA as a high-capacity reference, then fix the selected schedule across candidate programs and evaluation splits. Full details are provided in Appendix A.

Adaptation records. Each candidate program is independently executed for every episode and evaluated on acquisition, transfer, boundedness, and preservation. Meta-training and validation episodes use one adaptation seed; test episodes use three, whose outcomes are averaged to estimate expected geometry. Test outcomes are never available before compiler selection.

Episode–model features. Before adaptation, we compute (1) pooled frozen hidden-state representations of the adaptation examples, (2) frozen-model target-token loss statistics, and (3) module-level probes of sensitivity, gradient magnitude, cross-example gradient agreement, and activation magnitude across module types and normalized depth. All features are computed before adaptation, and none requires training a candidate program.

Geometry predictor. We fit a predictor $F_\psi(s_{D,M}, g)$ to estimate acquisition, transfer, boundedness, and preservation for each episode–program pair. The full episode–model representation was prespecified as the primary representation; within it, we compare ridge regression and random-forest regression and select the model and hyperparameters by validation mean absolute error (MAE), averaged across the four behavioral outcomes. Representation ablations are evaluated separately and are not candidates for primary-model selection. The test set is held out until final evaluation.

5 Experiment 1: Is Adaptation Compilation Necessary?

Before attempting to predict adaptation geometry, we first ask whether there is a meaningful program-selection problem to solve (RQ1). We consider two prerequisites for compilation. First, *configuration sensitivity* asks whether alternative adaptation programs produce meaningfully different outcomes for the same episode. Second, *selection headroom* asks whether the preferred program varies sufficiently across episodes to justify episode-conditioned selection. We compare the episode oracle with a *global-fixed* policy learned from meta-training episodes and an *objective-fixed* policy that selects one program per learning objective. The remaining gap between the objective-fixed policy and the oracle therefore measures headroom beyond objective identity alone.

On 100 held-out episodes, the global-fixed policy achieves mean utility 0.569, objective-fixed selection improves this to 0.600, and the episode oracle reaches 0.618 (Fig. 1a). Objective identity therefore explains substantial structure, but does not eliminate the selection problem. Specifically, objective-fixed selection remains oracle-optimal on only 64% of episodes, compared with 41% for the global policy. Moreover, 99% of episodes have a unique utility-maximizing program, and all four candidate programs are oracle-optimal for substantial subsets of episodes.

The remaining headroom is heterogeneous across learning objectives. It is largest for lexical binding and factual association, smaller for causal and behavioral learning, and nearly absent for procedural reasoning (Fig. 1b). Thus, some learning regimes admit strong structural defaults, whereas others retain meaningful episode-level variation in which program best balances acquisition, transfer, boundedness, and preservation. Additional analyses are provided in Appendix B.

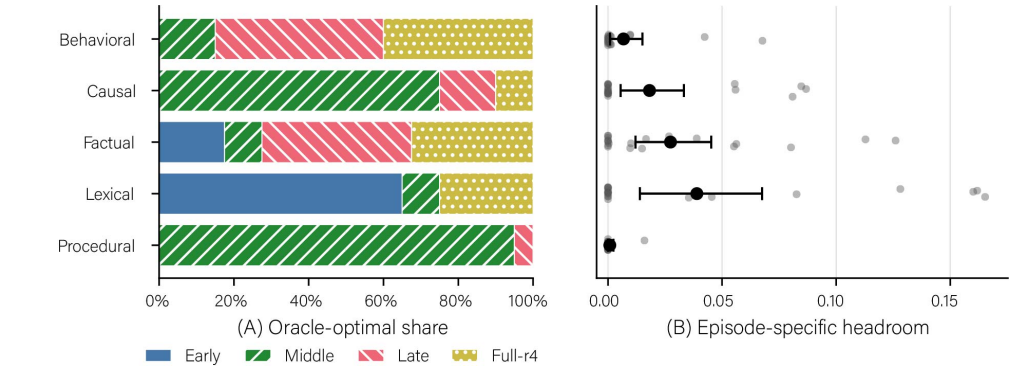


Figure 1: **Selection headroom in adaptation geometry.** **(A)** Fraction of held-out episodes for which each adaptation program is oracle-optimal, shown by learning objective. **(B)** Episode-specific headroom beyond the objective-fixed policy, measured as oracle utility minus objective-fixed utility. Gray points denote episodes; black points and error bars show the mean and 95% CI. Headroom is largest for lexical binding and factual association and nearly absent for procedural reasoning.

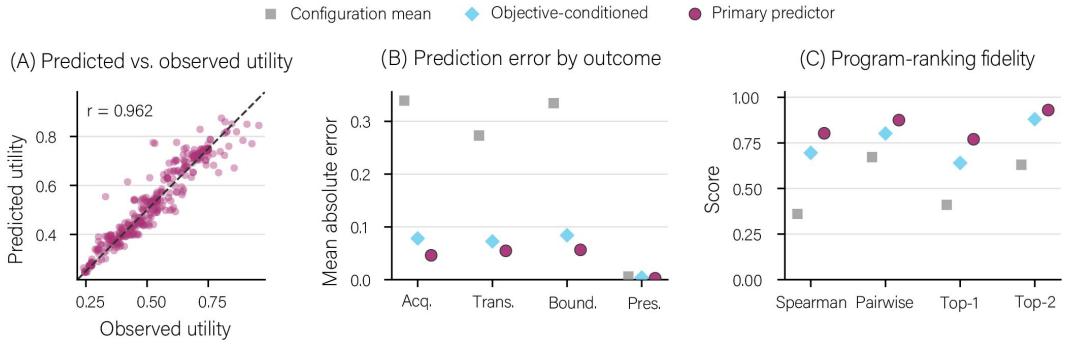


Figure 2: **Predicting adaptation geometry from pre-adaptation episode–model signals.** **(A)** Predicted and observed balanced utility across held-out episode–program pairs. **(B)** Mean absolute error for acquisition, transfer, boundedness, and preservation under the primary predictor and non-episode-conditioned baselines. **(C)** Within-episode program-ranking fidelity measured by Spearman correlation, pairwise ranking accuracy, and tie-aware top-1 and top-2 oracle recovery.

6 Experiment 2: Can Adaptation Geometry Be Predicted?

Having established meaningful selection headroom, we next ask whether adaptation geometry can be predicted before adaptation (RQ2). For each held-out episode–program pair, the predictor estimates acquisition, transfer, boundedness, and preservation from pre-adaptation information. The full representation uses a random forest, selected over ridge regression by validation MAE (0.044 vs. 0.099). We compare against a configuration-mean baseline and, diagnostically, an objective-conditioned mean predictor given the true learning objective.

On held-out episodes, the primary predictor achieves geometry MAE of 0.040, compared with 0.238 for the configuration-mean baseline and 0.060 for the objective-conditioned diagnostic (Fig. 2). Predicted and observed program utilities are also strongly aligned ($r = 0.962$). More importantly for compilation, the predictor preserves within-episode program ordering, reaching Spearman correlation 0.803, pairwise ranking accuracy 87.5%, and top-1/top-2 oracle recovery of 77%/93%. These results show that pre-adaptation episode–model signals contain information beyond objective identity about how alternative programs will behave, including the relative ordering needed for selection. Additional per-objective analyses are provided in Appendix C.

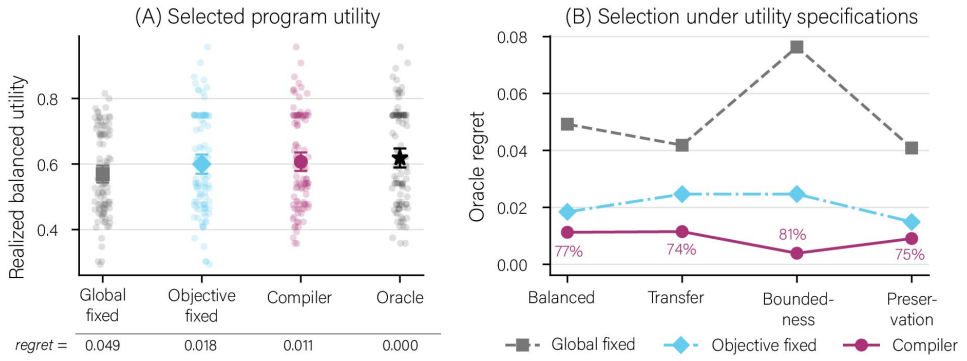


Figure 3: **Program selection from predicted adaptation geometry. (A)** Realized balanced utility for global-fixed, objective-fixed, compiler, and oracle selection; values below the axis show mean oracle regret. **(B)** Oracle regret under alternative utility specifications, using the same predicted geometry without retraining. Percentages show tie-aware top-1 oracle recovery.

7 Experiment 3: Does Prediction Yield Better Programs?

We next ask whether predicted geometry supports better program selection (RQ3). Under balanced utility, the objective-fixed policy already captures much of the available selection headroom (Fig. 3a), achieving 0.600 mean utility versus 0.618 for the exhaustive oracle. The compiler reaches 0.607, reducing oracle regret from 0.018 to 0.011 and recovering roughly 39% of the remaining gap. Across 100 held-out episodes, compilation improves over objective-fixed selection on 22 episodes, degrades performance on 7, and yields identical realized utility on 71 (mean paired gain $= 0.0072$, 95% bootstrap CI $[0.0011, 0.0136]$). Compiler regret remains below 0.006 for four of five learning objectives, with lexical binding the clearest remaining failure mode (Appendix D).

Because the predictor estimates behavioral outcomes rather than a single winner, the same predicted geometry can be recompiled under different priorities without retraining. The compiler remains closer to the oracle than either fixed policy under transfer-, boundedness-, and preservation-heavy utilities (Fig. 3b), showing that predicted geometry can support different adaptation objectives after prediction. Additional per-objective and utility-specific results are provided in Appendix D.

8 Experiment 4: Which Pre-Adaptation Signals are Needed?

We next ask how much pre-adaptation information is needed to predict geometry and support program selection (RQ4). Our primary representation combines adaptation-example representations, frozen-model behavioral statistics, and module-level probes. Surprisingly, the episode-only representation performs comparably to the full representation, achieving geometry MAE of 0.0434 versus 0.0435, with within-episode Spearman correlation of 0.848 and top-1 oracle recovery of 83%.

Module probes and frozen-model behavioral statistics are weaker independently, with geometry MAE of 0.053 and 0.054, respectively. These results suggest that, in the present setting, most of the signal needed for compilation is already available in frozen representations of the learning episode. Explicit gradient and behavioral diagnostics therefore provide limited additional benefit, indicating that useful compilation may require less pre-adaptation computation than expected (Fig. 4a). Full prediction and selection results are provided in Appendix E.

9 Experiment 5: Does Adaptation Compilation Generalize?

Experiments 2–4 evaluate unseen episodes drawn from learning families represented during meta-training. We finally ask whether the learned relationship between episode representations and adaptation geometry transfers to an entirely unseen learning family (RQ5). In leave-one-family-out evaluation, one objective is excluded entirely from both training and validation, and the resulting predictor is evaluated on test episodes from that unseen family.

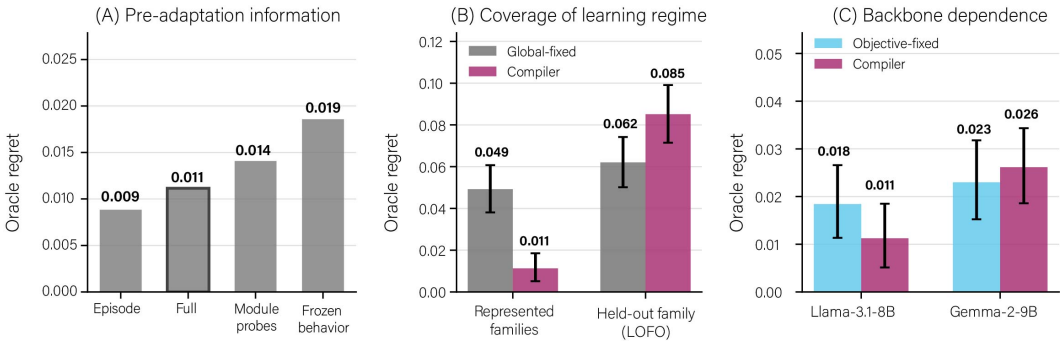


Figure 4: **Scope and boundary conditions of adaptation compilation.** **(A)** Episode representations alone support selection comparable to the full representation, while module probes and frozen-behavior features are weaker in isolation; the outlined bar marks the primary full representation. **(B)** Compilation reduces oracle regret when learning families are represented during meta-training, but this advantage reverses when an entire family is withheld (LOFO). **(C)** Compilation improves over objective-fixed defaults on Llama, whereas on Gemma direct selection from predicted geometry does not improve over the strong objective-fixed policy despite remaining episode-level headroom. Error bars show 95% episode-bootstrap CIs where applicable. Lower oracle regret is better.

Zero-shot family transfer is substantially more difficult than prediction for unseen episodes from represented objectives. Mean geometry MAE rises to 0.339, within-episode Spearman correlation falls to 0.058, and top-1 oracle recovery drops to 14%. Correspondingly, compiler utility falls to 0.533, below the global-fixed policy at 0.556 (Fig. 4b). Performance is heterogeneous across families, but the overall result is that strong generalization to unseen episodes within represented learning regimes does not imply zero-shot transfer to entirely unseen forms of learning. Additional per-family results are provided in Appendix F.

These results establish an important boundary on the present form of adaptation compilation. The predictor generalizes well to new episodes within learning families represented during meta-training, but does not reliably extrapolate adaptation geometry to qualitatively unseen learning objectives. The learned mapping therefore captures structure that is more general than individual latent specifications, but remains dependent on coverage of the underlying learning regime.

10 Experiment 6: Does the Case for Compilation Reproduce Across Backbones?

Experiments 1–5 establish adaptation compilation on Llama-3.1-8B-Instruct, but adaptation geometry is inherently model-dependent. Ramnauth & Scassellati (2026) finds broadly consistent localization signatures across models, alongside substantial model-specific variation and particular sensitivity in Gemma to Llama-calibrated adaptation budgets. We therefore repeat the compiler pipeline on Gemma-2-9B-IT, independently recalibrating the optimization schedule while holding the program library and evaluation protocol fixed. The predictor is trained only on Gemma adaptation records, testing replication of the compilation principle rather than zero-shot transfer from Llama.

On Gemma, the best fixed program is full-stack. Objective-specific defaults select full-stack for behavioral, factual, and procedural learning, middle for causal mapping, and late for lexical binding; only the causal default matches Llama. Yet no program dominates held-out episodes: across 100 test episodes, oracle selections are 40% full, 31% middle, 23% late, and 6% early. The best fixed program achieves .436 mean utility, versus .461 for the episode-wise oracle; objective-specific defaults reach .438 but retain .023 mean regret. Thus, adaptation geometry changes across backbones, but substantial episode-level selection headroom remains even after conditioning on learning type.

Recovering this headroom is more difficult for Gemma. The learned compiler achieves .435 mean utility versus .461 for the oracle, with .026 regret, 42% oracle recovery, and 73% top-2 recovery. Its aggregate performance is comparable to, but does not exceed, the global-fixed (.436) or objective-fixed (.438) baselines (Fig. 4c). The shortfall is concentrated, not uniform; the compiler largely

recovers strong defaults for behavioral, causal, and factual episodes, with most loss arising in procedural reasoning and a smaller deficit in lexical binding. In these cases, small predicted differences between candidate programs can trigger a switch even when the realized advantage is uncertain.

Gemma sharpens rather than weakens the case for compilation. Program heterogeneity and episode-level oracle headroom persist across backbones, but exploiting that headroom requires knowing when a predicted advantage over a strong default is reliable. On Llama, direct selection from predicted geometry improves over fixed policies; on Gemma, the same rule sometimes acts on small, noisy margins. This suggests an additional design requirement for adaptation compilers. Program selection should account not only for predicted geometry, but also for uncertainty in the decision to depart from a robust default.

11 Discussion

Adaptation compilation begins from a methodological mismatch: structural choices—where to update, which modules to expose, and how much capacity to allocate—are typically fixed before knowing how a given learning episode will respond. Prior work shows that these choices induce distinct behavioral profiles across learning objectives (Ramnauth & Scassellati, 2026), but identifying those profiles ordinarily requires running the very adaptations one hopes to choose among. We ask whether this search can instead be amortized across prior adaptation experience.

Our results show that it can, but also clarify the conditions under which doing so is useful. On Llama, approximately budget-matched programs produce meaningfully different outcomes across episodes, those differences are predictable before adaptation, and selecting from predicted geometry substantially closes the gap to exhaustive search while outperforming global and objective-specific defaults. The same predictions can also be reused under different behavioral priorities because the compiler estimates acquisition, transfer, boundedness, and preservation rather than directly predicting a single winning program. Experiment 6 shows that the case for compilation reproduces on Gemma: adaptation geometry changes across backbones, yet substantial episode-level selection headroom remains. At the same time, direct selection from predicted geometry is less reliable on Gemma, revealing that headroom alone is insufficient; a useful compiler must also know when a predicted advantage over a strong default is trustworthy.

11.1 Adaptation as a Prediction and Decision Problem

These results suggest treating the adaptation procedure itself as part of the learning problem. Rather than applying one fixed fine-tuning recipe, adaptation geometry makes the update program a decision variable because alternative programs induce different tradeoffs among acquisition, transfer, boundedness, and preservation. The compiler separates two questions that are often conflated: *what will happen if the model is adapted this way?* and *which of those outcomes is preferred?*

This separation is useful because there need not be a universally best program. Predicting the full geometry preserves behavioral tradeoffs and allows the same predictions to be re-evaluated under different utilities without retraining. It also changes where search cost is paid. Exhaustive configuration search repeats multiple adaptations for every new episode, whereas compilation learns from prior interventions and reuses that experience. Constructing the initial geometry dataset remains expensive, but future adaptation decisions can be made from pre-adaptation information before executing only the selected program.

11.2 What Determines an Adaptation Program?

The experiments suggest that adaptation structure is neither purely global nor reducible to learning-objective identity. Objective-conditioned defaults are strong, but oracle programs continue to vary within objectives, and episode-conditioned prediction improves beyond those defaults on Llama. The relevant unit for adaptation therefore lies between a universal fine-tuning recipe and a categorical task-level rule.

Experiment 4 suggests that adaptation compilation may be cheaper than its most general formulation implies. Frozen representations of the learning episode alone recover nearly all of the predictive performance of the full episode–model representation, despite omitting behavioral statistics and

module-level probes. In this setting, explicit diagnostics therefore appear better suited for uncertain or difficult cases than as a prerequisite for every compilation decision.

Experiment 6 adds a second source of variation: the mapping from learning problems to useful programs is model-dependent. Gemma retains clear episode-level heterogeneity even though its objective-level program preferences differ substantially from Llama. Adaptation geometry should therefore be understood as a property of the interaction among episode, model, and program rather than as a fixed localization map. Accordingly, these results should not be interpreted as evidence of strict modularity; depth and module placement are functional intervention variables, not canonical locations where particular forms of knowledge reside.

11.3 When Should a Compiler Trust Its Prediction?

The family- and backbone-generalization experiments expose an important boundary on the current formulation. In leave-one-family-out evaluation, performance degrades when an entire form of learning is absent from meta-training: geometry error increases, program rankings become less reliable, and compilation can underperform a fixed structural policy. The compiler therefore learns regularities over adaptation problems represented in its experience rather than a universal mapping from arbitrary learning problems to update programs.

Gemma reveals that, even when the learning families are represented and episode-level headroom remains, small errors in predicted program differences can make a deterministic argmax selector worse than a strong default. This effect is not uniform—most of the Gemma shortfall arises from lexical and especially procedural episodes, where small predicted advantages trigger program switches that do not consistently improve realized utility. Thus, successful compilation requires calibrated confidence in whether the predicted difference between competing programs is large enough to act on.

This suggests a natural extension from *geometry prediction* to *confidence-aware compilation*. A compiler could retain a robust global or objective-specific default when candidate programs are predicted to be nearly equivalent, and deviate only when the expected improvement is sufficiently large or well supported. Out-of-family detection, predictive uncertainty, and targeted empirical search could serve the same purpose when the compiler encounters unfamiliar adaptation problems. We do not introduce such a rule post hoc here; rather, the Gemma results identify it as an additional design requirement for future compilers.

11.4 Practical Scope and Limitations

Adaptation compilation is most attractive in settings involving many repeated but heterogeneous updates, where exhaustive per-episode tuning is impractical but repeatedly applying a poorly matched strategy can accumulate substantial cost. A system could maintain a library of feasible adaptation programs, characterize a new episode before training, and either execute the predicted program or fall back to a structural default or targeted search when confidence is low. Our representation ablations suggest that this decision does not always require expensive model diagnostics; richer probes could instead be reserved for uncertain cases.

The present study nevertheless uses a small and structured program space (four LoRA configurations across five controlled learning objectives). Real adaptation spaces may include finer-grained layer choices, heterogeneous rank allocation, optimizer settings, multiple adaptation mechanisms, and sequential or mixed objectives. Compilation also amortizes search rather than eliminating it: supervision still requires executing candidate programs on prior episodes, and adaptation outcomes remain stochastic. Scaling this approach will then require both richer program representations and more selective acquisition of adaptation experience.

Our broader conclusion is that adaptation itself can become learnable. Prior adaptation episodes contain information about how future learning should be carried out. Our results show that this information can support effective program selection within represented regimes, while the Gemma and leave-one-family-out experiments clarify two important limits: adaptation geometry is model-dependent, and predicted advantages must be reliable enough to justify departing from strong defaults. These boundaries turn adaptation compilation from a fixed recipe into a conditional decision problem—one that must reason not only about which program appears best, but also when that prediction is worth trusting.

AI USE STATEMENT

The conception of this work, including the research questions, hypotheses, methodological design, experimental planning, implementation decisions, execution of experiments, analysis, interpretation of results, and scientific conclusions, was carried out by the authors without the use of generative AI. Generative AI tools were used only as software-engineering aids to refactor portions of the existing codebase and to assist in generating and improving code documentation. All AI-assisted code changes and documentation were manually reviewed, verified against the intended functionality, and corrected where necessary by the authors.

ETHICS STATEMENT

This work studies methods for selecting adaptation programs for large language models based on prior adaptation experience. The experiments do not involve human subjects or the collection of personal or sensitive data. The proposed framework is intended as a methodological tool for understanding and improving how models are adapted to new learning objectives. Like other methods that improve the efficiency or precision of model adaptation, adaptation compilation is potentially dual-use: the same mechanisms that enable more targeted acquisition and preservation of desired behaviors could, in principle, be applied toward undesirable objectives. Our method does not provide guarantees regarding the safety, fairness, or downstream behavior of an adapted model, and the controlled evaluation objectives studied here should not be interpreted as such guarantees. Deployment in consequential settings would therefore require application-specific evaluation, appropriate safeguards, and consideration of the properties and limitations of the underlying model.

REPRODUCIBILITY STATEMENT

We provide the materials needed to reproduce the experimental pipeline and reported analyses. Section 4 specifies the learning episodes, data splits, adaptation-program space, utility functions, optimization calibration, adaptation records, pre-adaptation features, and predictor-selection protocol; Sections 5–10 define the evaluation procedures for each experiment. Appendices A–G provide additional calibration results, analyses of selection headroom and adaptation stochasticity, geometry-prediction and program-selection results, representation ablations, leave-one-family-out evaluation, and the complete Gemma replication protocol. The supplementary materials include the code, dataset, experiment outputs, configuration files, and a reusable Python implementation of adaptation compilation. Model and hyperparameter selection use only meta-training and validation data, and all reported test results use fixed held-out episodes and the adaptation seeds specified in the experimental protocol.

AUTHOR CONTRIBUTIONS

Omitted for anonymous submission and will be included in the final version.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless Fowlkes, Stefano Soatto, and Pietro Perona. Task2Vec: Task embedding for meta-learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6429–6438. IEEE, 2019.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

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

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

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

Shengding Hu, Zhen Zhang, Ning Ding, Yadao Wang, Yasheng Wang, Zhiyuan Liu, and Maosong Sun. Sparse structure search for parameter-efficient tuning. *arXiv preprint arXiv:2206.07382*, 2022b.

Neal Lawton, Anoop Kumar, Govind Thattai, Aram Galstyan, and Greg Ver Steeg. Neural architecture search for parameter-efficient fine-tuning of large pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 8506–8515, 2023.

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

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, pp. 4582–4597, 2021.

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.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. DoRA: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11662–11675, 2024.

Rebecca Ramnauth and Brian Scassellati. Localized adaptation reveals distinct learning signatures in transformers. *arXiv preprint arXiv:2607.25663*, 2026.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2017.

John R Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pp. 65–118. Elsevier, 1976.

Abdulmalek Saket. Aletheia: Gradient-guided layer selection for efficient lora fine-tuning across architectures. *arXiv preprint arXiv:2604.15351*, 2026.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. DyLoRA: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, 2023.

Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34:5250–5263, 2021.

Tu Vu, Tong Wang, Tsendsuren Munkhdalai, Alessandro Sordoni, Adam Trischler, Andrew Mattarella-Micke, Subhransu Maji, and Mohit Iyyer. Exploring and predicting transferability across NLP tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7882–7926, 2020.

Jixuan Wang, Kuan-Chieh Wang, Frank Rudzicz, and Michael Brudno. Grad2Task: Improved few-shot text classification using gradients for task representation. *Advances in Neural Information Processing Systems*, 34:6542–6554, 2021.

Yichen Xu, Yuyang Liang, Shan Dai, Tianyang Hu, Tsz Nam Chan, and Chenhao Ma. Understanding and guiding layer placement in parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2602.04019*, 2026.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

Suoxin Zhang, Run He, Di Fang, Xiang Tan, Kaixuan Chen, and Huiping Zhuang. Rethinking adapter placement: A dominant adaptation module perspective. *arXiv preprint arXiv:2605.06183*, 2026.

Han Zhou, Xingchen Wan, Ivan Vulić, and Anna Korhonen. AutoPEFT: Automatic configuration search for parameter-efficient fine-tuning. *Transactions of the Association for Computational Linguistics*, 12:525–542, 2024.

APPENDICES

The appendices provide additional methodological detail and analyses supporting the main experiments. We first document optimization calibration (Appendix A), then expand the Llama results with analyses of selection headroom and stochasticity, geometry prediction, program selection, and representation ablations (Appendices B–E). We next examine generalization to unseen learning families (Appendix F) and conclude with the full Gemma replication protocol and results (Appendix G).

A OPTIMIZATION SCHEDULE CALIBRATION

Because each adaptation episode contains a single latent specification, we calibrate the optimization schedule separately for this single-specification regime before comparing adaptation configurations. The purpose of this calibration is to ensure that subsequent differences in adaptation geometry are not artifacts of systematic undertraining.

We use five meta-training episodes from each of the five learning objectives (25 episodes total) and adapt each episode using the unconstrained full-stack LoRA configuration. Holding the remaining optimization settings fixed, we vary gradient accumulation over $\{1, 2, 4, 8\}$, thereby varying the number of optimizer updates available during adaptation. No validation or test episodes are used for calibration.

Table 1: Optimization calibration on meta-training episodes. Values are averaged across 25 episodes.

Gradient Accum.	Acquisition	Transfer	Boundedness	Preservation
8	0.408	0.240	0.393	0.997
4	0.648	0.587	0.500	0.992
2	0.896	**0.867**	0.547	0.971
1	**0.904**	0.853	**0.700**	0.876

Larger accumulation values leave several learning objectives near the acquisition or transfer floor. Reducing gradient accumulation to 2 substantially increases both acquisition and transfer across objectives. A further reduction to 1 yields negligible additional acquisition ($0.896 \rightarrow 0.904$) and slightly lower transfer ($0.867 \rightarrow 0.853$), while preservation decreases substantially ($0.971 \rightarrow 0.876$). We therefore select gradient accumulation 2 as the least aggressive schedule that reliably supports learning across objectives without unnecessary collateral interference. This optimization schedule is fixed for all subsequent configurations, episodes, and evaluation splits.

B EXPERIMENT 1: CHARACTERIZING SELECTION HEADROOM

The main text establishes that alternative adaptation programs induce meaningful selection headroom on held-out episodes. Here, we provide the full objective-level results, characterize the separation between oracle-optimal programs, and examine robustness to adaptation stochasticity. All headline test results use the same 100 held-out episodes reported in the main text, with each configuration's behavioral outcomes averaged across three adaptation seeds.

B.1 SELECTION HEADROOM BY LEARNING OBJECTIVE

Table 2 reports the full selection results by learning objective. The global-fixed program is selected using all meta-training episodes, while the objective-fixed policy selects one program per learning objective using only meta-training episodes. The episode oracle selects the highest-utility program independently for each held-out episode.

Table 2: **Selection headroom by learning objective.** Utilities are measured on held-out test episodes. Global- and objective-fixed programs are selected using meta-training data only. Regret is measured relative to the per-episode oracle, and "Opt." denotes the fraction of episodes on which the corresponding fixed policy is oracle-optimal.

Objective	Objective-fixed	Global U	Obj. U	Oracle U	Global Regret	Obj. Regret	Global Opt.	Obj. Opt.
Overall	—	.569	.600	.618	.049	.018	.41	.64
Behavioral	Late	.714	.747	.753	.039	.007	.15	.45
Causal	Middle	.489	.489	.507	.018	.018	.75	.75
Factual	Late	.504	.564	.591	.087	.027	.10	.40
Lexical	Early	.688	.751	.790	.102	.039	.10	.65
Procedural	Middle	.449	.449	.450	.001	.001	.95	.95

Objective conditioning accounts for a substantial portion of program preference, reducing mean regret from 0.049 under the global-fixed policy to 0.018. However, the magnitude of the remaining episode-specific headroom differs considerably across learning objectives. Lexical binding and factual association retain the largest objective-to-oracle gaps, whereas procedural reasoning admits a nearly universal middle-depth default. In the latter case, the objective-fixed policy is oracle-optimal on 95% of held-out episodes and differs from the oracle by less than 0.001 utility on average.

B.2 ORACLE PROGRAMS AND WINNER SEPARATION

Table 3 gives the distribution of oracle-optimal programs underlying Fig. 1 in the main text. We assign fractional credit when multiple configurations attain the same maximum utility. Across all 100 test episodes, middle-depth adaptation is oracle-optimal most frequently, but every candidate program is preferred for a nontrivial subset of episodes.

Table 3: **Oracle-program distribution and winner separation.** Program columns report the fraction of held-out episodes for which each configuration is oracle-optimal, using fractional credit for ties. The final columns report the tie rate and the mean and median utility margin between the highest- and second-highest-utility configurations.

Objective	Early	Middle	Late	Full-r4	Tie Rate	Mean Margin	Median Margin
Overall	.165	.410	.210	.215	.01	.054	.041
Behavioral	.000	.150	.450	.400	.00	.016	.007
Causal	.000	.750	.150	.100	.00	.072	.059
Factual	.175	.100	.400	.325	.05	.045	.023
Lexical	.650	.100	.000	.250	.00	.077	.076
Procedural	.000	.950	.050	.000	.00	.059	.053

The variation in oracle programs is not primarily driven by ties. Only 1% of test episodes contain multiple utility-maximizing configurations, and 99% therefore have a unique winner. The mean difference between the best and second-best configurations is 0.054 utility (median 0.041). Fig. 5 shows these margins at the episode level.

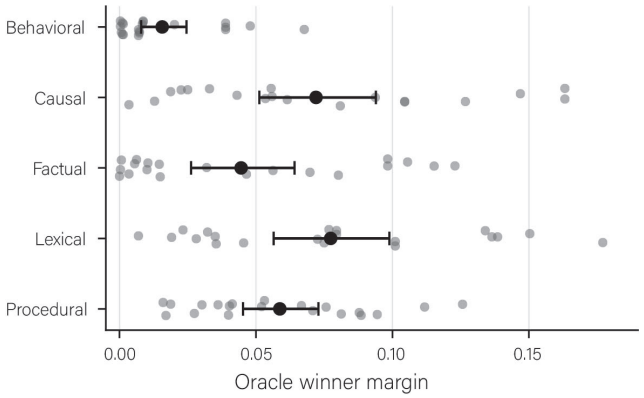


Figure 5: **Separation between oracle-optimal and second-best programs.** Points show the utility margin between the best and second-best adaptation programs for individual held-out episodes, grouped by learning objective. Summary markers indicate the objective-level mean.

B.3 Robustness to Adaptation Stochasticity

Adaptation outcomes also vary across optimization runs. To characterize this variability directly, we analyze a stratified subset of 20 held-out episodes for which all four candidate programs were independently adapted under three random seeds. Our headline test geometry averages three seeds for all 100 held-out episodes. Here, for each episode–configuration pair, we measure the variation in utility across seeds and examine whether the identity of the oracle-optimal program remains stable (Fig. 6).

Adaptation stochasticity is non-negligible: the mean within-configuration standard deviation in utility is 0.028, and only 40% of episodes retain the same unique oracle winner across all three individual seeds. A less restrictive, tie-aware criterion finds at least one common oracle configuration across all seeds for 55% of episodes. These results motivate defining adaptation geometry in terms of expected behavioral outcomes rather than the realization of a single optimization run.

Importantly, selection headroom persists after averaging over this stochasticity. On the robustness subset, the global-fixed, objective-fixed, and episode-oracle utilities are 0.581, 0.605, and 0.621, respectively. The resulting objective-to-oracle regret remains 0.016, and the objective-fixed policy is oracle-optimal on only 60% of episodes.

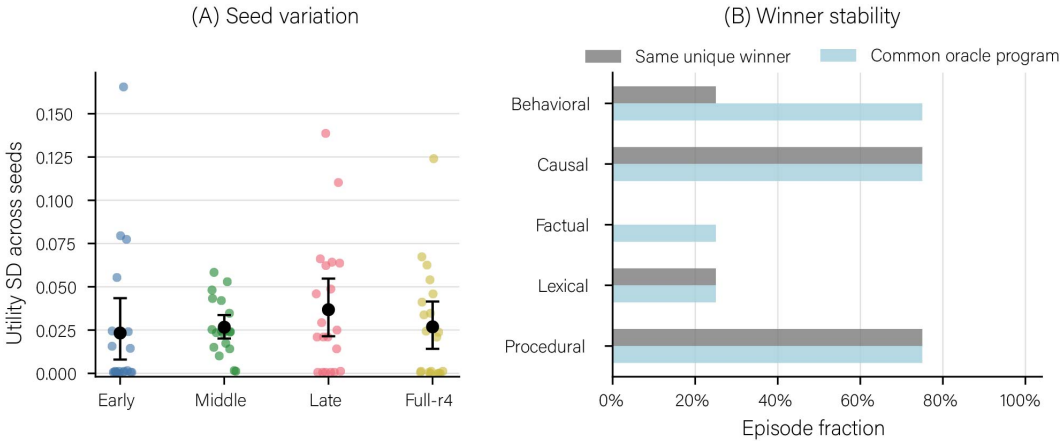


Figure 6: **Adaptation stochasticity across programs. (A)** Variation in utility across three adaptation seeds for each episode–program pair. **(B)** Stability of the oracle-optimal program across seeds, using both strict unique-winner and tie-aware criteria.

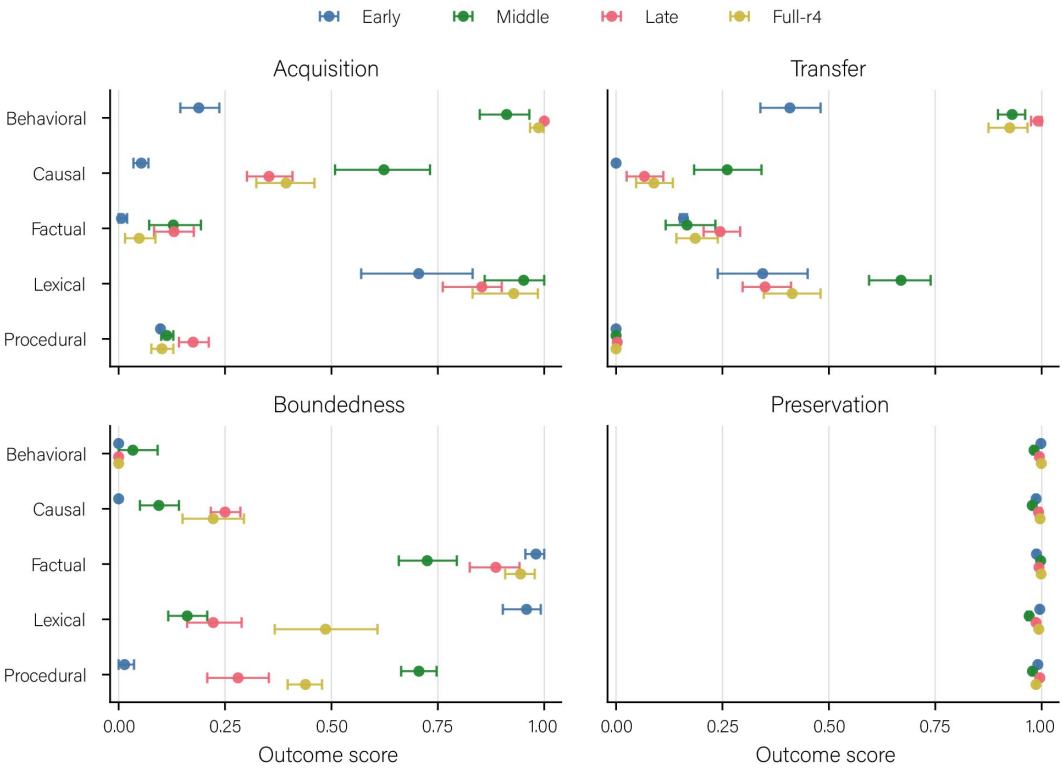


Figure 7: **Configuration sensitivity across adaptation outcomes.** Acquisition, transfer, boundedness, and preservation under each budget-matched adaptation program, grouped by learning objective. Values summarize held-out test episodes using seed-averaged outcomes.

B.4 Configuration Sensitivity Across Behavioral Outcomes

Selection utility compresses four distinct behavioral criteria into a single scalar. To verify that program sensitivity is not restricted to aggregate utility, Fig. 7 decomposes adaptation outcomes into acquisition, transfer, boundedness, and preservation. Alternative programs induce distinct behavioral tradeoffs across these dimensions, providing the outcome-level variation from which the selection differences reported in the main text arise.

C Experiment 2: Predicting Adaptation Geometry

We select the predictor using validation mean absolute error averaged across acquisition, transfer, boundedness, and preservation. The candidate models are ridge regression and random-forest regression; validation selects the random forest (0.044 MAE, compared with 0.099 for ridge). The selected predictor uses 300 trees, unrestricted depth, a minimum leaf size of one, and random seed 2026. Model selection and hyperparameter choice use only meta-training and validation episodes.

The aggregate prediction results in Sec. 6 obscure substantial variation across learning objectives. Table 4, depicted as Fig. 8, therefore evaluates the same frozen geometry predictor separately within each objective. Prediction error is lowest for behavioral episodes and remains relatively low for causal, factual, and procedural episodes, while lexical binding is markedly more variable and contains the largest-error episodes. The same distinction appears in program-ranking fidelity: predicted geometry preserves program order particularly well for behavioral, causal, and procedural episodes, whereas factual and lexical episodes are more difficult to rank. Thus, the aggregate performance of the predictor is not attributable to a single learning family, although the reliability of episode-level geometry prediction differs across objectives.

Table 4: **Geometry prediction by learning objective.** Geometry MAE is averaged across acquisition, transfer, boundedness, and preservation. Spearman correlation and pairwise accuracy measure agreement between predicted and observed program rankings within each episode. The same primary predictor is evaluated across all objectives without access to objective identity.

Objective	Geometry MAE	Acquisition	Transfer	Boundedness	Preservation	Spearman	Pairwise
Behavioral	.015	.023	.030	.007	.002	.900	.917
Causal	.033	.050	.051	.030	.002	.940	.958
Factual	.041	.045	.053	.061	.003	.603	.758
Lexical	.086	.090	.138	.112	.004	.630	.783
Procedural	.025	.022	.001	.073	.002	.940	.958

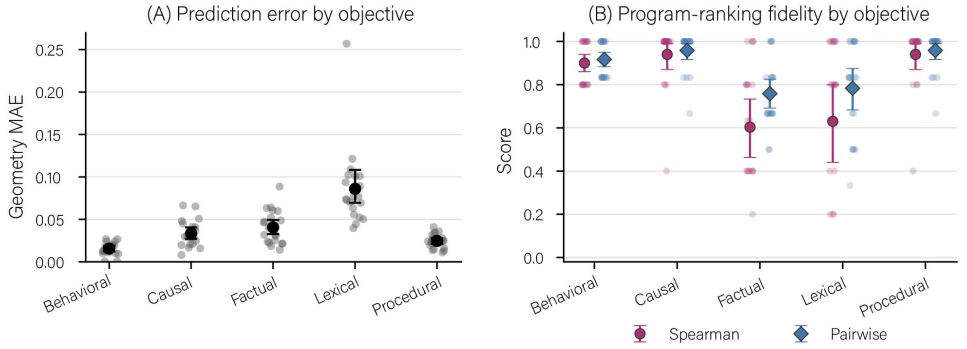


Figure 8: **Geometry prediction across learning objectives. (A)** Episode-level mean absolute error between predicted and observed adaptation geometry, grouped by learning objective. Gray points denote held-out episodes and black markers indicate means with 95% bootstrap confidence intervals. **(B)** Within-episode agreement between predicted and observed program orderings, measured by Spearman rank correlation and pairwise ranking accuracy. Points denote individual episodes and summary markers indicate means with 95% bootstrap confidence intervals. The same predictor is evaluated across all objectives without access to objective identity.

D Experiment 3: Evaluating Program Selection

These analyses clarify where compilation improves on structural defaults and how reusable the predicted geometry remains under changing behavioral priorities.

D.1 Selection by learning objective

Table 5, depicted as Fig. 9, decomposes balanced-utility selection performance by learning objective. The compiler substantially reduces oracle regret relative to the global-fixed policy across behavioral, causal, factual, and lexical episodes, and approaches the oracle particularly closely for behavioral and causal learning. Procedural reasoning already admits a strong fixed structural default, leaving little headroom for episode-conditioned selection.

Lexical binding is the clearest exception to the general improvement over the objective-fixed policy. Although the compiler selects an oracle-optimal program on 65% of lexical episodes, matching the objective-fixed policy, its mean realized utility is slightly lower (0.746 versus 0.751) because its mistakes occur on higher-regret episodes. This is consistent with the greater lexical prediction error observed in Appendix C. By contrast, factual association retains substantial objective-level headroom and benefits strongly from episode-conditioned selection, with regret decreasing from 0.027 to 0.006.

D.2 Paired comparison with objective-fixed selection

Across all 100 held-out episodes, the compiler improves realized utility over the objective-fixed policy on 22 episodes, performs worse on 7, and selects a program with identical realized utility on

Table 5: **Program selection by learning objective.** Realized utility and oracle regret are reported for the global-fixed, objective-fixed, and compiler policies. Top-1 denotes the fraction of held-out episodes for which each policy selects an oracle-optimal program.

	Realized Utility			Oracle Regret			Top-1		
Objective	Global	Obj.	Compiler	Global	Obj.	Compiler	Global	Obj.	Compiler
Behavioral	.714	.747	.753	.039	.007	.001	.15	.45	.70
Causal	.489	.489	.504	.018	.018	.004	.75	.75	.90
Factual	.504	.564	.585	.087	.027	.006	.10	.40	.65
Lexical	.688	.751	.746	.102	.039	.044	.10	.65	.65
Procedural	.449	.449	.448	.001	.001	.002	.95	.95	.95

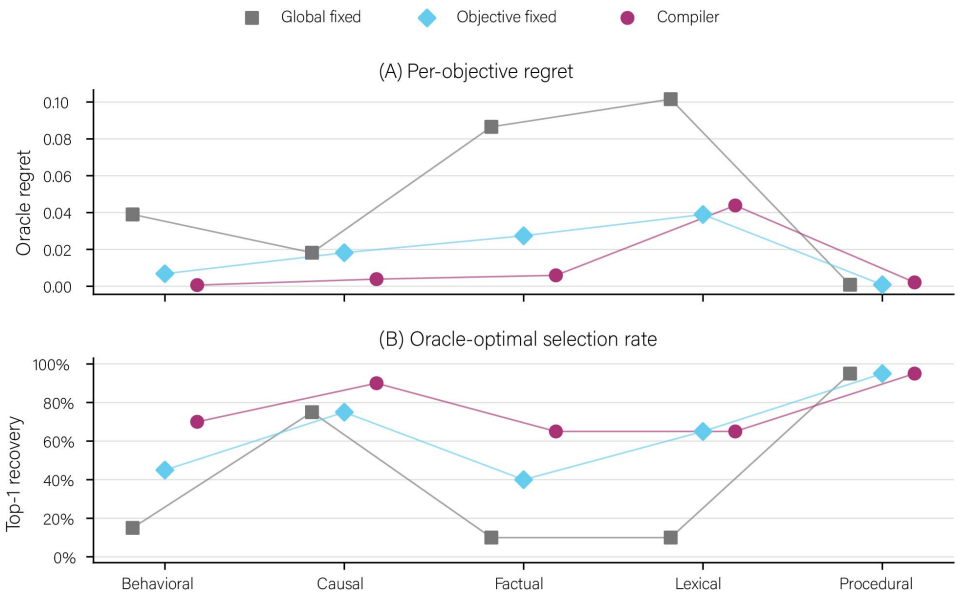


Figure 9: **Program-selection performance by learning objective. (A)** Oracle regret under global-fixed, objective-fixed, and compiler selection. **(B)** Fraction of held-out episodes for which each policy selects an oracle-optimal program. Compiler performance is strongest for behavioral, causal, factual, and procedural episodes, while lexical binding remains the principal residual failure case.

the remaining 71. The mean paired gain is 0.0072 utility; an episode-level bootstrap gives a 95% confidence interval of $[0.0011, 0.0136]$. Thus, the aggregate improvement over the objective-fixed baseline is not produced by symmetric exchanges of similarly valued programs: departures from the objective-level default are more often beneficial than harmful.

D.3 Recompilation under alternative utility specifications

Because the predictor estimates the components of adaptation geometry rather than a single preferred program, changing the utility function requires only rerunning program selection. Table 6 reports the resulting performance and the fraction of compiler decisions that change relative to balanced utility.

Reweighting transfer or boundedness changes the selected program for approximately 14% and 13% of episodes, respectively, whereas increasing the weight on preservation changes only 1% of selections. This difference is consistent with preservation being both comparatively stable across candidate programs and accurately predicted in Experiment 2. More importantly, the compiler remains closer to the oracle than the objective-fixed policy under every utility specification, without retraining the geometry predictor.

Table 6: **Recompilation under alternative utility specifications.** The same predicted adaptation geometry is evaluated under each utility without retraining the predictor. Regret reduction is measured relative to the objective-fixed policy. "Switch" reports the fraction of compiler-selected programs that differ from the balanced-utility selection.

Utility	Compiler U	Oracle U	Compiler Regret	Obj.-fixed Regret	Regret Reduction	Switch
Balanced	.607	.618	.011	.018	38.9%	—
Transfer-heavy	.567	.578	.011	.025	53.4%	14%
Boundedness-heavy	.602	.606	.004	.025	84.2%	13%
Preservation-heavy	.683	.692	.009	.015	39.1%	1%

E Experiment 4: Ablating Pre-Adaptation Information

Table 7 reports the complete prediction and selection metrics for each representation condition. These data are summarized and contextualized alongside the results of Experiments 2 and 3 in Fig. 10. Table 8 summarizes the information available to each predictor. All variants use the same candidate-program descriptors and are selected using validation episodes before evaluation on the held-out test set.

Table 7: **Complete representation-ablation results.** Each restricted predictor is selected using validation performance and evaluated on the same held-out test episodes. Geometry MAE measures prediction error across acquisition, transfer, boundedness, and preservation. Ranking metrics are computed across candidate programs within each episode.

Representation	Val. MAE	Test MAE	Utility Corr.	Spearman	Pairwise	Top-1	Top-2	Oracle Regret
Episode	.0434	.0399	.9633	.8478	.8983	.83	.96	.0088
Full	.0435	.0400	.9620	.8027	.8750	.77	.93	.0113
Module probes	.0526	.0530	.9348	.7655	.8600	.74	.89	.0141
Frozen behavior	.0563	.0538	.9176	.7406	.8400	.71	.91	.0186

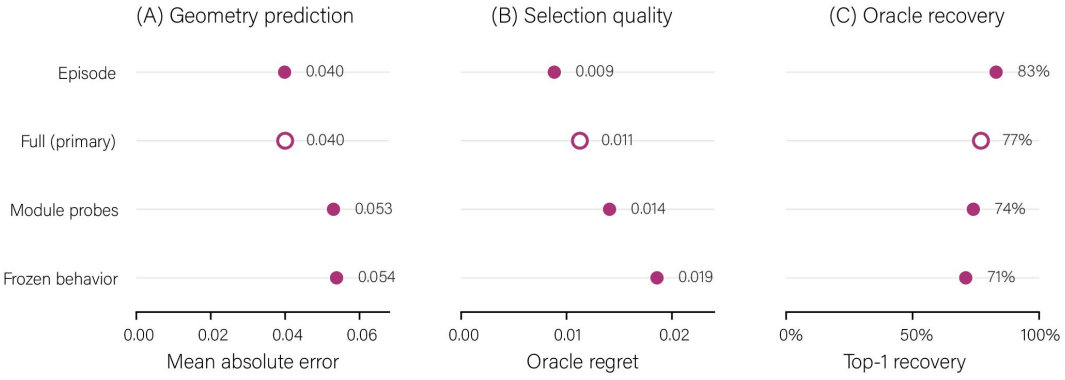


Figure 10: **Representation ablations.** We compare the full episode–model representation with three restricted feature sets. **(A)** Mean absolute error in predicted adaptation geometry. **(B)** Oracle regret of the program selected from predicted geometry. **(C)** Tie-aware top-1 recovery of an oracle-optimal program. The episode representation alone retains performance comparable to the full representation, while module-level probes and frozen-model behavioral statistics are weaker when used independently. The outlined marker denotes the full representation used as the primary predictor in Experiments 2 and 3.

Table 9 decomposes the representation ablations by learning objective. No restricted representation dominates every objective: episode representations perform particularly well for behavioral and factual episodes, while full representations yield the lowest geometry error for lexical and procedural episodes. Prediction error and selection quality also need not coincide; for example, module probes achieve zero oracle regret on procedural episodes despite not minimizing geometry MAE.

Table 8: **Feature sets used in the representation ablation.** Configuration descriptors are provided to all predictors; rows describe the episode–model information available in each condition.

Representation	Hidden-state representation	Frozen-model loss	Module diagnostics	Backward pass required
Episode	✓	–	–	No
Frozen behavior	–	✓	–	No
Module probes	–	–	✓	Yes
Full	✓	✓	✓	Yes

Table 9: **Representation ablations by learning objective.** Geometry MAE measures prediction error across acquisition, transfer, boundedness, and preservation; oracle regret measures the downstream quality of the program selected from each predicted geometry. Bold values indicate the best result within each objective and metric group.

	Geometry MAE ↓				Oracle Regret ↓			
Objective	Episode	Full	Probe	Frozen	Episode	Full	Probe	Frozen
Behavioral	**.0153**	.0155	.0155	.0167	**.0005**	.0006	**.0005**	.0009
Causal	.0346	.0335	.0450	**.0323**	**.0039**	**.0039**	.0092	.0074
Factual	**.0379**	.0406	.0679	.0501	**.0007**	.0059	.0236	.0345
Lexical	.0864	**.0860**	.1100	.1062	**.0370**	.0438	**.0370**	.0423
Procedural	.0252	**.0245**	.0267	.0639	.0021	.0021	**.0000**	.0078

F Experiment 5: Generalizing to Unseen Learning Families

For each leave-one-family-out (LOFO) evaluation, the indicated learning objective is excluded entirely from both meta-training and validation. The predictor is trained on 320 episodes and selected using 80 validation episodes from the remaining four objectives, then evaluated on the 20 test episodes from the held-out objective.

Fig. 11 summarizes the resulting distribution shift. Relative to held-out episodes from represented learning families, excluding an entire family substantially increases geometry-prediction error and degrades program selection. Table 10 decomposes prediction and ranking performance by held-out family. Fig. 12 shows that within-episode ranking fidelity declines across all five held-out families.

Table 11 shows that this degradation is heterogeneous. Zero-shot compilation reduces regret for behavioral and factual episodes, is neutral for causal mapping, and increases regret for lexical and procedural episodes. At the macro level, compiler regret increases from .062 under the global-fixed policy to .085 under LOFO selection.

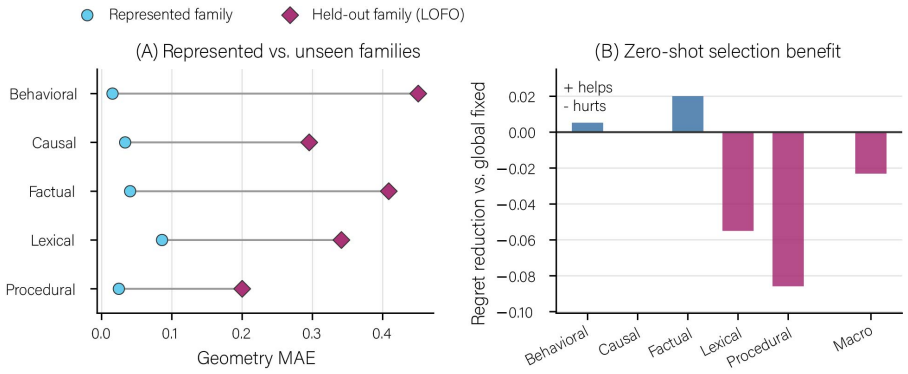


Figure 11: **Generalization beyond represented learning families. (A)** Geometry prediction error for held-out episodes from learning families represented during meta-training compared with leave-one-family-out (LOFO) evaluation, where the test objective is excluded from both training and validation. **(B)** Change in oracle regret of the LOFO compiler relative to the global-fixed policy; positive values indicate improved selection and negative values worse selection.

Table 10: **Leave-one-family-out geometry prediction.** For each fold, the indicated learning family is excluded entirely from both meta-training and validation. Validation MAE is measured on represented learning families, while test metrics are computed on episodes from the held-out family. Ranking metrics compare predicted and observed program utilities within each episode.

Held-out family	Val. MAE	Test MAE	Utility Corr.	Spearman	Pairwise	Top-1	Top-2
Behavioral	.050	.451	.862	.520	.708	.30	.60
Causal	.044	.295	.646	.470	.692	.10	.55
Factual	.043	.409	-.110	-.100	.483	.20	.45
Lexical	.026	.341	-.026	-.550	.258	.05	.15
Procedural	.046	.200	-.037	-.050	.475	.05	.60
Macro	.042	.339	.267	.058	.523	.14	.47

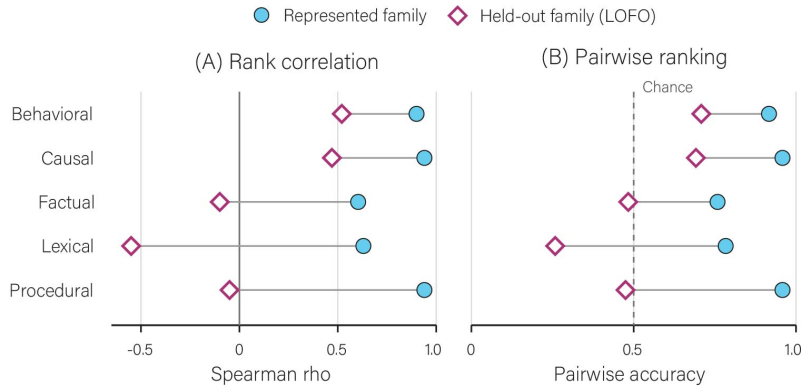


Figure 12: **Program-ranking fidelity under leave-one-family-out generalization.** Within-episode agreement between predicted and observed adaptation-program orderings is compared when the learning family is represented during meta-training versus excluded entirely under LOFO evaluation. (**A**) Spearman rank correlation. (**B**) Pairwise ranking accuracy. Ranking fidelity decreases across all five objectives when the learning family is unseen, with particularly large degradation for factual, lexical, and procedural episodes.

G Experiment 6: Cross-Backbone Replication on Gemma

Experiment 6 asks whether the conditions that support adaptation compilation recur on a second model backbone. This appendix provides the complete Gemma protocol, backbone-specific calibration, adaptation geometry, selection headroom, and compiler-selection results.

G.1 Replication Protocol

We replicate the compiler pipeline on Gemma-2-9B-IT. The learning episodes, train/validation/test partition, behavioral outcomes, utility function, and conceptual program library are unchanged from the Llama experiments. The compiler is retrained from scratch using only Gemma adaptation records; no Llama predictor parameters or adaptation outcomes are transferred.

The corpus contains 400 meta-training, 100 validation, and 100 held-out test episodes, equally divided across the five learning objectives. Each training and validation episode is adapted once per candidate program using seed 11. Each test episode is adapted under three seeds $\{11, 22, 33\}$, and test geometry is defined by averaging outcomes across those three adaptations before selection. This yields 3,200 adaptation runs: 1,600 meta-training, 400 validation, and 1,200 test runs.

Gemma contains 42 transformer layers. We instantiate the same conceptual programs at corresponding relative depths, using layers 0–10 for early adaptation, 15–25 for middle adaptation, and 31–41 for late adaptation. All programs target the attention and feed-forward projection modules.

Table 11: **Program selection under leave-one-family-out generalization.** Utilities are realized under balanced utility. Regret is measured relative to the exhaustive episode oracle. ΔR denotes regret reduction relative to the global-fixed policy, such that positive values indicate beneficial zero-shot compilation.

Held-out family	Global U	Compiler U	Oracle U	Global Regret	Compiler Regret	ΔR
Behavioral	.7144	.7197	.7534	.0390	.0337	+.0052
Causal	.4252	.4252	.5074	.0822	.0822	.0000
Factual	.5044	.5244	.5909	.0865	.0665	+.0201
Lexical	.6882	.6332	.7898	.1016	.1566	-.0550
Procedural	.4493	.3635	.4501	.0008	.0866	-.0858
Macro	.5563	.5332	.6183	.0620	.0851	-.0231

Table 12: Gemma optimization-schedule calibration. Values are averaged across the 25 calibration episodes.

Grad. accum.	Acquisition	Transfer	Boundedness	Preservation	Utility
1	.560	.527	.460	.998	.636
2	.316	.360	.400	.998	.519
4	.168	.207	.220	.998	.398
8	.068	.180	.213	.998	.365

G.2 Backbone-Specific Optimization Calibration

Because the study by Ramnauth & Scassellati (2026) found that Gemma was particularly sensitive to transferring a Llama-calibrated adaptation budget, we calibrate the optimization schedule independently before constructing the Gemma geometry dataset.

Calibration uses five meta-training episodes from each learning objective (25 episodes total), seed 11, and a high-capacity full-stack rank-16 adapter. We hold all other optimization choices fixed and vary gradient accumulation over $\{1, 2, 4, 8\}$. No validation or test episodes are used for this decision.

No candidate satisfied the pre-specified boundedness gate of .80, so the automatic calibration rule was formally inconclusive. We do not relax this criterion post hoc. Instead, after inspecting the calibration tradeoff, we select gradient accumulation 1 because it dominates the alternatives on acquisition, transfer, and boundedness while preserving essentially identical preservation. This schedule is then fixed for all subsequent Gemma adaptations.

Calibration uses the high-capacity full-stack rank-16 condition to determine whether the optimization schedule supports learning when capacity is not the limiting factor, whereas the primary compiler library uses approximately budget-matched localized rank-16 and full-stack rank-4 programs. Consequently, calibration establishes an executable Gemma-specific schedule rather than guaranteeing that every objective is equally learnable under every primary candidate. We do not recalibrate individual programs after observing their test performance.

G.3 Gemma Adaptation Geometry

Table 13 reports the complete test-set geometry after averaging the three adaptation seeds for each episode–program pair. The resulting profiles differ substantially across both learning objectives and adaptation programs. In particular, the mean program profiles show that highest mean utility within a learning family can differ from the program that is oracle-optimal for a substantial fraction of individual episodes. This episode-level heterogeneity is the quantity relevant to compilation.

G.4 Selection Headroom

As in Experiment 1, we construct fixed baselines using meta-training episodes only. The global-fixed policy selects full-stack rank 4. Objective-conditioned defaults select full-stack for behavioral

Table 13: Gemma adaptation geometry on held-out test episodes. Values are means over 20 episodes per objective after averaging the three adaptation seeds. A, T, B, and P denote acquisition, transfer, boundedness, and preservation; U is their balanced mean.

Objective	Program	A	T	B	P	U
Behavioral	Early	.087	.494	.000	.999	.395
	Middle	.343	.556	.000	.999	.474
	Late	.092	.492	.006	.971	.390
	Full	.378	.639	.014	.994	.506
Causal	Early	.015	.036	.397	.996	.361
	Middle	.057	.072	.422	.997	.387
	Late	.000	.072	.356	.962	.347
	Full	.047	.111	.400	.995	.388
Factual	Early	.027	.150	.269	.995	.360
	Middle	.050	.167	.478	.993	.422
	Late	.195	.369	.314	.989	.467
	Full	.142	.317	.364	.992	.454
Lexical	Early	.063	.147	.728	.994	.483
	Middle	.138	.211	.825	.998	.543
	Late	.220	.156	.783	.985	.536
	Full	.122	.175	.803	.995	.524
Procedural	Early	.003	.000	.000	.997	.250
	Middle	.048	.000	.106	.996	.287
	Late	.015	.042	.114	.955	.281
	Full	.023	.000	.219	.994	.309

Table 14: Gemma selection headroom on the 100 held-out test episodes. Global- and objective-fixed programs are selected using meta-training data only. "Opt." is the fraction of episodes on which the fixed policy belongs to the oracle set.

Objective	Obj.-fixed	Global U	Obj. U	Oracle U	Global regret	Obj. regret	Global Opt.	Obj. Opt.
Overall	—	.436	.438	.461	.025	.023	.40	.53
Behavioral	Full	.506	.506	.522	.016	.016	.75	.75
Causal	Middle	.388	.387	.399	.011	.012	.30	.55
Factual	Full	.454	.454	.492	.038	.038	.35	.35
Lexical	Late	.524	.536	.572	.048	.036	.10	.50
Procedural	Full	.309	.309	.322	.013	.013	.50	.50

policy, middle for causal mapping, full-stack for factual association, late for lexical binding, and full-stack for procedural reasoning.

Table 14 reports the resulting test performance. Across all 100 test episodes, the episode-wise oracle achieves mean utility .461, compared with .436 for the global-fixed policy and .438 for objective-fixed selection. The corresponding regrets are .025 and .023. Objective conditioning therefore explains some, but not all, of the variation in preferred adaptation programs.

The causal row illustrates why oracle frequency and mean realized utility need not induce the same ordering: middle adaptation, selected from meta-training data, is oracle-optimal more frequently than full-stack adaptation on causal episodes (.55 versus .30), although full-stack attains slightly higher mean test utility (.388 versus .387). We retain the meta-training-selected middle default rather than selecting between them using test performance.

G.5 Oracle-Program Heterogeneity

Table 15 decomposes the oracle distribution. Using fractional credit for ties, full, middle, late, and early programs account for 40%, 31%, 23%, and 6% of episode-level oracle selections, respectively. Only 2% of test episodes contain an exact top tie. The mean utility difference between the best and second-best program is .039 and the median is .023. Thus, Gemma reproduces the primary

Table 15: Distribution of oracle-optimal Gemma programs on held-out episodes. Program shares use fractional credit for ties.

Objective	Early	Middle	Late	Full
Overall	.06	.31	.23	.40
Behavioral	.00	.25	.00	.75
Causal	.15	.50	.05	.30
Factual	.00	.20	.45	.35
Lexical	.10	.30	.50	.10
Procedural	.05	.30	.15	.50

Table 16: Gemma compiler selection by learning objective. "Selections" reports the number of the 20 held-out episodes assigned to each program. Top-1 and Top-2 are oracle recovery rates.

Objective	Obj.-fixed	Compiler selections	Compiler U	Obj. U	Oracle U	Compiler regret	Top-1	Top-2
Behavioral	Full	Full: 20	.506	.506	.522	.016	.75	1.00
Causal	Middle	Full: 20	.388	.387	.399	.011	.30	.55
Factual	Full	Full: 20	.454	.454	.492	.038	.35	.85
Lexical	Late	Full: 10; Late: 10	.532	.536	.572	.040	.35	.55
Procedural	Full	Full: 8; Middle: 12	.297	.309	.322	.025	.35	.70
Overall	—	—	.435	.438	.461	.026	.42	.73

prerequisite for compilation which is that there is no single adaptation program, nor even a learning-family default, that eliminates episode-level selection headroom.

G.6 Compiler Selection on Gemma

The Gemma geometry predictor is trained only on the 400 Gemma meta-training episodes, with model class and hyperparameters selected on the 100 Gemma validation episodes. The predictor receives the same full pre-adaptation episode–model representation used in the primary Llama experiment and does not receive learning-objective identity.

Table 16 reports downstream selection. The compiler achieves mean utility .435 versus .461 for the episode-wise oracle, corresponding to .026 mean regret. It recovers an oracle-optimal program on 42% of episodes and an oracle top-two program on 73%.

Relative to objective-fixed selection, the compiler improves realized utility on 14 episodes, decreases it on 28, and makes an equivalent choice on the remaining 58. Its mean utility is therefore slightly below the objective-fixed policy (.435 versus .438) and essentially matches the global-fixed policy (.436).

G.7 Where Does Gemma Selection Fail?

The aggregate difference is highly concentrated rather than uniform across learning objectives. The compiler exactly reproduces the full-stack default on all behavioral and factual episodes. On causal mapping it also always selects full-stack; this yields slightly higher mean test utility than the meta-training-selected middle default, although lower oracle recovery. The net shortfall relative to the objective-fixed policy therefore comes from lexical and, especially, procedural episodes.

For lexical binding, the compiler divides its selections evenly between full and late adaptation. These switches improve utility relative to the global full-stack policy but underperform the objective-level late default on average. For procedural reasoning, the compiler selects middle adaptation on 12 of 20 episodes even though the meta-training default is full-stack. This reduces mean utility from .309 under the fixed full-stack policy to .297. Procedural reasoning alone accounts for approximately 79% of the compiler's aggregate utility deficit relative to objective-fixed selection; the smaller lexical deficit is partly offset by a gain on causal episodes.

The procedural errors also reveal a decision-calibration issue. Among the seven procedural episodes for which the compiler selects middle while full-stack is the observed oracle, the predicted advantage of middle over the oracle full-stack program is only .0027 on average (median .0031), yet the mean

Table 17: Diagnostic for procedural episodes on which the compiler selects middle adaptation while full-stack is oracle-optimal. The predicted margin is the predicted utility advantage that triggers the middle-program selection.

Statistic	Value
Number of episodes	7
Mean predicted margin	.0027
Median predicted margin	.0031
Mean realized regret	.0501

Table 18: Objective-conditioned adaptation defaults across backbones. Defaults are estimated independently from each backbone's meta-training episodes.

Objective	Llama	Gemma
Behavioral policy	Late	Full
Causal mapping	Middle	Middle
Factual association	Late	Full
Lexical binding	Early	Late
Procedural reasoning	Middle	Full

realized regret of making that switch is .050. Thus, the argmax decision rule can act on predicted differences that are small relative to the consequence of choosing the wrong program.

This distinction clarifies the Gemma result. Episode-specific adaptation headroom remains present, but exploiting that headroom requires not only accurate expected geometry but sufficiently reliable discrimination between nearby candidate programs. The current compiler treats any predicted utility advantage as actionable, regardless of its magnitude.

G.8 Relationship to the Llama Results

The Gemma replication does not support an architecture-invariant program map. Table 18 compares the objective-conditioned meta-training defaults across backbones. Only causal mapping selects the same region in both models.

This result is consistent with the earlier cross-model finding that adaptation geometry contains both objective-specific and model-specific structure (Ramnauth & Scassellati, 2026). More importantly for the present paper, the *selection problem* itself reproduces: multiple programs remain oracle-optimal across individual Gemma episodes and objective-specific defaults leave measurable residual regret.

What does not reproduce unchanged is the sufficiency of a deterministic argmax over predicted geometry. On Llama, episode-level predictions are accurate enough that direct compilation improves over both fixed policies. On Gemma, the same selection rule sometimes responds to small predicted differences that do not justify departing from a strong default. The cross-backbone result therefore suggests that successful adaptation compilation has two requirements: (1) candidate programs must expose meaningful episode-specific headroom, and (2) the compiler must be sufficiently calibrated to know when that headroom can be exploited reliably.

A natural extension is therefore confidence-aware compilation, in which a compiler retains a robust fixed default unless the predicted advantage of an alternative program exceeds a threshold or uncertainty criterion estimated from validation data. We do not introduce such a rule post hoc here; the Gemma results are reported using the same direct selection procedure specified for the primary experiments.