Automatic Configuration of LLM Post-Training Pipelines

Anonymous ACL submission

Abstract

LLM post-training pipelines that combine supervised fine-tuning and reinforcement learning are difficult to configure under realistic compute budgets: the configuration space is high-dimensional and heterogeneous, stages are strongly coupled, and each end-to-end evaluation is expensive. We propose AutoPipe, a budget-aware two-stage framework for configuration selection in LLM post-training. Offline, AutoPipe learns a dataset-conditioned learning-to-rank surrogate from historical runs, capturing within-dataset preferences and providing transferable guidance toward promising regions of the configuration space. Online, for a new dataset, AutoPipe uses the offline guidance to steer Bayesian optimization and models dataset-specific deviations with a Gaussian-process residual surrogate. To reduce evaluation cost, each trial is early-stopped and scored by a learned predictor that maps early training signals to a low-cost proxy for final post-training performance. Experiments on biomedical reasoning tasks show that AutoPipe consistently outperforms offline-only baselines and achieves comparable performance with the strongest online HPO baselines while using less than 10% of their computational cost.

1 Introduction

Large language models (LLMs) have become a central paradigm in modern AI, delivering strong generalization across language understanding, generation, and reasoning tasks (Lewis et al., 2020; Schick et al., 2023; Singhal et al., 2023). This progress is commonly enabled by a two-stage training recipe: *pre-training* on large-scale corpora to acquire broad linguistic and world knowledge, followed by *post-training* to adapt and align a pre-trained base model to downstream domains, user preferences, and task-specific evaluation criteria. In many practical settings, post-training is implemented as a multi-stage pipeline that combines supervised fine-tuning (SFT), which provides a task-adapted initialization, with reinforcement learning (RL), which further optimizes behavior against downstream objectives (Wang et al., 2024a).

While effective, post-training pipelines are difficult to configure: practitioners must jointly choose a base model together with a heterogeneous collection of stage-specific hyperparameters, and interactions across stages can make the resulting performance landscape highly nontrivial. A central challenge is *cross-stage coupling*: choices made in earlier stages can affect the optimization regime of downstream stages, so optimizing one stage in isolation does not reliably optimize the full pipeline. For instance, Kang et al. (2025) found that improvements in SFT metrics do not necessarily translate into better post-RL performance. As a result, expert-driven tuning often becomes an expensive trial-and-error process that can still yield suboptimal end-to-end performance. Yet automation is also difficult under realistic compute budgets.

Recent approaches reduce this problem to an online black-box hyperparameter optimization (HPO) problem, in which configurations are sequentially proposed and evaluated through expensive end-to-end pipeline executions, so only a small number of trials are feasible in practice. Without a strong prior, such online search also suffers from cold-start inefficiency, wasting early trials in low-quality regions of the search space. This cost can in principle be reduced through multi-fidelity techniques (Li et al., 2017). However, in multi-stage post-training with RL, partial-training signals can be noisy and only weakly, or even non-monotonically, related to the final post-RL outcome, making them unreliable when used in isolation as direct optimization signals. Purely offline solutions are also insufficient: users often optimize on proprietary or domain-specific datasets whose distributions may not appear in historical runs, and offline surrogates learned from past data can suffer from dataset shift.

These challenges motivate a hybrid approach that (i) leverages offline experience to provide a transferable inductive bias over promising configurations, (ii) adapts online to the target dataset using a small number of evaluations, and (iii) uses proxy signals from truncated runs to reduce the cost of online adaptation.

We propose *AutoPipe*, a two-phase framework for budgeted configuration selection in LLM post-training pipelines. In the offline phase, AutoPipe learns a dataset-conditioned learning-to-rank surrogate from historical post-training runs, capturing within-dataset preferences among configurations and providing transferable guidance toward promising regions of the search space. In the online phase, for a new target dataset, AutoPipe uses these offline ranking scores as meta-learned guidance and performs Bayesian optimization over residual corrections, concentrating limited online capacity on correcting dataset-specific miscalibration. To reduce evaluation cost, each online candidate is assessed via a truncated (early-stopped) run; a learned early-stop predictor maps early training signals to a calibrated proxy of final performance, which is then used to support robust online updates under noisy feedback. Across biomedical question answering tasks, AutoPipe achieves performance comparable to the strongest baseline while using less than 10% of its computational cost.

Contributions. (1) We propose *AutoPipe*, a budget-aware framework for configuration selection in LLM post-training pipelines. (2) We develop a dataset-conditioned ranking surrogate that transfers configuration preferences across datasets and offers strong offline guidance toward promising candidates. (3) We introduce an online adaptation mechanism based on Gaussian-process residual modeling and an early-stop proxy predictor to correct dataset-specific miscalibration under tight evaluation budgets. (4) Extensive experiments on biomedical reasoning datasets show that AutoPipe consistently outperforms strong offline-only baselines and remains competitive with the strongest online baselines at substantially lower training cost.

2 Related Work

Hyperparameter Optimization. Bayesian optimization (BO) is a standard approach for sample-efficient hyperparameter optimization (Snoek et al., 2012). Systems such as SMAC (Hutter et al., 2011) and TPE (Bergstra et al., 2011) extend BO to mixed discrete-continuous spaces, while multi-fidelity methods like Hyperband (Li et al., 2017) and BOHB (Falkner et al., 2018) adaptively allocate compute. These methods often assume low-fidelity signals predict final performance; in multi-stage LLM post-training, early SFT may correlate weakly with downstream RL outcomes. Classic BO can also require many evaluations, which is impractical when full pipelines are expensive.

Meta- and Transfer-Learning. To reduce optimization costs, we may leverage traces from related tasks via similarity-weighted surrogate ensembles (Feurer et al., 2015), two-stage offline-to-online procedures (Wistuba et al., 2016), or learned priors and search-space biases from meta-datasets (Perrone et al., 2018). Recent offline recommendation solutions, including Quick-Tune (Arango et al., 2024) and PFNs4BO (Müller et al., 2023), often treat learned priors or predictors as static initializations and can fail under distribution shift. This motivates hybrid strategies that reuse offline signal while adapting to new observations.

Automatic LLM Post-Training Pipelines. The modern paradigm combining SFT with RL was popularized by InstructGPT (Ouyang et al., 2022). Configuration choices across stages substantially affect final quality, yet end-to-end optimization remains limited. Recent work optimizes components of this process, including black-box search for instruction-tuning hyperparameters (Tribes et al., 2024), multi-objective optimization of LLM and RAG pipelines at inference time (Barker et al., 2025), and agent-based frameworks for pipeline construction such as LaMDAgent (Yano et al., 2025). In contrast, our work targets end-to-end SFT+RL pipelines, emphasizing budget-limited optimization by leveraging historical runs while allowing online adaptation to new datasets.

3 Methodology

Modern LLM applications are increasingly implemented as *multi-stage* systems, commonly involving components such as SFT, retrieval augmentation, RL, and routing mechanisms. In this work, we focus on a widely deployed core pipeline consisting of SFT followed by RL.

Given a target dataset, post-training performance depends on a *large, heterogeneous* configuration space spanning both discrete and continuous choices (e.g., base model, batch sizes and learning

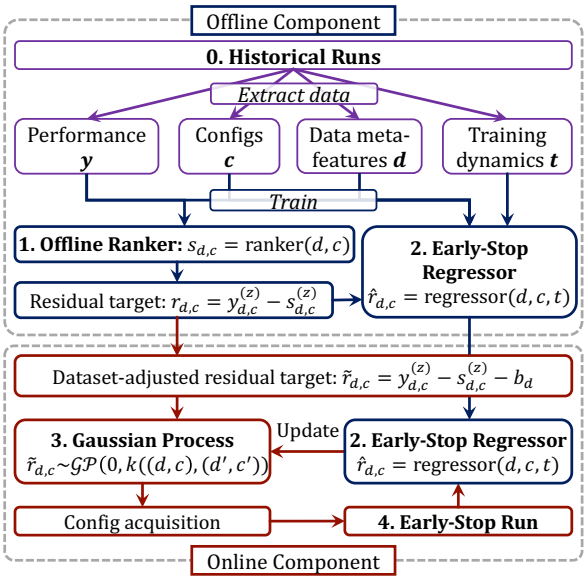

Figure 1: Overview of the proposed two-phase configuration selection framework.

rate schedule). End-to-end evaluation of a single configuration requires running the full pipeline, including expensive RL rollouts, thus prohibiting exhaustive or even moderately broad search. This motivates methods that (i) leverage historical evidence across datasets and (ii) allocate limited online compute toward only the most promising candidates.

Problem Formulation. Let d denote a dataset and $c \in \mathcal{C}$ denote a post-training configuration covering SFT and RL hyperparameters. Running the pipeline with (d, c) yields a performance score $y(d, c)$ (e.g., accuracy on a target evaluation suite). Given a new dataset $d^\star$, the goal is to identify the best configuration

$$c^\star \in \arg\max_{c \in \mathcal{C}} y(d^\star, c), \tag{1}$$

subject to a strict online evaluation budget measured in number of evaluations.

Two considerations guide our design. First, our objective is selection: for each dataset, we aim to identify the best configuration (or a small set of top candidates), rather than precisely predict absolute performance. Second, datasets differ in ways that break global consistency, that is, the relative ordering of configurations is not stable across datasets. Hence, we seek to learn *dataset-conditioned* preferences over configurations.

Two-Phase Optimization. We describe our solution below and in Figure 1:

1. *Offline phase (amortized learning):* Using historical runs across datasets, we learn a dataset-conditioned ranking surrogate that produces configuration ranking scores $s_{d,c}$.
2. *Online phase (budgeted adaptation):* For a new dataset $d^\star$, the surrogate scores $s_{d^\star,c}$ are used as meta-learned guidance within BO. To reduce evaluation cost, configurations are executed via early-stopped runs whose outcomes are mapped to proxy scores by an early-stop predictor.

This design separates knowledge acquisition (offline) from dataset-specific adaptation (online), allowing for historical evidence to be exploited while remaining flexible enough to correct for dataset-specific deviations under tight compute constraints. Below we elaborate on these two phases.

Offline Surrogate Learning. We assume an offline corpus of completed runs, each consisting of a dataset d, configuration c, and performance score $y(d, c)$ (e.g., average benchmark accuracy). Runs are organized into *dataset groups*, where each group contains evaluations of multiple configurations on the same dataset.

Ranking supervision. Rather than predicting absolute performance, we formulate the surrogate learning problem as *learning to rank*. For each dataset group, observed outcomes induce a partial ordering over configurations. Training supervision thus consists of pairwise preference constraints of the form $c_i \succ c_j$ whenever $y(d, c_i) > y(d, c_j)$. We restrict comparisons to *within-dataset* comparisons to reduce sensitivity to cross-dataset shifts in performance scale, focusing the surrogate on identifying the most promising configurations for a given dataset.

Surrogate model. We instantiate the surrogate using a gradient-boosted tree ranker (XGBRanker) trained with the pairwise ranking objective (`rank:pairwise`). Each example uses features

$$\phi(d, c) = \left[\phi_{\text{cfg}}(c)\ ;\ \phi_{\text{meta}}(d)\right],$$

where ϕ_{cfg} encodes SFT/RL hyperparameters and ϕ_{meta} encodes dataset features. During inference, the surrogate scores candidate configurations for a target dataset $d^\star$, producing ranking scores $s_{d^\star,c}$ that estimate the relative quality of configurations. We adopt gradient-boosted trees for their robustness in low-data regimes and their ability to capture non-linear interactions between configuration choices and dataset properties. Detailed feature definitions, preprocessing steps, and surrogate hyperparameters are discussed in Appendix B.1.

Online BO with Meta-Learned Guidance. The offline ranker captures transferable structure in configuration quality, but directly selecting configurations by $s_{d^\star,c}$ can be suboptimal due to shifts in performance landscape for the new dataset $d^\star$. We thus perform BO using the offline score as meta-learned guidance, adapting to the target dataset through a small number of online evaluations.

Offline-Guided Decomposition. Let $s^{(z)}_{d^\star,c}$ denote the standardized offline score assigned by the ranker to configuration c for dataset $d^\star$. In the online phase, we do not model absolute performance from scratch; instead, we maintain a GP over dataset-specific correction terms relative to this offline guidance. Conceptually, we define the correction target as

$$\tilde{r}_{d^\star,c} = y^{(z)}_{d^\star,c} - s^{(z)}_{d^\star,c} - b_{d^\star}, \tag{2}$$

where $b_{d^\star}$ is a dataset-specific offset that captures systematic mismatch between the offline score and the target dataset. The GP models configuration-dependent variation in this correction term:

$$f_{d^\star}(c) \approx \tilde{r}_{d^\star,c}, \quad f_{d^\star}(c) \sim \mathcal{GP}(0, k(c, c')). \tag{3}$$

In practice, the online procedure never requires direct access to true full-fidelity values $y^{(z)}_{d^\star,c}$ during search. Instead, it updates the GP using low-cost pseudo-observations derived from the early-stop predictor, while the offline score $s^{(z)}_{d^\star,c}$ serves as a persistent guidance signal throughout optimization, with its influence modulated over iterations. For the GP, we use a Matérn-$5/2$ kernel with automatic relevance determination (ARD) over configuration dimensions, providing robustness against non-uniform sensitivity in the configuration space (Snoek et al., 2012).

Posterior prediction and tempered guidance. At BO iteration t, let $\mu_{f,t}(c)$ and $\sigma_{f,t}(c)$ denote the GP posterior mean and standard deviation over the correction term. We form the acquisition score using a tempered combination of offline guidance and online correction:

$$\hat{y}^{(z)}_{d^\star,c,t} = w_t\, s^{(z)}_{d^\star,c} + b_{d^\star,t} + \mu_{f,t}(c), \tag{4}$$

where $b_{d^\star,t}$ is the current dataset-specific offset estimate and $w_t \in [0, 1]$ controls how strongly the search remains anchored to the offline ranker. We linearly anneal w_t over iterations so that early search is guided more strongly by transferable cross-dataset structure, while later iterations place greater weight on dataset-specific evidence.

Acquisition and posterior updates. We select the next configuration by maximizing an Upper Confidence Bound objective,

$$\alpha_t(c) = \hat{y}^{(z)}_{d^\star,c,t} + \beta_t\, \sigma_{f,t}(c), \tag{5}$$

where β_t controls the exploration–exploitation trade-off. Since $s^{(z)}_{d^\star,c}$ is fixed across iterations, exploration is driven by uncertainty in the residual correction, while the overall search trajectory remains anchored to the meta-learned guidance. Each selected configuration is then evaluated via the early-stop predictor; the resulting pre-bias residual pseudo-observation is incorporated into the online update by first refreshing the dataset-specific offset and then refitting the GP on centered targets. Additional implementation details are provided in Appendix B.3 and Section B.3.2.

Early-Stop Predictor. Full end-to-end SFT+RL evaluation is too expensive to serve as the online feedback signal. We instead employ an *early-stop predictor* that uses partial training signals from an inexpensive truncated run to approximate the full evaluation outcome. The key idea is to exploit the fact that downstream RL performance depends strongly on the quality and stability of the SFT initialization, making early SFT dynamics a useful source of low-cost evidence for online updates.

Predicting standardized residual corrections. For each executed run, we extract early training features $x_{\text{es}}(d, c)$ such as summary statistics computed from logged training dynamics. Rather than regressing directly on absolute end-to-end performance, we train the predictor to estimate the standardized residual correction

$$r_{d,c} = y^{(z)}_{d,c} - s^{(z)}_{d,c}, \tag{6}$$

where $y^{(z)}_{d,c}$ is the standardized full evaluation score and $s^{(z)}_{d,c}$ is the standardized offline ranking score. We instantiate $g(\cdot)$ as an ensemble of XGBoost regressors (XGBRegressor), trained to predict

$$\hat{r}_{d,c} = g(x_{\text{es}}(d, c)) \approx r_{d,c}, \tag{7}$$

so that the final proxy score is

$$\hat{y}^{(z)}_{d,c} = s^{(z)}_{d,c} + \hat{r}_{d,c}. \tag{8}$$

This makes the predictor a conservative correction mechanism: when early-stop signals are uninformative, predictions naturally remain anchored to the offline score.

Use in online optimization. For a new dataset $d^\star$, each candidate is run only to the early-stop point, from which $x_{\text{es}}(d^\star, c)$ is computed. The predictor then produces a residual correction $\hat{r}_{d^\star,c}$, which is used as a pseudo-observation to update the GP in residual space. The reconstructed score

$$\hat{y}^{(z)}_{d^\star,c} = s^{(z)}_{d^\star,c} + \hat{r}_{d^\star,c} \tag{9}$$

serves as a low-cost proxy for comparing candidate configurations and deciding which ones merit full end-to-end evaluation. Feature definitions, training and other implementation details are deferred to Appendix B.2.

Remark. Our key design principle is to separate *cross-dataset regularities* from *dataset-specific correction*. The offline stage amortizes knowledge about configuration quality across datasets and supplies meta-learned guidance, while the online stage refines it through residual updates from inexpensive early-stopped runs. This enables adaptive configuration selection under tight evaluation budgets.

4 Experiments

4.1 Experimental Setup

We evaluate on biomedical chain-of-thought question answering (QA), a setting with datasets varying in style and reasoning depth, thereby providing a meaningful testbed for configuration transfer.

Table 1: Post-training datasets used in this work.

Dataset	Reference
MQA	Ours
MQAR	Ours
PQA	Ours
MQA(R)	Ours
MQA(R)PQA	Ours
HuaTuo	(Chen et al., 2024)
MedReason	(Wu et al., 2025)
MedS3	(Jiang et al., 2025)
ReasonMed	(Sun et al., 2025)
m23k	(Huang et al., 2026)

Datasets and Benchmarks. We use a diverse collection of medical reasoning datasets for post-training and evaluate selected configurations on a fixed suite of downstream biomedical benchmarks. Half of the datasets are newly constructed in this work, while the remainder are open-source datasets from recent medical reasoning papers. Tables 1 and 2 summarize datasets and benchmarks; detailed descriptions are provided in Appendix A.1.

Table 2: Downstream benchmarks in our evaluation.

Benchmark	Reference
BioASQ (yes/no)	(Xiong et al., 2024)
GPQA (Biology)	(Rein et al., 2024)
MedBullets (4 options)	(Chen et al., 2025)
MedBullets (5 options)	(Chen et al., 2025)
MedMCQA	(Pal et al., 2022)
MedQA	(Jin et al., 2021)
MedXpertQA	(Zuo et al., 2025)
MMLU (Medicine)	(Hendrycks et al., 2021)
MMLU-Pro (Bio & Health)	(Wang et al., 2024b)
PubMedQA	(Jin et al., 2019)
PubMedQA (questions only)	(Jin et al., 2019)
PubMedQA (with distractor abstracts)	(Jin et al., 2019), this work

Table 3: Configuration space searched by all methods.

Phase	Hyperparameter	Possible values
	Base model	Qwen2.5-1.5B-Instruct Qwen2.5-3B-Instruct Qwen2.5-7B-Instruct Llama3.2-1B-Instruct Llama3.2-3B-Instruct Llama3.1-8B-Instruct
SFT	Number of epochs	1, 2, 3
	Effective batch size	32, 64, 128
	Learning rate	$1e^{-5}, 2e^{-5}, 5e^{-5}$
RL	Effective batch size	64, 128, 256
	Learning rate	$1e^{-6}, 2e^{-6}, 5e^{-6}, 1e^{-5}$
	Beta	0, 0.05, 0.1
	Number of rollouts	8, 16
	Temperature	0.7, 0.9, 1.1

We evaluate transfer to unseen datasets using a leave-one-dataset-out (LODO) protocol. In each split, one dataset is held out as the target $d^\star$, and the remaining datasets form the offline corpus used to train both the ranking surrogate and early-stop regressor. The online procedure is then run on $d^\star$ under a fixed number of online evaluations, reflecting a practical deployment setting in which extensive prior experimentation is infeasible. Additional details are provided in Appendix C.1.

Configuration space. All methods search a shared configuration space spanning base model choice and SFT/RL hyperparameters. Table 3 summarizes the space; Appendix A.2 details how we sample from the space to construct the offline corpus.

Baselines. We compare against two types of baselines: (1) *static / zero-shot* methods, which do not

perform any online evaluations on the target dataset, and (2) *sequential* methods, which adapt through online evaluations under the budget constraint. Under *static / zero-shot baselines*, we consider: (1) *Random*: one random configuration; (2) *LLM Recommendation (Description-only)*: prompts an LLM to propose a configuration, given only task and dataset description; (3) *LLM Recommendation (History-conditioned)*: additionally provides a set of similar historical datasets with evaluated configurations and outcomes; (4) *Global*: selects the configuration that achieved the best performance across the offline corpus (excluding the held-out dataset); (5) *Offline-only*: selects $\arg\max_c(s_{d^\star,c})$ without online adaptation. For *sequential baselines*, we compare: (1) *Random Search*: randomly samples configurations; (2) *BOHB* (Falkner et al., 2018): combines multi-fidelity resource allocation with kernel density estimators to prioritize promising configurations; (3) *SMAC* (Hutter et al., 2011): uses random-forest-based BO for algorithm configuration; (4) *TPE* (Bergstra et al., 2011): fits separate density models to high- and low-performing regions of the search space and selects configurations that maximize their density ratio; (5) *AutoPipe*, our full solution. Baseline implementation details are provided in Appendix C.2.

Metrics. We report metrics that capture end-to-end performance and selection quality under a limited number of online evaluations.

Pipeline accuracy. We report the (1) *final benchmark score* achieved by the configuration $\hat{c}$ selected by each method after full post-training, (2) the corresponding *regret*, $y(d^\star, c^\star_{\text{opt}}) - y(d^\star, \hat{c})$, where $c^\star_{\text{opt}}$ is the best-performing configuration in the reference pool for $d^\star$, i.e., the set of all configurations evaluated on $d^\star$ anywhere in this study, rather than the entire search space, and (3) the *evaluation cost*, measured in H200 GPU hours, which includes both the cost incurred during optimization and the cost of evaluating the selected configuration. Final benchmark score is the primary metric, while regret quantifies the remaining optimality gap.

Ranking and selection diagnostics. For component-level evaluation, we additionally report diagnostics for the models trained on the offline corpus, namely the offline surrogate and the early-stop predictor. Specifically, we report: (1) *Recall@k*, whether $c^\star_{\text{opt}}$ appears in the top-k ranked configurations; (2) *nDCG@k*, which emphasizes correct ordering near the top of the ranking; (3) *Pairwise accuracy*, the fraction of configuration pairs ordered correctly within each dataset; and (4) *Spearman correlation*, the rank correlation between predicted and ground-truth within-dataset orderings.

Table 4: LODO configuration selection performance under a fixed number of online evaluations using either full-fidelity signals or early-stop (ES) proxy signals. Performance is measured in average benchmark accuracy and cost in average H200 hours required.

Method	Performance	Regret	Cost
Static & Zero-Shot Baselines			
Random (Single run)	0.391	0.237	3.62
LLM Rec. (Description-only)	0.307	0.321	7.60
LLM Rec. (History-conditioned)	0.442	0.186	5.59
Global	0.529	0.099	4.05
Offline-only (Ranker)	0.558	0.070	6.16
Sequential Baselines			
Random Search			
ES	0.446	0.182	4.95
Full	0.565	0.063	58.5
BOHB	0.590	0.038	95.5
SMAC			
ES	0.504	0.123	5.99
Full	0.580	0.048	64.0
TPE			
ES	0.545	0.083	6.27
Full	0.570	0.058	66.8
Ours (Two-phase, ES)	**0.590**	**0.038**	**8.71**

4.2 Results and Analysis

We evaluate the framework to assess overall end-to-end effectiveness and to clarify the contribution of its components. Our analysis has three parts: (i) configuration selection under low-cost proxy-feedback and higher-cost full-fidelity optimization regimes, measured by full-fidelity performance of the selected configuration; (ii) ablations of key design choices; and (iii) sensitivity to online budget and early-stop fidelity.

Efficacy across budget regimes. We study configuration selection when optimization uses early-stopped, noisy evaluations, but success is measured by full end-to-end performance. We compare proxy-guided methods in a resource-constrained regime against sequential optimizers that operate on full-fidelity evaluations. In the constrained regime, methods use early-stopped runs scored by the early-stop predictor; after the budget is exhausted, the selected configuration is verified with a full-fidelity run.

Table 4 reports full-fidelity performance, regret and optimization cost across offline and online baselines. The results shed light on the role of learned offline structure and online adaptation in configuration selection. Firstly, the offline ranker is already a strong baseline, establishing a per-

formance floor that outperforms other static baselines. While random selection and LLM-based recommendations struggle with the interaction-heavy post-training landscape, the ranker captures transferable patterns across datasets, producing a markedly improved initial configuration ordering. The performance of the experience-informed yet dataset-agnostic *Global* baseline further highlights the importance of dataset conditioning. Although it achieves competitive average performance, it exhibits higher variability across datasets, performing comparably or even better than dataset-conditioned selection in certain cases while incurring substantial failures in others (dataset-level comparisons are provided in Appendix D.1). This suggests that experience alone is insufficient; without dataset conditioning, strong average performance can mask brittle behavior under distributional shift.

Secondly, incorporating the online phase further elevates performance, bringing the full two-phase solution into a regime competitive with expensive sequential baselines like BOHB, SMAC, and TPE. These methods, which make use of multiple end-to-end evaluation cycles, represent a high-resource point of comparison. Notably, their early-stopped variants consistently outperform static methods but remain inferior to their full-fidelity counterparts, indicating that early-stop signals are noisy yet directionally informative. This gap underscores the value of offline structure: proxy signals alone are insufficient, but when used to refine a strong dataset-conditioned prior, they can recover much of the benefit of sequential search.

Overall, the end-to-end results show that competitive configuration selection is achievable even when online optimization is limited to early-stopped signals, provided that these signals are combined with structure learned across datasets. The two-phase framework narrows the efficiency-quality gap by amortizing global interaction structure offline and reserving online computation for dataset-specific correction.

Offline Component Analysis. We next examine whether the offline components provide useful signals before online adaptation. Table 5 reports ranking metrics between true end-to-end performance and predictions for the offline ranker and early-stop regressor. The offline ranker performs strongly across nDCG@k, recall@k and pairwise ordering metrics, indicating that it transfers useful ordering structure across datasets despite heterogeneous performance scales. The early-stop regressor is competitive with, and on several metrics slightly stronger than, the offline ranker, while serving a different role: it predicts dataset-specific residual corrections from partial training dynamics, thus acting as a cautious correction mechanism rather than a standalone selector. Taken together, the two components are complementary: the ranker provides broad cross-dataset guidance, and the regressor supplies targeted low-cost corrections.

Table 5: LODO ranking performance of offline surrogate and early-stop (ES) regressor.

Metric	Offline Ranker	ES Regressor
Regret	0.110	0.079
Recall@1	0.100	0.100
Recall@3	0.300	0.300
Recall@5	0.400	0.300
Recall@10	0.700	0.500
nDCG@5	0.853	0.862
nDCG@10	0.869	0.861
Pairwise Accuracy	0.750	0.764
Spearman ρ	0.684	0.706

4.3 Ablation Studies

We ablate three key components of the framework to verify that observed gains arise from their interaction rather than from any single design choice: (1) the residual formulation of the GP, (2) the BO procedure itself, and (3) the offline ranker.

No Residual Formulation. This ablation removes the residual formulation, where the GP models corrections to the ranking score ($y_{d,c}^{(z)} - s_{d,c}^{(z)} - b_d$), and instead models $y_{d,c}^{(z)}$ directly. The offline ranker is still used to guide candidate selection. This tests whether learning residuals from a strong prior improves sample efficiency compared to learning the objective from scratch.

No BO. This ablation removes the BO loop and evaluates the top k configurations ranked by the offline ranker. This tests whether adaptive search adds value beyond trusting the offline ranker alone.

No Offline. This ablation removes all involvement of the offline ranker from the online phase. The GP models $y_{d,c}^{(z)}$ directly, candidate selection is no longer guided by the ranker, and the early-stop regressor is retrained to predict $y_{d,c}^{(z)}$ rather than residuals. This tests the overall value of incorporating offline knowledge into the search process.

Table 6 clarifies the role of each component in the framework. Removing any single one degrades

Table 6: Ablation results for main components.

Variant	Final Score	Regret	Cost
Full method	0.590	0.038	8.71
No Residual Formulation	0.516	0.112	7.88
No BO	0.527	0.101	9.14
No Offline	0.497	0.131	5.60

performance, with the largest drop occurring when offline structure is removed entirely in the *No Offline* variant. This variant exhibits performance comparable to the early-stopped versions of the sequential baselines, suggesting that without offline learned guidance, the method largely reverts to classical BO under noisy and weakly informative signals. The *No Residual Formulation* ablation isolates the effect of residual modeling. Despite retaining adaptive BO and ranker-guided candidate selection, directly modeling absolute performance instead of residual corrections leads to a substantial performance drop, suggesting that correcting the offline score is more sample-efficient than learning the full objective from noisy proxy observations. Interestingly, *No BO* outperforms *No Residual Formulation* despite the removal of adaptive search. This indicates that a structured prior with conservative correction can be more effective than adaptive optimization applied to a poorly specified objective. Overall, these ablations show that performance gains arise from the interaction between offline structural knowledge and residual-based online updates. BO is an effective refinement mechanism only when this structure is present.

4.4 Sensitivity Analysis

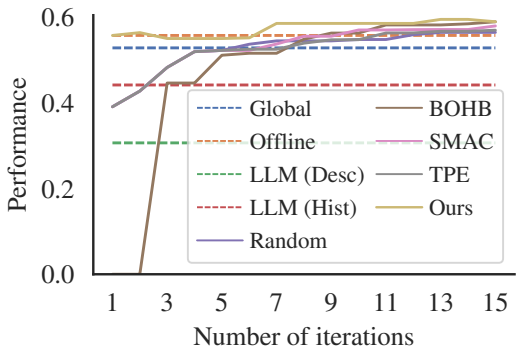

Figure 2: Sensitivity of performance (average benchmark accuracy) to online evaluation budget.

Sensitivity to budget. Figure 2 plots performance against the number of online evaluations. For our method, online updates are driven by early-stopped SFT signals passed through the early-stop regressor, whereas the full-fidelity sequential baselines use full end-to-end evaluations as their optimization signal. Despite operating under this noisier low-cost feedback regime, our approach performs strongly even at low evaluation counts, due to rapid early gains from starting search in a favorable region identified by the offline ranker. With larger budgets, improvements become incremental, suggesting diminishing returns once the initial guidance has been locally refined. The mild non-monotonicity at very low budgets reflects noise in the proxy-feedback loop, but this effect is small and is quickly corrected as more evidence accumulates.

Table 7: Sensitivity to early-stop truncation length.

Variant	Final Score	Regret	Cost
25%	0.565	0.063	7.02
50%	0.590	0.038	8.71
75%	0.583	0.045	9.58

Sensitivity to early-stop fidelity. Table 7 reports performance when early-stop runs are truncated at 25%, 50%, and 75% of the full SFT trajectory. Very short truncation (25%) degrades performance, suggesting that coarse early-stop signals are insufficiently reliable. Increasing truncation to 50% yields substantial improvement, while extending to 75% provides no consistent gains and can slightly hurt performance. Overall, signal utility appears to saturate beyond moderate fidelity, and performance remains stable across multiple truncation lengths. This robustness suggests that partial training dynamics are a practical proxy for online adaptation without requiring precise truncation tuning.

5 Conclusion

AutoPipe's core takeaway is that ranking transfer with local correction can effectively tune coupled SFT–RL pipelines when full end-to-end trials are too expensive; an offline, dataset-conditioned ordering narrows the search to promising regions, and a small amount of online evidence suffices to correct dataset-specific miscalibration. In biomedical QA, this strategy consistently outperforms strong offline-only baselines and remains competitive with the strongest online HPO baselines while using substantially less compute.

Limitations

A remaining weakness is reliance on early-stopped proxy signals and historical priors: when truncated SFT dynamics are noisy or weakly related to post-RL outcomes, or under dataset shift, performance can be mildly non-monotonic at very low budgets before correcting as more evidence accrues.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA. Association for Computing Machinery.

Sebastian Pineda Arango, Fabio Ferreira, Arlind Kadra, Frank Hutter, and Josif Grabocka. 2024. Quick-tune: Quickly learning which pretrained model to finetune and how. In *The Twelfth International Conference on Learning Representations*.

Matthew Barker, Andrew Bell, Evan Thomas, James Carr, Thomas Andrews, and Umang Bhatt. 2025. Faster, cheaper, better: Multi-objective hyperparameter optimization for llm and rag systems. *Preprint*, arXiv:2502.18635.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA. Curran Associates Inc.

Hanjie Chen, Zhouxiang Fang, Yash Singla, and Mark Dredze. 2025. Benchmarking large language models on answering and explaining challenging medical questions. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3563–3599, Albuquerque, New Mexico. Association for Computational Linguistics.

Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. 2024. Huatuogpt-o1, towards medical complex reasoning with llms. *Preprint*, arXiv:2412.18925.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. BOHB: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2755–2763, Cambridge, MA, USA. MIT Press.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Xiaoke Huang, Juncheng Wu, Hui Liu, Xianfeng Tang, and Yuyin Zhou. 2026. m1: Unleash the potential of test-time scaling for medical reasoning with large language models. In *Proceedings of the Fifth Machine Learning for Health Symposium*, volume 297 of *Proceedings of Machine Learning Research*, pages 369–383. PMLR.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer Berlin Heidelberg.

Shuyang Jiang, Yusheng Liao, Zhe Chen, Ya Zhang, Yanfeng Wang, and Yu Wang. 2025. Meds^3: Towards medical slow thinking with self-evolved soft dual-sided process supervision. *Preprint*, arXiv:2501.12051.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14).

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Qiao Jin, Won Kim, Qingyu Chen, Donald C Comeau, Lana Yeganova, W John Wilbur, and Zhiyong Lu. 2023. Medcpt: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval. *Bioinformatics*, 39(11).

Feiyang Kang, Michael Kuchnik, Karthik Padthe, Marin Vlastelica, Ruoxi Jia, Carole-Jean Wu, and Newsha Ardalani. 2025. Quagmires in sft-rl post-training: When high sft scores mislead and what to use instead. *Preprint*, arXiv:2510.01624.

Anastasia Krithara, Anastasios Nentidis, Konstantinos Bougiatiotis, and Georgios Paliouras. 2023. Bioasq-qa: A manually curated corpus for biomedical question answering. *Scientific Data*, 10(1):170.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: a novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18(1):6765–6816.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. Smac3: a versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.*, 23(1).

Samuel Müller, Matthias Feurer, Noah Hollmann, and Frank Hutter. 2023. Pfns4bo: in-context learning for bayesian optimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cedric Archambeau. 2018. Scalable hyperparameter transfer learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

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

Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: language models can teach themselves to use tools. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Abubakr Babiker, Nathanael Schärli, Aakanksha Chowdhery, Philip Mansfield, Dina Demner-Fushman, and 13 others. 2023. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA. Curran Associates Inc.

Yu Sun, Xingyu Qian, Weiwen Xu, Hao Zhang, Chenghao Xiao, Long Li, Deli Zhao, Wenbing Huang, Tingyang Xu, Qifeng Bai, and Yu Rong. 2025. ReasonMed: A 370K multi-agent generated dataset for advancing medical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 26446–26467, Suzhou, China. Association for Computational Linguistics.

Christophe Tribes, Sacha Benarroch-Lelong, Peng Lu, and Ivan Kobyzev. 2024. Hyperparameter optimization for large language model instruction-tuning. *Preprint*, arXiv:2312.00949.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, and 3 others. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. 2024a. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024b. Mmlu-pro: a more robust and challenging multi-task language understanding benchmark. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Martin Wistuba, Nicolas Schilling, and Lars Schmidt-Thieme. 2016. Two-stage transfer surrogate model

for automatic hyperparameter optimization. In *Machine Learning and Knowledge Discovery in Databases*, pages 199–214. Springer International Publishing.

Juncheng Wu, Wenlong Deng, Xingxuan Li, Sheng Liu, Taomian Mi, Yifan Peng, Ziyang Xu, Yi Liu, Hyunjin Cho, Chang-In Choi, Yihan Cao, Hui Ren, Xiang Li, Xiaoxiao Li, and Yuyin Zhou. 2025. Medreason: Eliciting factual medical reasoning steps in llms via knowledge graphs. *Preprint*, arXiv:2504.00993.

Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. 2024. Benchmarking retrieval-augmented generation for medicine. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6233–6251, Bangkok, Thailand. Association for Computational Linguistics.

Taro Yano, Yoichi Ishibashi, and Masafumi Oyamada. 2025. Lamdagent: An autonomous framework for post-training pipeline optimization via llm agents. *Preprint*, arXiv:2505.21963.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. 2025. Medxpertqa: benchmarking expert-level medical reasoning and understanding. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org.

A Problem Setup

A.1 Dataset and Benchmark Details

A.1.1 Dataset details

We study our methods on ten instruction-style datasets for medical reasoning post-training. The datasets differ along several axes, including prompt sources, teacher model, use of retrieval augmentation, and response format. Taken together, they form a heterogeneous testbed for investigating configuration selection in multi-stage LLM post-training pipelines for medical reasoning.

Datasets introduced in this paper

For the datasets newly introduced in this work, both reasoning traces and final responses are distilled from DeepSeek-R1, and examples are retained only if the final answer is correct. The datasets differ primarily in how prompts are constructed, as detailed below.

MQA. MQA is built from a subset of medical and clinical questions sampled from the training splits of MedQA (Jin et al., 2021) and MedMCQA (Pal et al., 2022).

MQAR. MQAR extends MQA with retrieval augmentation and is intended to assess whether a model can reason effectively in the presence of external evidence. Following the general spirit of MIRAGE (Chen et al., 2025), we assemble a specialized medical corpus from chunked medical textbooks provided by Jin et al. (2021) together with PubMed abstracts collected by Jin et al. (2023). For each question, we retrieve the top 10 passages from this corpus using BM25, and then distill both the response and reasoning trace conditioned on the retrieved evidence. The resulting examples therefore represent evidence-grounded clinical question answering.

PQA. PQA is derived from the `pqa_artificial` split of PubMedQA (Jin et al., 2019). To broaden the range of contextual and linguistic settings seen during post-training, we construct three variants using distinct prompt templates. The first preserves the original question-context pairing, where the question is adapted from the article title and the context is the corresponding abstract. The second and third variants introduce a variable number of distractor abstracts retrieved by BM25 using the question as the query. In one variant, article titles are included before each abstract, emphasizing

structured information extraction; in the other, titles are omitted, forcing greater semantic integration across the provided contexts. The final PQA dataset is obtained by mixing these three variants in equal proportions.

MQA(R). MQA(R) is a composite dataset created by balanced sampling from MQA and MQAR.

MQA(R)PQA. MQA(R)PQA is a composite dataset created by balanced sampling from MQA, MQAR, and PQA.

Open-source datasets from recent medical reasoning papers

HuaTuo (Chen et al., 2024). HuaTuo is an open-ended medical instruction dataset constructed from difficult MedQA and MedMCQA examples reformatted into free-response form. The responses are distilled from GPT-4o. For SFT, we use the `en_mix` subset of the `medical-o1-reasoning-SFT` split; for RL, we use the `medical-o1-verifiable-problem` split.

MedReason (Wu et al., 2025). MedReason is designed to encourage explainable medical problem solving in LLMs through the use of medical knowledge graphs. For each example, relevant portions of a medical knowledge graph are retrieved and used to guide GPT-4o in synthesizing a reasoning chain. The underlying questions are drawn from a diverse mixture of medical and general benchmarks, including MedQA, MedMCQA, PubMedQA, MMLU, MedXpertQA, and Humanity's Last Exam.

MedS3 (Jiang et al., 2025). MedS3 employs a self-evolution framework based on Monte Carlo Tree Search to generate and verify long-form reasoning trajectories. We use the resulting MedS3 dataset, whose prompts come from 16 medical and general-domain datasets and are paired with trajectories produced by the MedS3 pipeline.

ReasonMed (Sun et al., 2025). ReasonMed is produced by a multi-agent pipeline that generates, verifies, and iteratively revises reasoning trajectories distilled from multiple teacher models. Its curation pipeline reduces an initial pool of 1.75 million trajectories to 370k high-quality examples. The prompts are sourced from MedQA, MedMCQA, MMLU, and PubMedQA.

m23k (Huang et al., 2026). m23k is created through a multi-stage refinement process that includes difficulty filtering (to remove instances already solvable by standard base models), reasoning generation with DeepSeek-R1, and filtering based on final-answer correctness. Its prompts come from MedMCQA, MedQA, HeadQA, and PubMedQA.

We apply the same preprocessing pipeline to all datasets, consisting of length filtering followed by sampling. For datasets introduced in this paper, we explicitly enforce balanced representation across prompt sources in the final dataset (for example, MQA contains 50% MedQA prompts and 50% MedMCQA prompts). For open-source datasets from prior work, we instead perform stratified sampling according to the original source distribution, unless a single source overwhelmingly dominates the pool (e.g., more than 90% of examples). In such cases, we switch to equal sampling across all available sources in order to preserve source diversity.

A.1.2 Benchmark details

Final model performance is measured on twelve biomedical question-answering benchmarks. These benchmarks span different answer formats, domain scopes, and difficulty levels, allowing us to evaluate generalization across both biomedical knowledge recall and clinical reasoning.

BioASQ-Y/N (Tsatsaronis et al., 2015; Krithara et al., 2023). Following Xiong et al. (2024), we use a modified version of the BioASQ Task B machine reading comprehension benchmark. The benchmark contains 618 binary (Yes/No) questions drawn from the 2019-2023 competition cycles. In contrast to the original task formulation, which provides gold literature snippets, the modified version removes these supporting snippets, requiring the model to answer using only parametric biomedical knowledge.

GPQA (Biology) (Rein et al., 2024). This benchmark consists of graduate-level biology multiple-choice questions authored by domain experts.

MedBullets (Chen et al., 2025). MedBullets is a recently introduced medical licensing exam-style benchmark composed of multiple-choice questions based on realistic clinical scenarios. Chen et al. (2025) report that it is more difficult than older benchmarks such as MedQA, likely due in part to

its more recent curation. We evaluate both the 4-option and 5-option versions, with the latter adding an extra distractor choice to increase difficulty.

MedMCQA (Pal et al., 2022). MedMCQA is a large-scale multiple-choice medical QA benchmark derived from Indian medical entrance examinations.

MedQA (Jin et al., 2021). MedQA contains medical licensing examination questions collected from multiple geographic regions. We use the US subset.

MedXpertQA (Zuo et al., 2025). MedXpertQA is curated from medical professional and specialty board examinations through a pipeline involving AI-based filtering, question rewriting, and physician review. Each question has 10 answer options, which is substantially more than is typical in medical QA benchmarks. In addition, the questions cover a broad range of body systems and specialties, making this benchmark relatively challenging.

MMLU (Medicine) (Hendrycks et al., 2021). We use the six medicine-related MMLU subjects: anatomy, clinical knowledge, college biology, college medicine, medical genetics, and professional medicine.

MMLU-Pro (Biology & Health) (Wang et al., 2024b). MMLU-Pro is a more challenging variant of MMLU. We use its biology and health subsets.

PubMedQA (Jin et al., 2019). PubMedQA is a biomedical QA benchmark in which PubMed abstracts are paired with question statements derived from article titles, and the task is to determine whether the abstract entails the statement (yes/no/maybe).

PubMedQA (Questions Only). In the **questions-only** setting, the model receives the question statement without any accompanying abstract. This isolates parametric biomedical knowledge.

PubMedQA (With Distractor Abstracts). In the **distractor** setting introduced in this work, each question is paired with its gold abstract along with several distractor abstracts. This setting requires the model to identify useful evidence in addition to producing the correct answer. Distractor abstracts are retrieved with BM25 using the question statement as the query.

To keep evaluation costs tractable, we evaluate on fixed subsets rather than the complete benchmark suites. Unless stated otherwise, we randomly sample 100 examples per benchmark from the test split, or from the validation split if no test split is available. Two benchmarks are exceptions to this default: GPQA (Biology), which contains 78 examples, and MMLU (Medicine), which contains 123 examples after aggregating the six medicine-related subjects. Altogether, the final evaluation suite contains 1,201 prompts.

A.2 Sampling from the Configuration Space

To build the offline corpus, we sample 600 configurations from the search space using a scheme that guarantees equal coverage of every possible value of each hyperparameter. This design avoids concentrating too heavily on a narrow region of the search space and increases the diversity of the training data available to both the offline ranker and the early-stop regressor.

We further increase diversity in the offline corpus by varying dataset scale across training stages. For SFT, we use subsets of 1000, 2000, and 4000 samples. For RL, we vary the training horizon across 50, 100, and 200 steps, adjusting the amount of RL data accordingly. As a result, the offline corpus captures variation not only in hyperparameter settings but also in the scale of the underlying training procedure, thereby exposing downstream models to a broader range of optimization behaviors.

B Methodological Details

B.1 Offline Surrogate Model

This section provides additional implementation details for the offline ranker that provides guidance during online optimization.

B.1.1 Feature Construction and Selection

For each dataset-configuration pair, we construct a heterogeneous feature representation that jointly describes the configuration and the dataset. The full initial feature set is divided into four groups:

- **Configuration features (13).** These represent the post-training hyperparameter choices, including learning rates, batch sizes, numbers of epochs, and other discrete design decisions. They correspond to the configuration space in Table 3, with the base model one-hot encoded.
- **Simple dataset features (26).** These are coarse descriptive statistics such as token

counts, vocabulary size, mean sequence length, and average embedding distances, intended to summarize basic structural properties of each dataset.

- **Rich dataset features (82).** These capture higher-level linguistic and structural properties, such as lexical diversity, type–token ratio, and measures of instructional or textual complexity.
- **Dataset overlap features (204).** These quantify similarity between the training dataset and a set of target benchmarks using several similarity measures, including TF-IDF cosine similarity, embedding-based similarity, and Jaccard overlap.

All of these features are available before any post-training run is executed and do not depend on labels or post hoc evaluation signals from target datasets. To reduce redundancy and improve robustness, we apply a dataset-aware feature filtering procedure before training. For each dataset, we flag features whose variance falls below 10^{-2} times the median feature variance, as well as features that are highly correlated (Pearson correlation ≥ 0.97). Any feature flagged in at least 9 of the 10 datasets is removed. After filtering, the dimensionality decreases from 325 to 224 features, consisting of 13 configuration features, 22 simple dataset features, 59 rich dataset features, and 129 dataset overlap features.

B.1.2 Training and Validation Protocol

We train and evaluate the ranker under a leave-one-dataset-out (LODO) protocol in order to match the intended deployment setting. Each dataset is treated as a ranking group, with all associated configurations forming a ranked list. We also treat multiple subsets sampled from the same underlying dataset as belonging to the same ranking group, since their computed dataset characteristics are highly similar. In each fold, the model is trained on nine datasets and evaluated on the held-out tenth dataset.

We use XGBRanker as the base model because it can represent non-linear interactions between dataset features and configuration features while directly optimizing a ranking objective. To improve stability, we train an ensemble of three XGBRanker models with identical hyperparameters but different random seeds. Final scores and evaluation metrics are computed from the average of the ensemble predictions.

Hyperparameters are selected by grid search over standard XGBoost settings such as tree depth, learning rate, and number of estimators. We choose the final setting based on both the mean and minimum NDCG@5 across datasets so as to favor robust cross-dataset performance rather than strong average performance alone.

B.2 Early-Stop Regressor for Low-Cost Online Updates

This section provides implementation details for the early-stop regressor used in the online phase. In contrast to the offline ranker, whose scores can be computed without running new experiments, the early-stop regressor operates on partial signals observed from a small number of executed runs. Its purpose is to produce a low-cost proxy signal that can be used to update the online optimizer without repeatedly paying the cost of full RL training.

B.2.1 Feature Construction from SFT Training Dynamics

The regressor uses training dynamics recorded during SFT. We focus on SFT because it is much cheaper than RL while still yielding informative signals about downstream behavior: the eventual RL outcome depends heavily on the quality and stability of the SFT initialization, so early SFT dynamics can provide useful evidence about which configurations are likely to perform well in the full post-training pipeline.

For each run, we extract logged time-series metrics such as training loss, validation loss, gradient norm statistics, entropy-like quantities, and related training signals. From each trajectory, we compute summary statistics intended to capture both overall behavior and local stability. These include the mean, maximum, minimum, standard deviation, slope, and mean difference.

To avoid leaking information from beyond the early-stop point, all training-dynamic features are computed only on truncated prefixes of the trajectory. We define a *prefix window* covering the first 50% of the SFT trajectory. We also define a *late window* corresponding to the last 20% of that prefix window, which is intended to capture late-stage stability and local convergence behavior. For every metric and summary statistic, we compute features over both windows.

B.2.2 Feature Filtering

As with the offline ranker, we prune features to reduce redundancy and improve generalization. We remove features with near-zero variance (below 10^{-2} times the median feature variance) and features with high pairwise correlation (Pearson correlation ≥ 0.97). The same pruning rule is applied across datasets to reduce the risk of overfitting to logging artifacts specific to any one dataset.

B.2.3 Regression Target: Standardized Residual Correction

An important design choice is the regression target. The offline ranker yields a dataset-conditioned ordering over configurations, but its score is only an imperfect proxy for true end-to-end performance. At the same time, directly regressing on absolute performance is undesirable because the target may be dominated by dataset-level shifts in performance scale, i.e., some datasets may simply be easier or harder than others. Because the regressor is used to update the online procedure from noisy partial observations, we instead define the target as a *standardized residual correction*. Let $y_{d,c}$ denote the full end-to-end performance of configuration c on dataset d, and let $s^{(z)}_{d,c}$ denote the within-dataset z-normalized score assigned by the offline ranker. We first z-normalize the end-to-end performance:

$$y^{(z)}_{d,c} = \frac{y_{d,c} - \mu_d}{\sigma_d + \epsilon}$$

Here, μ_d and σ_d denote the normalization statistics used for dataset d in that fold; for training datasets they are estimated from the offline corpus, while the held-out dataset is never used to fit these statistics. We then define the regression target as

$$r_{d,c} = y^{(z)}_{d,c} - s^{(z)}_{d,c}$$

The regressor is trained to predict $r_{d,c}$ from configuration features, dataset features, and early-stop dynamic features only. In particular, the offline score $s^{(z)}_{d,c}$ is not supplied directly as an input feature.

Inference and evaluation. At inference time, we reconstruct the final score used for ranking or configuration selection by anchoring to the offline ranker and adding the predicted correction $\hat{r}_{d,c}$:

$$\hat{y}^{(z)}_{d,c} = s^{(z)}_{d,c} + \hat{r}_{d,c}$$

This makes the regressor a *conservative correction* mechanism: if the early-stop signals are weak or noisy, predictions remain close to the offline score, reducing the chance that BO is misled by unreliable pseudo-observations. For evaluation, we compare $\hat{y}^{(z)}_{d,c}$ against the true standardized performance $y^{(z)}_{d,c}$ using ranking and selection metrics.

B.2.4 Training and Validation Protocol

We use XGBRegressor models to map early-stop features to residual corrections. The regressor is trained and evaluated under the same LODO protocol as the offline ranker. To improve stability and obtain a crude uncertainty estimate for online BO updates, we train an ensemble of five regressors with identical hyperparameters but different random seeds. Their predictions are averaged, and the ensemble spread is used as a noise estimate when the regressor outputs are incorporated as pseudo-observations in the online phase. Hyperparameters are selected by grid search over standard XGBoost settings. Final model selection is based on the mean and minimum NDCG@5 across datasets.

B.3 Online Implementation

This section describes how post-training configurations are selected during the online phase. The BO loop combines prior guidance from the offline ranker with sequential updates derived from partial training signals and the early-stop regressor.

B.3.1 Gaussian Process over Residual Corrections

For each dataset–configuration pair (d, c), the offline ranker provides a standardized score $s^{(z)}_{d,c}$. As described above, this score induces a dataset-conditioned ordering over configurations and is available before any online evaluation is performed.

Rather than modeling absolute performance directly, we again work in standardized residual space, analogous to the early-stop regressor. In the online setting, however, we additionally introduce a dataset-specific offset term to improve adaptation. Concretely, we define

$$\tilde{r}_{d,c} = y^{(z)}_{d,c} - s^{(z)}_{d,c} - b_d$$

where $y^{(z)}_{d,c}$ is standardized end-to-end performance, $s^{(z)}_{d,c}$ is the standardized offline ranker score, and b_d is a dataset-specific offset. The GP then models a latent function $f_d(c)$ such that

$$f_d(c) \sim \mathcal{GP}(0, k(c, c'))$$

By construction, the GP therefore has zero mean in residual space. We use a kernel with automatic relevance determination (ARD) across feature dimensions, allowing different features and hyperparameters to operate on different characteristic length scales. This is desirable because different components of the post-training pipeline can influence optimization behavior at very different granularities.

B.3.2 Updating the GP

Warm-start Initialization. We initialize BO with a diverse set of configurations chosen using the offline ranker scores. The goal is to improve early coverage of the search space and reduce the risk of premature convergence.

Acquisition Function. At BO iteration t, let $\mu_{f,t}(c)$ and $\sigma_{f,t}(c)$ denote the GP posterior mean and standard deviation for the residual correction. The standardized performance estimate is

$$\hat{y}^{(z)}_{d,c,t} = w_t \, s^{(z)}_{d,c} + b_{d,t} + \mu_{f,t}(c)$$

where $b_{d,t}$ is the current estimate of the dataset-specific offset and $w_t \in [0, 1]$ is a tempering coefficient that controls how strongly the search is anchored to the offline ranker. We linearly anneal w_t from 1.0 to 0.7, so that earlier iterations rely more heavily on transferable offline structure, while later iterations are allowed to adapt more strongly to dataset-specific evidence. We use an Upper Confidence Bound (UCB) acquisition function:

$$\alpha_t(c) = \hat{y}^{(z)}_{d,c,t} + \beta_t \, \sigma_{f,t}(c)$$

where β_t governs the exploration–exploitation trade-off. Because the offline score does not vary across BO iterations, exploration is driven by uncertainty in the residual correction, while the overall ranking remains tied to the offline guidance. We use a decaying schedule for β_t, starting with stronger exploration and gradually shifting toward exploitation as evidence accumulates.

Early-Stop Updates. At each BO iteration, the selected configuration is partially evaluated by running SFT to a predefined early-stop point. The resulting training dynamics are fed into the early-stop regressor (Appendix B.2), which outputs a predicted pre-bias residual correction $\hat{r}_{d,c}$. We store these pseudo-observations in their pre-bias form. After updating the dataset-specific offset $b_{d,t}$, the GP is refit on centered targets:

$$\hat{r}_{d,c} - b_{d,t} = f_d(c) + \eta, \quad \eta \sim \mathcal{N}(0, \sigma^2_{\text{ens}})$$

where σ^2_{ens} is the ensemble variance of the regressor prediction, subject to a minimum variance floor. This separation allows the offset term to capture dataset-wide miscalibration while the GP models configuration-specific deviations.

Dataset-specific Offset. To account for dataset-level mismatch between offline guidance and target-dataset behavior, we maintain a dataset-specific offset b_d. Let Δ_i denote the i-th stored pre-bias residual pseudo-observation. After t online observations have been collected, we update the offset using the inverse-variance-weighted mean

$$b_{d,t} = \frac{\sum_{i=1}^{t} \Delta_i / v_i}{\sum_{i=1}^{t} 1/v_i},$$

where v_i is the observation variance associated with Δ_i. The GP at iteration t is then refit on centered targets $(\Delta_i - b_{d,t})$, so that $b_{d,t}$ absorbs dataset-wide shift while the latent function $f_d(c)$ models configuration-specific departures.

C Evaluation Methodology

C.1 End-to-end Evaluation

To ensure a controlled comparison across LODO splits, we define a fixed evaluation-scale version of each of the ten held-out datasets. Specifically, we fix the number of SFT prompts across target datasets and likewise fix the number of RL prompts. We refer to these fixed-scale dataset versions as the *canonical distributions*. Dataset meta-features are computed on these canonical distributions, and all methods and baselines are evaluated under the same data configuration to ensure a fair comparison.

All training workflows are implemented using the HuggingFace library, with `SFTTrainer` for supervised fine-tuning and `GRPOTrainer` for reinforcement learning. For SFT, we use a cosine learning-rate schedule with warmup ratio 0.1. For GRPO, we use the `dr_grpo` loss with $\epsilon_{\text{high}} = 0.28$. Across all tasks, the reward is defined as a composite rule-based signal consisting of:

- Format reward: a small incentive for following the requested output structure
- Accuracy reward: a binary reward based on exact match with the ground-truth answer

To enable automatic answer parsing, we append the instruction "At the end of your response, output `{answer_format}` where X is your chosen answer." to all RL prompts. To make the model less sensitive to a single answer-format instruction, we vary `{answer_format}` across prompts using eight distinct formats, including `\boxed{X}` and `**Answer: X**`.

We conduct all evaluations with the **lm-evaluation-harness** framework, which provides a standardized interface across heterogeneous tasks. All benchmarks use the same prompt template (Prompt 1) and require a free-form response ending in a final answer. In post-processing, we apply a regex-based filter to extract the first short-form answer token following the delimiter "Answer:". We then compute case- and punctuation-insensitive exact-match accuracy against the gold label. Generation settings are kept consistent across benchmarks, including stochastic decoding, a fixed maximum generation length, and standard end-of-sequence stopping rules. Because decoding is stochastic, all benchmark results are averaged over three random seeds.

Prompt 1: Evaluation prompt template

Solve the following question step-by-step. At the end of your answer, output "Answer: X" where X is the option corresponding to your final answer.

{question}
A: ...
B: ...
C: ...
D: ...
(additional options if any)

C.2 Baseline Implementations

This section describes the implementation details of the baselines.

C.2.1 Static & Zero-Shot Baselines

Random (Single run). We uniformly sample one configuration from the search space and execute it.

LLM Recommendation (Description-only). For this baseline, we query GPT 5.2 using the prompt shown in Figure C.1 and obtain a single recommended configuration for each dataset. The prompt includes only the task description together with a basic summary of the unseen dataset.

LLM Recommendation (History-conditioned). For this baseline, we query GPT 5.2 using the prompt in Figure C.2 and obtain one recommendation per dataset. To provide historical context, we retrieve neighboring datasets by cosine similarity to the target dataset in the joint meta-feature space.

Global. For each held-out dataset, we define the "global best" configuration as the single hyperparameter setting with the highest absolute performance across the other nine datasets. That configuration is then evaluated on the held-out dataset. This baseline therefore benefits from prior experimental evidence, but it does not adapt to target-dataset-specific properties such as training size, which may vary across the offline corpus.

Offline-only (Ranker). This baseline measures the zero-shot transfer ability of the offline ranker. For each held-out dataset, the ranker scores all $34{,}992$ configurations in the search space. We then run a single full-fidelity evaluation on the highest-ranked configuration and report its final accuracy.

C.2.2 Sequential Baselines

With the exception of BOHB, all sequential baselines are evaluated in both full-training and early-stop variants. The full-training variants execute the entire pipeline (SFT, RL, and evaluation) and use the resulting end-to-end accuracy as the optimization signal. The early-stop variants instead rely on a separate regressor trained to predict final performance $y_{d,c}$ directly, rather than residual corrections, since these baselines do not have access to the offline ranker used by our method. Every sequential method is given a fixed budget of 15 evaluations. SMAC and TPE are warm-started with the first five configurations sampled by Random Search. In the full-training setting these initial points use ground-truth scores, whereas in the early-stop setting they use regressor-predicted scores.

Random Search. We sample 15 configurations uniformly at random from the search space and evaluate them.

BOHB (Falkner et al., 2018). We use the official `HpBandSter` implementation. The budget b is defined in terms of RL-stage training fidelity because SFT contributes relatively little to total computational cost. We set the minimum and maximum budgets to $b_{\min} = 1$ and $b_{\max} = 4$, corresponding respectively to 25% and 100% of the full RL training horizon, and use elimination rate $\eta = 2$.

Prompt 2: LLM Recommendation (Description-only) Template

You are an expert researcher executing an LLM post-training workflow consisting of supervised fine-tuning (SFT) followed by reinforcement learning (RL). You are tasked with finding a suitable hyperparameter configuration for this workflow. You are provided with the objective of the post-training workflow, the configuration space, and some characteristics about the dataset you will be using.

Objective

The outcome of the post-training should be a medical reasoning model. Its performance will be evaluated on a series of medical benchmarks, including MedQA, MedMCQA, PubMedQA, MedBullets, MMLU (Med), MMLU-Pro (Biology/Health) GPQA (Biology), MedXpertQA and BioASQ. You should aim to maximize performance on this set of benchmarks. You cannot run experiments or use historical results; decide only from the information given.

Configuration space

- `base_model`: {L1 (Llama-3.2-1B-Instruct), L3 (Llama-3.2-3B-Instruct), L8 (Llama-3.1-8B-Instruct), Q1 (Qwen2.5-1.5B-Instruct), Q3 (Qwen2.5-3B-Instruct), Q8 (Qwen2.5-7B-Instruct)}
- `sft_lr`: $\{1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$, `sft_batch`: {32, 64, 128}, `sft_epoch`: {1, 2, 3}
- `rl_lr`: $\{1 \times 10^{-6}, 2 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}\}$, `rl_batch`: {64, 128, 256}, `rl_beta`: {0, 0.05, 0.1}, `rl_rollouts`: {8, 16}, `rl_temp`: {0.7, 0.9, 1.1}

Dataset characteristics

SFT Stage:

- prompt sources: ⟨src_sft⟩
- n_samples: ⟨data_n_sft_samples⟩
- prompt_len_tokens: mean=$\langle \mu_{p,sft} \rangle$, sd=$\langle \sigma_{p,sft} \rangle$
- completion_len_tokens: mean=$\langle \mu_{c,sft} \rangle$, sd=$\langle \sigma_{c,sft} \rangle$
- completion_source: ⟨comp_src⟩
- task_format: ⟨fmt_sft⟩
- num_choices_min: ⟨min_c⟩, max: ⟨max_c⟩

RL Stage:

- prompt sources: ⟨src_rl⟩
- n_samples: ⟨data_n_rl_samples⟩
- prompt_len_tokens: mean=$\langle \mu_{p,rl} \rangle$, sd=$\langle \sigma_{p,rl} \rangle$
- task_format: ⟨fmt_rl⟩
- num_choices_min: ⟨min_c⟩, max: ⟨max_c⟩
- reward_type: accuracy (exact match)

Your recommendation

Based on the above configuration space and task information, recommend a suitable hyperparameter configuration. All hyperparameter values must be chosen from the configuration space. Format your recommendation in a single JSON object:

```
{
    "base_model": "...",
    "sft_lr": "...",
    "sft_batch": "...",
    "sft_epoch": "...",
    "rl_lr": "...",
    "rl_batch": "...",
    "rl_beta": "...",
    "rl_rollouts": "...",
    "rl_temp": "..."
}
```

Figure C.1: Prompt template used for LLM Recommendation (Description-only) baseline. Values enclosed in angled brackets are dependent on the unseen dataset.

Prompt 3: LLM Recommendation (History-conditioned) Template

You are an expert researcher executing an LLM post-training workflow consisting of supervised fine-tuning (SFT) followed by reinforcement learning (RL). You are tasked with finding a suitable hyperparameter configuration for this workflow. You are provided with the objective of the post-training workflow, the configuration space, some characteristics about the dataset you will be using, and the results of some previous experiments using similar datasets.

Objective

The outcome of the post-training should be a medical reasoning model. Its performance will be evaluated on a series of medical benchmarks, including MedQA, MedMCQA, PubMedQA, MedBullets, MMLU (Med), MMLU-Pro (Biology/Health) GPQA (Biology), MedXpertQA and BioASQ. You should aim to maximize performance on this set of benchmarks. You cannot run new experiments; decide only from the information given.

Configuration space

- `base_model`: {L1 (Llama-3.2-1B-Inst), L3 (Llama-3.2-3B-Inst), L8 (Llama-3.1-8B-Inst), Q1 (Qwen2.5-1.5B-Inst), Q3 (Qwen2.5-3B-Inst), Q8 (Qwen2.5-7B-Inst)}
- `sft_lr`: $\{1 \times 10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}\}$, `sft_batch`: {32, 64, 128}, `sft_epoch`: {1, 2, 3}
- `rl_lr`: $\{1 \times 10^{-6}, 2 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}\}$, `rl_batch`: {64, 128, 256}, `rl_beta`: {0, 0.05, 0.1}, `rl_rollouts`: {8, 16}, `rl_temp`: {0.7, 0.9, 1.1}

Dataset characteristics

SFT Stage:

- prompt sources: $\langle$sft_prompt_sources$\rangle$
- n_samples: 2000
- prompt_len_tokens: mean=$\langle \mu_{p,s} \rangle$, sd=$\langle \sigma_{p,s} \rangle$
- completion_len_tokens: mean=$\langle \mu_{c,s} \rangle$, sd=$\langle \sigma_{c,s} \rangle$
- completion_source: $\langle$teacher_model$\rangle$
- task_format: $\langle$task_format$\rangle$

RL Stage:

- prompt sources: $\langle$rl_prompt_sources$\rangle$
- n_samples: 800
- prompt_len_tokens: mean=$\langle \mu_{p,r} \rangle$, sd=$\langle \sigma_{p,r} \rangle$
- task_format: $\langle$task_format$\rangle$
- reward_type: accuracy (exact match)

Nearest neighbor instances (historical results from other datasets)

Neighbor instance i (similarity=rank i/9 among training datasets)

- **Dataset summary:**
  - Prompt sources: $\langle \text{src}_i \rangle$, Task format: $\langle \text{fmt}_i \rangle$
  - SFT: $n = \langle n_{s,i} \rangle$, prompt_len_mean=$\langle \mu_{p,s,i} \rangle$, completion_source=$\langle \text{teacher}_i \rangle$
  - RL: $n = \langle n_{r,i} \rangle$, prompt_len_mean=$\langle \mu_{p,r,i} \rangle$
- **Config 1:** `{"base_model": "...", "sft_lr": ..., "rl_lr": ..., ...}` [acc=$\langle \text{acc}_{i,1} \rangle$]
- **Config 2:** `{"base_model": "...", "sft_lr": ..., "rl_lr": ..., ...}` [acc=$\langle \text{acc}_{i,2} \rangle$]

. . . (Repeated for top-5 neighbors) . . .

Your recommendation

Based on the above configuration space and task information, recommend a suitable hyperparameter configuration. Format your recommendation in a single JSON object:

```
{
    "base_model": "...",
    "sft_lr": "...",
    "sft_batch": "...",
    "sft_epoch": "...",
    "rl_lr": "...",
    "rl_batch": "...",
    "rl_beta": "...",
    "rl_rollouts": "...",
    "rl_temp": "..."
}
```

Figure C.2: The full history-conditioned prompt template, utilizing meta-feature-based retrieval to provide historical experiment results (nearest neighbors) to the LLM surrogate.

Note that BOHB's fidelity notion is defined over RL-stage training budget, whereas our method uses SFT-truncated runs combined with an early-stop predictor. The two are therefore not identical notions of low-fidelity evaluation, but each reflects the most natural low-cost signal available to the corresponding method.

SMAC (Hutter et al., 2011). We use the SMAC3 framework (Lindauer et al., 2022), version 2.3.1.

TPE (Bergstra et al., 2011). We use the Optuna implementation (Akiba et al., 2019), version 4.7.0, with `TPESampler` in multivariate mode so as to better capture dependencies among hyperparameters.

D Extended Results

This section provides the dataset-level breakdowns for all aggregated results presented in the main paper. We report performance (average accuracy on all test benchmarks), regret, cost, offline surrogate quality, and sensitivity to the online evaluation budget.

D.1 End-to-End Configuration Selection Performance

Tables D.1, D.2 and D.3 report dataset-level performance, regret and cost. Although certain datasets prove harder than others, the results overall demonstrate that AutoPipe remains competitive across the held-out datasets, while requiring substantially lower cost than full-evaluation sequential baselines.

D.2 Offline Component Analysis

Table D.4 reports dataset-level offline surrogate ranking performance (in terms of ranking metrics such as regret, recall@k, nDCG, pairwise accuracy and Spearman ρ) for both the offline ranker and the early stop regressor.

D.3 Sensitivity to Budget

Figure D.1 shows how the performance of the best observed configuration varies as the online evaluation budget increases. Across datasets, AutoPipe generally reaches competitive performance within a small number of online evaluations, suggesting that the offline ranker effectively guides the initial search space.

Table D.1: Configuration selection performance across ten held-out datasets under a fixed number of online evaluations. Performance is calculated as the average benchmark accuracy. Bold indicates the best performance per dataset; italics indicate the second-best.

	Datasets										
Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
Static & Zero-Shot Baselines											
Random (Single run)	**0.620**	0.383	0.503	0.370	0.570	0.141	0.342	0.425	0.088	0.468	0.391
LLM Rec. (Description-only)	0.513	0.544	0.423	0.547	0.477	0.371	0.088	0.018	0.052	0.035	0.307
LLM Rec. (History-conditioned)	0.465	**0.630**	0.435	0.558	0.461	0.382	0.532	0.410	0.431	0.117	0.442
Global	0.556	*0.628*	*0.632*	0.624	*0.642*	0.622	0.454	**0.659**	0.018	0.451	0.529
Offline-only (Ranker)	0.493	0.622	0.611	0.611	0.577	*0.634*	0.542	0.620	0.276	*0.591*	0.558
Sequential Baselines											
Random Search											
ES	**0.620**	0.416	0.554	0.524	0.570	0.512	0.513	0.480	0.227	0.042	0.446
Full	**0.620**	0.584	0.583	0.578	0.570	0.624	0.513	0.548	0.502	0.526	0.565
BOHB	0.612	0.610	0.541	0.599	0.620	0.633	0.582	0.629	0.494	0.576	**0.590**
SMAC											
ES	**0.620**	0.611	**0.644**	0.494	0.570	0.385	0.426	0.433	0.255	**0.606**	0.504
Full	**0.620**	0.610	0.602	0.626	0.573	**0.647**	0.575	0.571	*0.502*	0.473	0.580
TPE											
ES	**0.620**	0.601	0.591	0.601	0.589	0.430	*0.602*	0.600	0.227	0.588	0.545
Full	**0.620**	0.612	0.592	0.476	**0.654**	0.615	0.551	0.606	**0.505**	0.468	0.570
Ablations											
No Residual Formulation	0.601	0.614	0.620	0.589	0.577	*0.634*	0.320	0.620	0.352	0.234	0.516
No BO	0.601	0.213	0.611	*0.679*	0.565	0.607	0.542	0.597	0.259	*0.591*	0.527
No Offline	0.599	0.620	0.515	0.515	0.415	0.478	0.581	0.533	0.305	0.406	0.497
Sensitivity Analysis											
25%	*0.617*	0.614	0.611	0.589	0.577	0.589	0.542	0.641	0.276	*0.591*	0.565
75%	0.601	0.614	0.611	**0.683**	0.577	0.607	0.542	*0.655*	0.344	*0.591*	*0.583*
Ours (Two-phase, ES)	*0.617*	0.614	0.619	**0.683**	0.603	*0.634*	**0.606**	0.641	0.344	0.536	**0.590**

Table D.2: Configuration selection regret across ten held-out datasets under a fixed number of online evaluations. Regret is calculated relative to the best-performing configuration in the reference pool for each dataset. Bold indicates the best performance (lowest regret) per dataset; italics indicate the second-best.

	Datasets										
Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
Static & Zero-Shot Baselines											
Random (Single run)	**0.003**	0.247	0.141	0.313	0.084	0.506	0.264	0.234	0.417	0.157	0.237
LLM Rec. (Description-only)	0.110	0.086	0.221	0.136	0.177	0.276	0.518	0.641	0.453	0.590	0.321
LLM Rec. (History-conditioned)	0.158	**0.000**	0.209	0.125	0.193	0.265	0.074	0.249	0.074	0.508	0.186
Global	0.067	*0.002*	*0.012*	0.059	*0.012*	0.025	0.152	**0.000**	0.487	0.174	0.099
Offline-only (Ranker)	0.130	0.008	0.033	0.072	0.077	*0.013*	0.064	0.039	0.229	*0.034*	0.070
Sequential Baselines											
Random Search											
ES	**0.003**	0.214	0.090	0.159	0.084	0.135	0.093	0.179	0.278	0.583	0.182
Full	**0.003**	0.046	0.061	0.105	0.084	0.023	0.093	0.111	0.003	0.099	0.063
BOHB	0.011	0.020	0.103	0.084	0.034	0.014	0.024	0.030	0.011	0.049	**0.038**
SMAC											
ES	**0.003**	0.019	**0.000**	0.189	0.084	0.262	0.180	0.226	0.250	**0.019**	0.123
Full	**0.003**	0.020	0.042	0.057	0.081	**0.000**	0.031	0.088	*0.003*	0.152	0.048
TPE											
ES	**0.003**	0.029	0.053	0.082	0.065	0.217	*0.004*	0.059	0.278	0.037	0.083
Full	**0.003**	0.018	0.052	0.207	**0.000**	0.032	0.055	0.053	**0.000**	0.157	0.058
Ablations											
No Residual Formulation	0.022	0.016	0.024	0.094	0.077	*0.013*	0.286	0.039	0.153	0.391	0.112
No BO	0.022	0.417	0.033	*0.004*	0.089	0.040	0.064	0.062	0.246	*0.034*	0.101
No Offline	0.024	0.010	0.129	0.168	0.239	0.169	0.025	0.126	0.200	0.219	0.131
Sensitivity Analysis											
25%	*0.006*	0.016	0.033	0.094	0.077	0.058	0.064	0.018	0.229	*0.034*	0.063
75%	0.022	0.016	0.033	**0.000**	0.077	0.040	0.064	*0.004*	0.161	0.034	0.045
Ours (Two-phase, ES)	*0.006*	0.016	0.025	**0.000**	0.051	*0.013*	**0.000**	0.018	0.161	0.089	**0.038**

Table D.3: Configuration selection cost across ten held-out datasets under a fixed number of online evaluations. Cost is expressed in terms of H200 hours.

	Datasets										
Method	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
Static & Zero-Shot Baselines											
Random (Single run)	3.62	2.15	4.19	1.98	6.93	0.49	1.19	2.61	6.72	6.36	3.62
LLM Rec. (Description-only)	7.95	8.55	8.73	11.56	10.33	3.22	5.52	7.40	5.04	7.74	7.60
LLM Rec. (History-conditioned)	4.13	5.30	4.15	6.57	11.47	3.05	5.37	2.88	3.83	9.09	5.59
Global	3.58	6.14	3.77	4.67	4.52	3.77	3.62	2.76	3.48	4.19	4.05
Offline-only (Ranker)	8.80	5.32	5.83	7.47	10.52	3.57	5.32	4.50	3.92	6.33	6.16
Sequential Baselines											
Random Search											
ES	4.71	2.64	3.78	5.17	7.94	3.82	3.84	2.73	7.94	6.91	4.95
Full	56.71	67.76	56.18	60.85	79.14	34.03	49.61	38.47	68.52	74.04	58.53
BOHB	99.13	100.02	128.24	116.07	107.88	69.23	72.15	62.22	56.64	143.80	95.54
SMAC											
ES	5.11	3.31	10.87	6.91	8.30	4.83	7.46	3.20	5.60	4.36	5.99
Full	80.51	51.92	60.16	104.02	112.49	33.76	36.47	33.36	49.29	77.97	63.99
TPE											
ES	4.99	5.92	6.09	6.50	8.05	3.44	4.52	5.70	8.51	8.98	6.27
Full	55.38	74.29	78.72	64.18	90.93	75.67	42.19	50.82	55.60	80.38	66.81
Ablations											
No Residual Formulation	10.13	6.36	6.19	9.44	12.45	5.29	7.50	5.92	7.18	8.35	7.88
No BO	10.86	12.16	7.89	11.29	12.33	6.96	7.17	4.73	8.82	9.15	9.14
No Offline	8.15	3.91	3.90	9.00	6.24	7.37	5.57	4.84	3.39	3.64	5.60
Sensitivity Analysis											
25%	9.87	5.41	6.63	8.60	11.64	3.64	6.37	5.32	5.30	7.46	7.02
75%	11.86	7.66	8.27	11.75	13.54	7.52	8.29	6.61	10.58	9.74	9.58
Ours (Two-phase, ES)	10.95	6.63	6.98	10.87	12.03	5.46	7.72	6.46	9.18	10.88	8.71

Table D.4: Offline surrogate ranking performance across ten held-out datasets under LODO evaluation.

	Datasets										
Metric	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Avg**
Offline Ranker											
Regret	0.060	0.057	0.048	0.021	0.032	0.042	0.585	0.070	0.000	0.183	0.110
Recall@1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.100
Recall@3	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	1.000	0.000	0.300
Recall@5	0.000	1.000	0.000	0.000	1.000	1.000	0.000	0.000	1.000	0.000	0.400
Recall@10	0.000	1.000	1.000	0.000	1.000	1.000	1.000	1.000	1.000	0.000	0.700
nDCG@5	0.743	0.954	0.943	0.980	0.958	0.861	0.609	0.937	0.706	0.836	0.853
nDCG@10	0.811	0.959	0.952	0.964	0.966	0.908	0.697	0.951	0.617	0.868	0.869
Pairwise Accuracy	0.789	0.724	0.814	0.796	0.776	0.745	0.721	0.755	0.679	0.705	0.750
Spearman ρ	0.751	0.623	0.830	0.778	0.733	0.665	0.632	0.721	0.523	0.581	0.684
Early Stop Regressor											
Regret	0.060	0.057	0.048	0.021	0.032	0.299	0.022	0.070	0.000	0.183	0.079
Recall@1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.100
Recall@3	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	1.000	0.000	0.300
Recall@5	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	1.000	0.000	0.300
Recall@10	0.000	1.000	1.000	0.000	1.000	1.000	0.000	0.000	1.000	0.000	0.500
nDCG@5	0.722	0.918	0.940	0.980	0.978	0.796	0.702	0.939	0.800	0.840	0.862
nDCG@10	0.798	0.925	0.952	0.926	0.956	0.846	0.778	0.887	0.676	0.861	0.861
Pairwise Accuracy	0.778	0.750	0.830	0.797	0.788	0.767	0.753	0.742	0.709	0.721	0.764
Spearman ρ	0.720	0.681	0.845	0.767	0.766	0.714	0.691	0.676	0.581	0.622	0.706

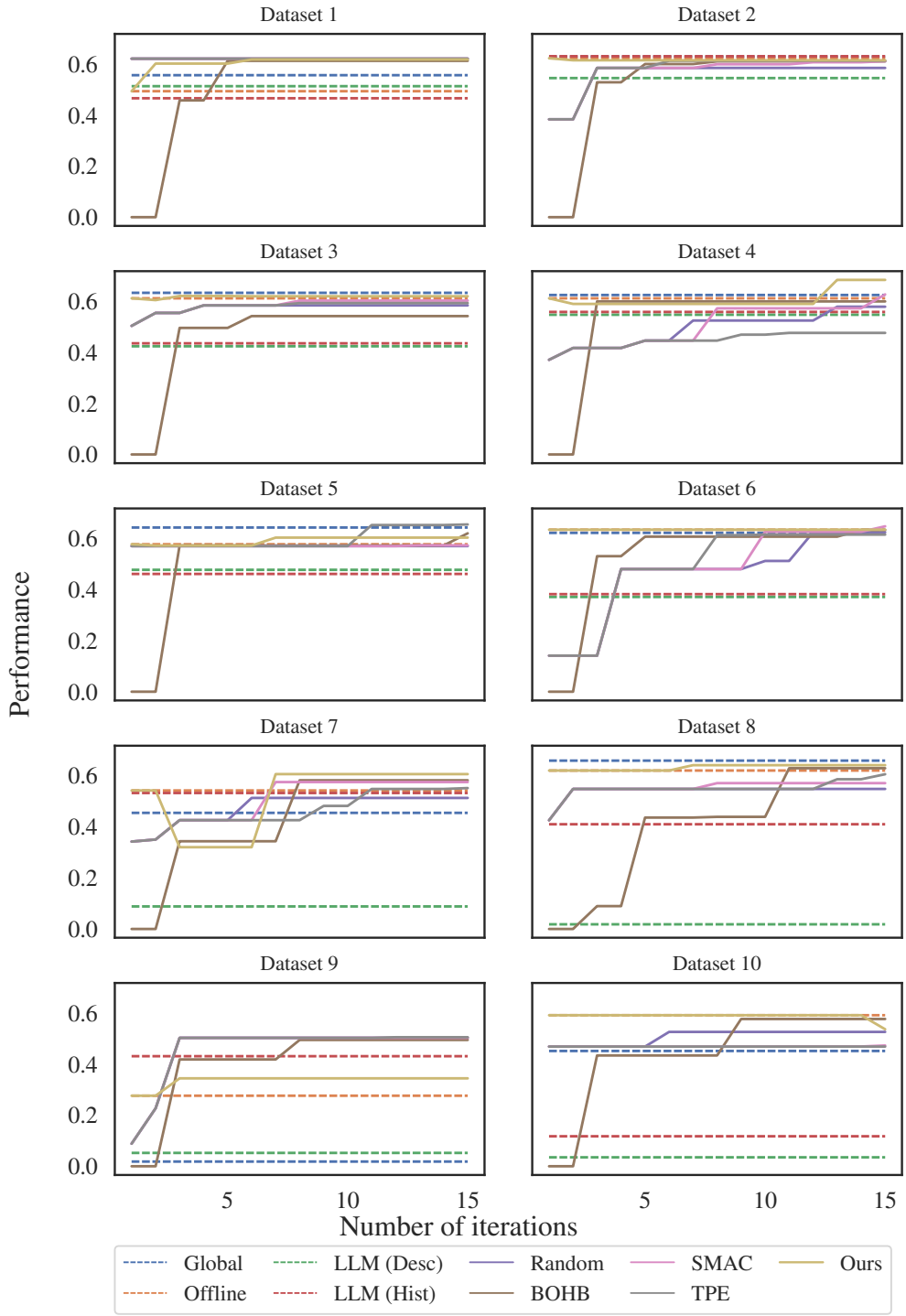

Figure D.1: Sensitivity of configuration selection performance to the online evaluation budget across ten held-out datasets.