Treating Hyperparameters as Interventions: Task-Invariant Representation Learning for Transferable HPO

Mengyang Li
Tianjin Key Laboratory of Wireless Mobile Communications
and Power Transmission, Tianjin Normal University
Tianjin, China
limengyang@tjnu.edu.cn

Ou Wu*
HIAS, University of Chinese Academy of Sciences
Hangzhou, China
wuou@ucas.ac.cn

Abstract

We study transfer hyperparameter optimization (HPO) by treating configurations as controlled interventions and learning task representations from short training trajectories. We propose CA-HPO (Configuration-invariAnt HPO), a transfer HPO framework that learns a task representation that is constrained to be invariant across hyperparameter configurations of the same task and is trained to predict performance. We define the target as the solution of a constrained optimization problem over the encoder and predictor classes rather than as an identified latent variable, so the framework avoids the identifiability assumptions of causal representation learning. We prove three guarantees: the invariance violation of the penalized population minimizer is at most the price of invariance divided by the penalty weight, with recovery of the target in the limit; a task-level finite-sample excess-risk bound; and a counterfactual error bound in which the probe-averaging error scales with the within-task spread that the penalty controls. Once a one-off meta-training corpus is available, CA-HPO attains the highest final accuracy at the lowest search cost on four Llama-3-8B fine-tuning tasks, and on twelve classical benchmarks it achieves a 5.7 times geometric-mean online speedup over random search, against 2.0 times for HEBO and at most 2.1 times for multi-fidelity methods. Under matched probing signal and model capacity, removing the invariance constraint alone reduces this speedup to 2.6 times. The learned representations remain stable across four orders of magnitude of learning rate and recover a known task invariant in a controlled synthetic study, and task-wise HSIC permutation tests do not reject their independence from λ on 11 of the 12 classical benchmarks.

1 Introduction

Hyperparameter optimization (HPO) sits awkwardly between two roles. From one angle it is a black-box search problem and is studied with the tools of black-box optimization: Bayesian surrogates [Snoek et al., 2012, Shahriari et al., 2016, Cowen-Rivers et al., 2022, Li and Zhao, 2026a], multi-fidelity scheduling [Falkner et al., 2018, Jamieson and Talwalkar, 2016], and more recently

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

transformer-based meta-learners [Chen et al., 2022, Rakotoarison et al., 2024] and LLM-driven schemes [Liu et al., 2024a, 2025, Yang et al., 2024]. From another angle it is the meta-level of a learning problem: a hyperparameter λ is not really an unknown variable to be probed, it is a knob the practitioner deliberately turns. The two roles are usually fused in practice, and the cost shows up as expensive evaluations and poor cross-task transfer.

In this paper we take the second view seriously. We treat hyperparameters as *interventions* rather than as unknowns. Training dynamics [Li et al., 2025, 2026b] then mix a task *invariant*, such as loss landscape curvature and gradient noise structure, with an *intervention effect* caused by λ, which is under the practitioner's control. Once the two are separated, the task-invariant component can be summarized in a small representation estimated from short training trajectories, and predicting performance under any candidate λ becomes a lookup against that representation.

We make this precise through a constrained definition: among encoders that map short training trajectories to a vector, we ask for one whose output depends on the task alone, and is therefore independent of λ given the task, and which predicts performance across configurations as well as any such encoder in the class. The target is a minimizer of an optimization problem rather than an identified latent variable in a structural model. The distinction matters: identifiability-based formulations of causal representation learning [Khemakhem et al., 2020, Lippe et al., 2022, Brehmer et al., 2022] need strong, often untestable assumptions about the data-generating process, whereas our analysis needs boundedness and Lipschitz conditions on the function classes, plus compactness for the limit statement. For estimation, intrinsic task properties already manifest in the very first hundred steps of training, and we exploit this through a *micro-sensing* protocol: a small number of short probe runs at diverse λ estimate the task representation, after which HPO reduces to lookup against a learned counterfactual surrogate, with a local trust-region check that falls back to Bayesian optimization when a proposed candidate moves far from the configurations already evaluated on the target task. In standard HPO terms, micro-sensing is short-horizon learning-curve probing (a low-fidelity signal), the counterfactual predictor is a meta-surrogate, and the reliability check is a trust region. CA-HPO is therefore a low-fidelity meta-HPO method; what it adds to this family is a within-task constraint that removes configuration information from the task representation while retaining predictive information.

In this setting we prove three guarantees: for the penalized objective, the invariance violation of the population minimizer is at most the price of invariance divided by the penalty weight, and its limit points attain the target; the empirical minimizer over N meta-training tasks has excess risk bounded by a task-level Rademacher complexity plus an $\mathcal{O}(N^{-1/2})$ term; and the counterfactual prediction error splits into a probe-averaging term proportional to the within-task spread of the encoder, the surrogate's error at the task's mean representation, and aleatoric noise. We instantiate the framework as CA-HPO (Configuration-invariAnt HPO), with an iTransformer [Liu et al., 2024b] encoder, an MLP counterfactual predictor, and the reliability-guarded protocol above. On Llama-3-8B fine-tuning across four downstream tasks, CA-HPO attains the highest final accuracy of all methods at the lowest search cost; on twelve classical benchmarks it achieves a 5.7$\times$ geometric-mean online speedup over Random Search, against 2.0$\times$ for HEBO. These warm-start figures exclude the one-off offline cost of about 488 GPU-hours, which pays off only when amortized over many comparable tasks (Section 5.2). The gain is not explained by the probing signal alone: with the same probes and model capacity, dropping the invariance term reduces the speedup to 2.6$\times$, and multi-fidelity methods reach 1.5 to 2.1$\times$. Representations remain stable across four orders of magnitude of learning rate, transfer from vision to NLP tasks at 2.9$\times$ speedup over random search, and recover a known task invariant in a controlled synthetic setting, and task-wise HSIC permutation tests do not reject their independence from λ on 11 of 12 classical tasks.

Our contributions are:

- We treat λ as an intervention and summarize the task by what is invariant to it, made precise through a constrained definition that does not rely on identifiability assumptions.
- Three guarantees in this setting: a finite-penalty invariance bound with recovery in the limit, a task-level finite-sample excess-risk bound, and a counterfactual error bound that ties the probe budget to the invariance penalty.
- A complete instantiation as a low-fidelity meta-surrogate with a within-task representation constraint, with matched ablations and multi-fidelity comparisons that isolate the contribution of the

constraint, a controlled synthetic test against a known invariant, and task-wise independence tests of the encoder output against λ.

2 Problem Formulation

Setup. Let $p(\mathcal{T})$ be a distribution over tasks; each task $t \sim p(\mathcal{T})$ is specified by a dataset and loss. A hyperparameter configuration $\lambda \in \Lambda$ lives in a mixed discrete-continuous space (learning rates, batch sizes, scheduler types, and so on). Training under (t, λ) produces training dynamics $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^{T_{\text{total}} \times D}$, a multivariate time series of D statistics over the full run of T_{total} steps; we use $\mathbf{X}^{(T)}$ for the prefix of the first $T \ll T_{\text{total}}$ steps. The final performance is $y \in \mathbb{R}$, and HPO seeks $\lambda^*(t) = \arg\max_\lambda \mathbb{E}[y(t, \lambda)]$. Although multi-fidelity and transfer methods reduce the number of full-training evaluations, expensive evaluations remain a dominant cost in many large-model HPO pipelines.

Scope. Within each HPO problem, the base architecture, training objective, and optimizer family are fixed; the search space contains training hyperparameters such as learning rate, weight decay, batch size, dropout, scheduler type, warmup ratio, gradient clipping, augmentation intensity, and LoRA rank and alpha. Architectures may differ across tasks, but architecture itself is not searched, and LoRA rank changes adapter capacity within a fixed base model rather than providing evidence of invariance to architecture. How much predictive performance the invariance constraint costs depends on the chosen configuration family; when this price of invariance is high, the framework helps less, and Section 5.3 shows a controlled case of this.

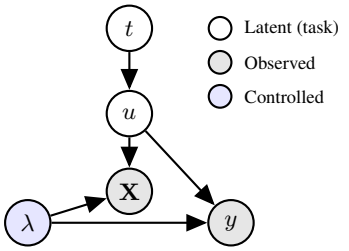

Figure 1: Motivating diagram for the invariance constraint in Definition 2.1. The constraint $u \perp \lambda \mid t$ corresponds to the independence between cause and intervention.

Target representation. Rather than postulate a hidden physical task variable that the encoder must recover, we define the target through the function classes we actually use. Let $\mathfrak{E}$ be a class of encoders $\mathcal{E} : \mathcal{X} \to \mathbb{R}^{d_u}$ and $\mathfrak{F}$ a class of predictors $f(u, \lambda)$. Configurations are drawn from a sampling distribution q on Λ, and $(\mathbf{X}, y)$ denotes the outcome of one training run under (t, λ), which is random through initialization and mini-batch sampling. The prediction risk is $R(\mathcal{E}, f) = \mathbb{E}[(f(\mathcal{E}(\mathbf{X}), \lambda) - y)^2]$ with $t \sim p(\mathcal{T})$ and $\lambda \sim q$. An encoder is *task-determined* if $\mathcal{E}(\mathbf{X}) = u_{\mathcal{E}}(t)$ almost surely for some function $u_{\mathcal{E}}$ of the task alone, that is, if its output changes neither with the configuration nor with the training randomness.

Definition 2.1 (Invariant predictive representation)**.** A pair $(\mathcal{E}^*, f^*) \in \mathfrak{E} \times \mathfrak{F}$ is an *invariant predictive pair* if $\mathcal{E}^*$ is task-determined and $(\mathcal{E}^*, f^*)$ minimizes R among all pairs whose encoder is task-determined. The *invariant predictive representation* of task t is $u^*(t) = u_{\mathcal{E}^*}(t)$. We write R^*_{inv} for the infimum of R over pairs with a task-determined encoder.

A task-determined encoder satisfies $\mathcal{E}(\mathbf{X}) \perp \lambda \mid t$, but not conversely: task-determinedness is a stronger sufficient condition for the requirement $u \perp \lambda \mid t$ that also removes variation due to training randomness. The content of the constraint is that a map reading the trajectory, which does depend on λ, must return the same vector for every configuration of the same task. Two clarifications bound what the definition says. First, u^* is not unique: any reparametrization that the predictor class can absorb, such as $u \mapsto -u$, gives another invariant predictive pair with the same risk, so our results concern risk and invariance rather than coordinates. Second, R^*_{inv} can exceed the Bayes risk of predicting y from (t, λ) when $\mathfrak{E}$ cannot represent enough task information; we make no claim of Bayes sufficiency.

Where the structural picture comes in. Suppose, hypothetically, that task t acts on $\mathbf{X}$ only through an intrinsic summary u independent of λ, while λ acts on $\mathbf{X}$ as a separate cause, and y depends on (u, λ) and noise. Figure 1 draws this picture, which motivates the constraint $u \perp \lambda \mid t$. The diagram serves only as motivation; the analysis in Section 3 does not require it to describe the true data-generating process.

3 Theoretical Analysis

We analyze the penalized form of Definition 2.1 that CA-HPO optimizes. For an encoder $\mathcal{E}$ let

$$P(\mathcal{E}) \;=\; \tfrac{1}{2}\,\mathbb{E}_t\,\mathbb{E}\big[\|\mathcal{E}(\mathbf{X}) - \mathcal{E}(\mathbf{X}')\|_2^2 \,\big|\, t\big] \;=\; \mathbb{E}_t\big[s_t^2(\mathcal{E})\big], \tag{1}$$

where $(\lambda, \mathbf{X})$ and $(\lambda', \mathbf{X}')$ are two independent runs of the same task with $\lambda, \lambda' \sim q$, and $s_t^2(\mathcal{E}) = \mathbb{E}[\|\mathcal{E}(\mathbf{X}) - \mathbb{E}[\mathcal{E}(\mathbf{X}) \mid t]\|_2^2 \mid t]$ is the within-task spread of the encoder output. $P(\mathcal{E}) = 0$ exactly when $\mathcal{E}$ is task-determined, and P upper-bounds the variation of the conditional mean, $V(\mathcal{E}) = \mathbb{E}_t[\mathrm{Var}_\lambda(\mathbb{E}[\mathcal{E}(\mathbf{X}) \mid t, \lambda])] \le P(\mathcal{E})$ (Lemma A.1). We minimize

$$\mathcal{L}_\beta(\mathcal{E}, f) \;=\; R(\mathcal{E}, f) \;+\; \beta\, P(\mathcal{E}), \tag{2}$$

whose empirical counterpart is the prediction and invariance part of the training objective in Section 4. The implementation additionally uses an auxiliary reconstruction loss, so Theorems 3.1 and 3.3 do not directly characterize minimizers of the full training objective (6). Let $R_0 = \inf_{\mathfrak{E}\times\mathfrak{F}} R$ be the unconstrained optimal risk; $R^*_{\mathrm{inv}} - R_0 \ge 0$ is the price of invariance.

Theorem 3.1 (Penalized population objective)**.** *Suppose that for every $\beta > 0$ the infimum of $\mathcal{L}_\beta$ over $\mathfrak{E} \times \mathfrak{F}$ is attained at $(\mathcal{E}_\beta, f_\beta)$, and that $R^*_{\mathrm{inv}} < \infty$. Then: (a) for every $\beta > 0$, $R(\mathcal{E}_\beta, f_\beta) \le R^*_{\mathrm{inv}}$ and $V(\mathcal{E}_\beta) \le P(\mathcal{E}_\beta) \le (R^*_{\mathrm{inv}} - R_0)/\beta$; (b) if, in addition, $\mathfrak{E} \times \mathfrak{F}$ is compact in a topology for which R and P are lower semicontinuous, then every limit point of $(\mathcal{E}_\beta, f_\beta)$ as $\beta \to \infty$ is an invariant predictive pair.*

Part (a) says the invariance violation decays as $1/\beta$, with the price of invariance as the constant; part (b) is the limit statement. Neither bounds a distance between encoders, which is not meaningful without fixing coordinates. Proofs of all results are in Appendix A. At the $\beta_1 = 1$ used in practice, approximate invariance is supported empirically: the plug-in estimate σ_u of s_t drops from 0.34 at $\beta_1 = 0$ to 0.042 at $\beta_1 = 1$ with little further change (Table 9).

Finite-sample excess risk. The meta-training data consist of N tasks $t_n \sim p(\mathcal{T})$ drawn independently, each with $m \ge 2$ runs at configurations $\lambda_{nj} \sim q$ drawn independently (our corpus uses Sobol sequences, which this analysis does not cover). Runs of the same task share t_n, so the independent units are the task blocks $Z_n = \{(\lambda_{nj}, \mathbf{X}_{nj}, y_{nj})\}_{j=1}^m$. The empirical objective is $\hat{\mathcal{L}}_\beta = N^{-1}\sum_n \ell_\beta(\mathcal{E}, f; Z_n)$, where the block loss ℓ_β averages the squared prediction error over the m runs of a task and adds β times the average of $\frac{1}{2}\|\mathcal{E}(\mathbf{X}_{nj}) - \mathcal{E}(\mathbf{X}_{nj'})\|_2^2$ over pairs $j \ne j'$; it is an unbiased estimate of $\mathcal{L}_\beta$ (Appendix A.3).

Assumption 3.2 (Bounded and Lipschitz classes)**.** $\|\mathcal{E}(\mathbf{X})\|_2 \le B_u$ for all $\mathcal{E} \in \mathfrak{E}$; $|f| \le B$ for all $f \in \mathfrak{F}$ and $|y| \le B$; and every $f \in \mathfrak{F}$ is L_f-Lipschitz in u for each fixed λ.

These can be enforced by output clipping and spectral normalization; our implementation uses layer normalization and weight decay, which encourage but do not enforce them.

Theorem 3.3 (Finite-sample excess risk)**.** *Under Assumption 3.2, let $(\hat{\mathcal{E}}, \hat{f})$ minimize $\hat{\mathcal{L}}_\beta$ over $\mathfrak{E} \times \mathfrak{F}$, let $\mathfrak{R}_N$ be the Rademacher complexity of the block-loss class $\{\ell_\beta(\mathcal{E}, f; \cdot)\}$ over N task blocks, and let $M = 4B^2 + 2\beta B_u^2$. With probability at least $1 - \delta$,*

$$\mathcal{L}_\beta(\hat{\mathcal{E}}, \hat{f}) - \mathcal{L}_\beta(\mathcal{E}_\beta, f_\beta) \;\le\; \varepsilon_N \;:=\; 2\,\mathfrak{R}_N + M\sqrt{\frac{2\log(2/\delta)}{N}}, \tag{3}$$

and consequently $R(\hat{\mathcal{E}}, \hat{f}) \le R^*_{\mathrm{inv}} + \varepsilon_N$ and $P(\hat{\mathcal{E}}) \le (R^*_{\mathrm{inv}} - R_0 + \varepsilon_N)/\beta$.

The proof is a uniform-convergence argument over task blocks: the sample size that matters is the number of tasks N, and the m runs per task enter only through the block loss. As usual, $(\hat{\mathcal{E}}, \hat{f})$ is an exact empirical minimizer, which stochastic training only approximates.

Counterfactual prediction error. At inference, the trained pair $(\hat{\mathcal{E}}, \hat{f})$ is fixed and k probes are run on a target task t. Let $\hat{u}_t$ be the probe average in (7), $\bar{u}_t = \mathbb{E}[\hat{\mathcal{E}}(\mathbf{X}) \mid t]$ its population counterpart, $s_t = s_t(\hat{\mathcal{E}})$, and $\mu(t, \lambda) = \mathbb{E}[y \mid t, \lambda]$.

Theorem 3.4 (Counterfactual prediction error)**.** *Under Assumption 3.2, let the probe configurations be drawn independently from q with independent runs. With probability at least $1-\delta$ over the probes, for every $\lambda \in \Lambda$ and a fresh run at (t, λ) with outcome $y(t, \lambda)$,*

$$|\hat{f}(\hat{u}_t, \lambda) - y(t, \lambda)| \leq L_f \left(s_t \sqrt{\tfrac{2\log(2/\delta)}{k}} + \tfrac{4B_u \log(2/\delta)}{k} \right) + \epsilon_t + |y(t, \lambda) - \mu(t, \lambda)|, \tag{4}$$

where $\epsilon_t = \sup_\lambda |\hat{f}(\bar{u}_t, \lambda) - \mu(t, \lambda)|$ is the error of the learned surrogate at the task's mean representation. We call the three terms (I) probe averaging, (II) surrogate error, and (III) aleatoric noise.

Term (I) is the only term the probes control, and its leading part is proportional to s_t. The penalty P averages s_t^2 over $p(\mathcal{T})$, and this average is what is bounded by Theorems 3.1 and 3.3; these bounds hold on average over tasks and do not bound s_t for an arbitrary new or out-of-distribution task. For a given task, a smaller s_t reduces the probe budget: $\|\hat{u}_t - \bar{u}_t\|_2 \leq \varepsilon$ holds with probability $1-\delta$ as soon as

$$k \;\geq\; \max\left\{ \frac{8\, s_t^2 \log(2/\delta)}{\varepsilon^2},\; \frac{8 B_u \log(2/\delta)}{\varepsilon} \right\}, \tag{5}$$

and the corresponding prediction error from term (I) is at most $L_f \varepsilon$. Term (II) is not bounded pointwise by our results; it contains the finite-β bias and the estimation error of $\hat{f}$, and it grows under domain shift. Term (III) is irreducible. The bound assumes independent probes; we use Latin Hypercube sampling and do not analyze it. A constant encoder has $P = 0$ but cannot keep term (II) small when tasks are distinguishable (Proposition A.2).

What is and is not testable. Task-determinedness implies that per-probe encodings are independent of λ within a task, which we probe with HSIC permutation tests (Section 5.3), and s_t is estimated by σ_u. The diagram in Figure 1 is not testable from observational HPO data, and we make no claim about it.

4 Methodology

CA-HPO instantiates the framework via two coupled neural modules: a representation module that estimates $\hat{u}$ from short trajectories, and a counterfactual module that predicts performance for a candidate λ given $\hat{u}$. Figure 2 depicts the architecture.

4.1 Architecture

The architecture operates on optimization dynamics $\mathbf{X}^{(T)} \in \mathbb{R}^{T \times D}$, where $D = 5$ statistics are recorded per step (training loss, gradient norm, gradient variance, parameter norm, effective step size).

Encoder $\mathcal{E}$. We use the iTransformer [Liu et al., 2024b], which applies attention across variates rather than time steps. Each of the D time series $\mathbf{x}_d \in \mathbb{R}^T$ is embedded into a token; cross-variate attention then captures interactions between landscape geometry signals (such as gradient variance) and convergence signals (such as loss decrement). A layer-normalized average pool and a linear projection produce $\hat{u} \in \mathbb{R}^{d_u}$. We selected iTransformer over PatchTST [Nie et al., 2023] in early experiments and over Set Transformer and LSTM in the ablation of Appendix C.

Predictor f. An MLP with one hidden layer of width 128 and ReLU activation produces $\hat{y} = f(\hat{u}, \lambda)$ from the concatenation of $\hat{u}$ and an embedding of λ. Scale-type continuous components (learning rate, weight decay) are log-transformed, all continuous components are then scaled to $[0, 1]$, and categorical ones use Gumbel-Softmax [Jang et al., 2017] so the predictor is differentiable in λ.

Generator G_X (auxiliary). A second branch reconstructs the trajectory from $(\hat{u}, \lambda)$. This branch is used only during meta-training and acts as an information-sufficiency regularizer: it discourages $\hat{u}$ from collapsing to a feature that is predictive on the training distribution but discards information about $\mathbf{X}$ itself.

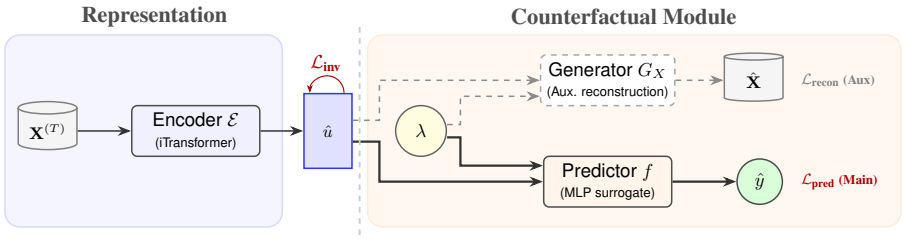

Figure 2: CA-HPO architecture. The encoder maps a short training trajectory $\mathbf{X}^{(T)}$ to a representation $\hat{u}$ that is regularized to be invariant across hyperparameter configurations for the same task ($\mathcal{L}_{\text{inv}}$). The predictor estimates performance for any candidate λ given $\hat{u}$; the generator is an auxiliary information-sufficiency regularizer used only during meta-training.

4.2 Training Objective

Meta-training minimizes

$$\mathcal{L} = \underbrace{\mathbb{E}[(y - f(\hat{u}, \lambda))^2]}_{\mathcal{L}_{\text{pred}}} + \beta_1 \underbrace{\mathbb{E}_{t,i,j}\left[\|\mathcal{E}(\mathbf{X}_{ti}) - \mathcal{E}(\mathbf{X}_{tj})\|_2^2\right]}_{\mathcal{L}_{\text{inv}}} + \beta_2 \underbrace{\mathbb{E}\|\mathbf{X} - G_X(\hat{u}, \lambda)\|_F^2}_{\mathcal{L}_{\text{recon}}}, \tag{6}$$

with $\beta_1 = 1.0$ and $\beta_2 = 0.5$ chosen on a held-out meta-validation split. The invariance term $\mathcal{L}_{\text{inv}}$ is the empirical counterpart of $2P(\mathcal{E})$ in (1), computed over pairs of runs $(\mathbf{X}_{ti}, \mathbf{X}_{tj})$ that share task t but differ in λ, so $\beta = 2\beta_1$ in (2).

4.3 Inference Protocol

For a target task, the encoder is kept frozen. We start from $k_0 = 9$ probe configurations drawn by Latin Hypercube Sampling, run each for only $T \ll T_{\text{total}}$ steps, and aggregate the representation as

$$\hat{u}_t \;=\; \frac{1}{k}\sum_{i=1}^{k} \mathcal{E}(\mathbf{X}_i^{(T)}), \tag{7}$$

and check stability via $\sigma_u = \sqrt{\sum_i \|\mathcal{E}(\mathbf{X}_i^{(T)}) - \hat{u}_t\|_2^2/k}$, adding three probes at a time while $\sigma_u > \tau_{\text{var}}$. Here σ_u is the plug-in estimate of s_t in Theorem 3.4, and (5) explains why more probes are needed when it is large. Because σ_u estimates the within-task spread rather than the standard error of $\hat{u}_t$, this rule need not stop on its own, so the number of probes should be capped ($k_{\max}$ in Algorithm 1). On the four LLM tasks ($d_u = 16$) the check never triggered, so all runs used $k = 9$. In our experiments task-discriminative signal already shows in the first hundred steps (Appendix C), and the probing cost is $\mathcal{O}(kT)$ instead of $\mathcal{O}(NT_{\text{total}})$ for N candidates. Counterfactual selection then maximizes $f(\hat{u}_t, \lambda)$ over Λ via gradient ascent (with Gumbel-Softmax for categorical components). A reliability check then acts as a local trust region on the target task: scale-type hyperparameters such as learning rate and weight decay are log-transformed, continuous coordinates are scaled to $[0, 1]$, categorical ones are one-hot encoded, and Bayesian optimization is used instead of the counterfactual candidate when its minimum distance to the configurations already evaluated on the target task exceeds a fraction τ_{dist} of the diameter of the normalized space. The first counterfactual candidate is always accepted, so the check limits later jumps but does not protect the first recommendation, and it does not test whether a configuration lies inside the meta-training support. After each full evaluation the predictor f is refined on the target observations with the encoder frozen. Algorithm 1 in Appendix B provides the full pseudocode. With $k = 9$, $T = 100$, and $T_{\text{total}} = 10^4$, micro-sensing uses $kT/T_{\text{total}} \approx 0.09$ of the training steps of one full run (measured wall-clock overheads are in Appendix H); the counterfactual queries themselves are essentially free.

5 Experiments

We evaluate CA-HPO on two complementary tracks. The foundation model track tests practical speedup on a setting where evaluations are expensive. The classical benchmark track validates the framework's properties on a more controlled set of tasks. Results are averaged over five seeds; error bars show one standard deviation where plotted.

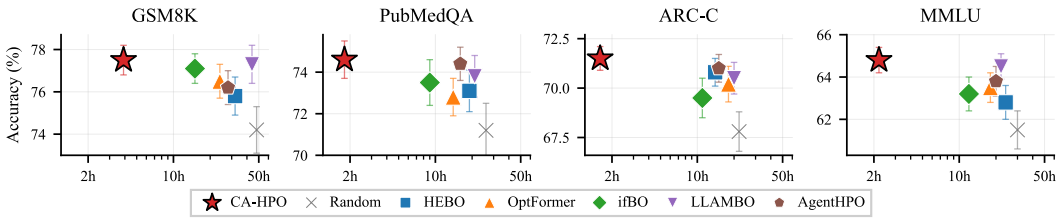

Figure 3: Efficiency-performance trade-off on Llama-3-8B LoRA fine-tuning. Each marker is a method's final accuracy against its search cost; the horizontal axis is search time in hours. On all four tasks CA-HPO (red star) attains the highest mean accuracy at the lowest cost. Error bars show ± 1 std over 5 seeds.

Table 1: Aggregated results on 12 classical benchmarks. Speedup is the geometric mean over tasks of the time-to-target ratio relative to Random Search (target defined in Table 3). Best/Tied/Other (B/T/O) counts the tasks on which a method alone attains the best final performance among all nine methods, shares it, or does not (Table 3). HEBO+MF augments HEBO's GP with the same early-dynamics meta-features that CA-HPO uses; Quick-Tune is run without meta-initialization.

	Random	HEBO	HEBO+MF	OptFormer	ifBO	LLAMBO	AgentHPO	Quick-Tune	**CA-HPO**
Speedup	1.0×	2.0×	3.5×	3.1×	2.8×	2.2×	2.6×	4.3×	**5.7**×
B/T/O	0/0/12	0/0/12	0/0/12	0/0/12	0/1/11	0/0/12	1/0/11	0/0/12	**10/1/1**

5.1 Setup

Foundation Model Track. We fine-tune Llama-3-8B [Grattafiori et al., 2024] with LoRA [Hu et al., 2022, Zhou et al., 2026] on four downstream tasks spanning different reasoning modes: GSM8K [Cobbe et al., 2021] (7,473 train), PubMedQA [Jin et al., 2019] (1,000 expert), ARC-Challenge [Clark et al., 2018] (1,119), and a curated 2,000-sample MMLU [Hendrycks et al., 2021] subset. The meta-training corpus comprises Alpaca [Taori et al., 2023], Dolly-15k [Conover et al., 2023], and OpenAssistant [Köpf et al., 2023]. The hyperparameter space is six-dimensional: learning rate $\eta \in [10^{-6}, 5 \times 10^{-4}]$, LoRA rank $r \in \{8, 16, 32, 64, 128\}$, LoRA alpha $\alpha \in \{16, 32, 64\}$, dropout $p \in [0, 0.3]$, weight decay $\gamma \in [10^{-6}, 10^{-2}]$, and scheduler type.

Classical Benchmarks. Twelve tasks across vision (MNIST [LeCun et al., 1998], F-MNIST [Xiao et al., 2017], CIFAR-10 and CIFAR-100 [Krizhevsky and Hinton, 2009], SVHN [Netzer et al., 2011], with ResNet-18 [He et al., 2016]), text (AG News [Zhang et al., 2015], SST-2 [Socher et al., 2013], IMDB [Maas et al., 2011], TREC [Li and Roth, 2002], with BERT-base [Devlin et al., 2019]), and tabular regression (Boston Housing [Harrison and Rubinfeld, 1978], California Housing [Pace and Barry, 1997], Diabetes [Efron et al., 2004], with an MLP). Vision tasks tune learning rate, weight decay, batch size, and augmentation intensity; NLP tasks additionally tune warmup ratio and gradient clipping.

Baselines. We compare against twelve methods covering the major paradigms. Random Search [Bergstra and Bengio, 2012] as a sanity check; BOHB [Falkner et al., 2018], ASHA [Li et al., 2020], PriorBand [Mallik et al., 2023], and LAMDA [Li et al., 2026a] as multi-fidelity methods that also exploit low-fidelity training signal; HEBO [Cowen-Rivers et al., 2022] as a state-of-the-art Bayesian optimizer; HEBO+Meta-Features, an instance of multi-task BO [Swersky et al., 2013, Perrone et al., 2018] where the GP is augmented with the same early-dynamics features that CA-HPO uses; OptFormer [Chen et al., 2022] and ifBO [Rakotoarison et al., 2024] as transformer-based meta-learners; LLAMBO [Liu et al., 2024a] and AgentHPO [Liu et al., 2025] as LLM-driven schemes; and Quick-Tune [Pineda Arango et al., 2024] as a meta-feature-aware HPO baseline. Quick-Tune was originally meta-learned on a different ecosystem (ImageNet pretrained models on Meta-Album), so we run its algorithmic framework without its meta-learned initialization, corresponding to the QT:−M,+C,+G variant in the original paper; we make no claim about Quick-Tune in its native setting with its transferred prior. All methods get equivalent wall-clock budgets. Multi-fidelity

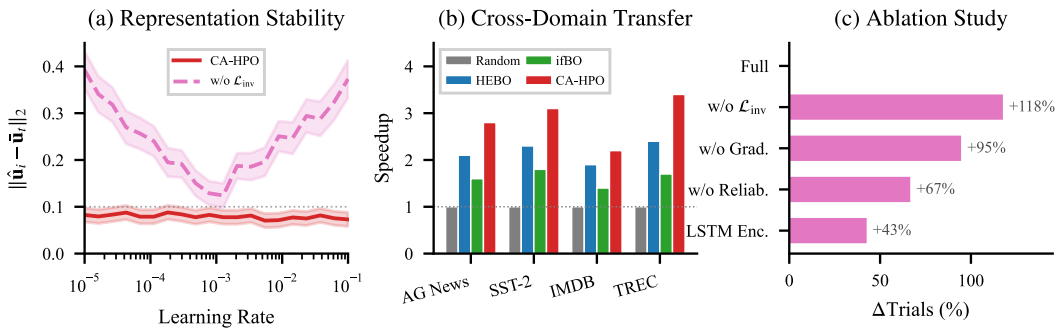

Figure 4: (a) Representations stay close to the per-task centroid across four orders of magnitude of learning rate; ablating $\mathcal{L}_{\text{inv}}$ causes substantial drift. (b) Cross-domain transfer from 5 vision tasks to 4 NLP tasks at 2.9× average speedup. (c) Component ablation in terms of additional trials over the full model.

methods use fidelities from 1/16 of a run to one full run with four parallel workers, and CA-HPO is charged for all probes, fallbacks, and full evaluations.

5.2 Main Results

Foundation model efficiency. Figure 3 shows the efficiency-accuracy trade-off on Llama-3-8B fine-tuning. On all four tasks CA-HPO attains the highest mean accuracy at the lowest search cost: on GSM8K it reaches 77.5% after 3.9 hours of search, while Random Search reaches 74.2% after 48 hours, and the other three tasks show the same pattern. Because these are final operating points rather than time-to-target curves, we do not convert them into a single speedup factor. On the four evaluated tasks, micro-sensing takes about six minutes on a single A40, after which counterfactual selection proposes a candidate without further training. Full evaluation is then needed for only a few configurations (5 on GSM8K, against about 48 for HEBO at the same per-run cost; Appendix H).

Classical benchmark summary. Table 1 aggregates over the 12 classical tasks. CA-HPO is best on 10 of 12 tasks, ties with ifBO on TREC, and has the highest geometric-mean speedup (5.7×). It also stays ahead of HEBO+Meta-Features (3.5×) and Quick-Tune (4.3×). Multi-fidelity methods reach 1.5 to 2.1× (Table 2). The single loss is on IMDB, where AgentHPO reaches 92.8% versus CA-HPO's 92.5%.

Accounting for offline cost. The speedups above are warm-start results. Building the meta-training corpus costs 480 GPU-hours and training the encoder and predictor about 8 GPU-hours, both one-off. On GSM8K the online search costs about 3.95 GPU-hours for CA-HPO and 36 for HEBO on one A40, so the offline cost of 488 GPU-hours breaks even after $488/(36 - 3.95) \approx 16$ comparable tasks, assuming the corpus is reused and future tasks cost about the same. CA-HPO is thus unfavorable for a one-off cold start.

5.3 Analysis

Testing independence from λ. A task-determined encoder (Definition 2.1) produces per-probe encodings that are independent of λ within a task. We probe this necessary condition with task-wise permutation tests based on the Hilbert–Schmidt Independence Criterion (HSIC) [Gretton et al., 2007]: on each of the 12 classical benchmarks we draw 200 per-probe encodings $\hat{u}_i = \mathcal{E}(\mathbf{X}_i)$ paired with their configurations λ_i, compute HSIC, and form the null distribution by permuting λ (1,000 permutations). At the unadjusted 0.05 level, independence is not rejected on 11 of 12 tasks (p from 0.11 to 0.52; Table 4 in Appendix C), the exception being IMDB ($p = 0.04$). Non-rejection does not establish independence. That IMDB is also the only task where CA-HPO loses to a baseline is an observed association, not an established mechanism: with 12 tests the single $p = 0.04$ is not conclusive, and none of the per-dimension p-values on IMDB (0.31 for learning rate, 0.08 for batch size, 0.03 for warmup ratio, 0.05 for gradient clipping) remains significant after Holm correction. The representation-stability curve in Figure 4(a) tells the same story from a different angle: for CA-HPO

Table 2: Isolating the invariance constraint on the 12 classical benchmarks (geometric-mean speedup over Random Search, time-to-target as in Table 3, 5 seeds). The middle block has access to the same short-trajectory signal as CA-HPO but no invariant representation.

Group	Method	Speedup
Multi-fidelity, no meta-knowledge	BOHB / ASHA / PriorBand / LAMDA	1.5/1.6/1.7/2.1×
Same probes, no invariance	HEBO + raw early-dynamics features	3.5×
	CA-HPO, $\beta_1 = 0$ (same probes, capacity)	2.6×
Full method	**CA-HPO**	**5.7**×

the mean deviation of the per-probe representation from the task centroid stays below about 0.1 across four orders of magnitude of learning rate, while removing $\mathcal{L}_{\text{inv}}$ pushes it past 0.35 at the extremes.

Is the gain from probing or from invariance? Short probes are a low-fidelity signal, so part of the gain could come from probing alone; Table 2 separates the two. Multi-fidelity methods that use early training signal without meta-knowledge reach 1.5 to 2.1×. Giving HEBO the same raw early-dynamics features reaches 3.5×, and CA-HPO itself with the invariance term switched off, with identical probes and model capacity, reaches only 2.6×. Adding the within-task invariance constraint raises this to 5.7×. The probing signal is therefore useful but not sufficient, and the largest single step in the table comes from the constraint on the representation; Figure 4(c) gives the same picture in trial counts.

Controlled test against a known invariant. On synthetic quadratic tasks where a known task variable sets the Hessian spectrum and gradient noise while λ sets step size and momentum, a held-out linear probe recovers that variable from $\hat{u}$ with $R^2 = 0.93$. A λ-dependent task-configuration interaction, which raises the price of invariance within the implemented model classes, lowers this to 0.41 and raises counterfactual error 2.8 times (Appendix D).

Comparison with alternative invariance designs. Definition 2.1 imposes invariance along one particular axis. We compare three variants on the classical benchmarks: (V1) cross-task invariance, in the spirit of domain generalization, $\mathrm{Var}_t[\mathbb{E}[u \mid t]] \to 0$; (V2) IRM-style [Arjovsky et al., 2019] predictor invariance across λ buckets; (V3) our design, $u \perp \lambda \mid t$. V1 removes exactly the task information that transfer needs and yields 1.7×; V2 constrains only the predictor, leaves task and configuration entangled in the representation, and yields 3.0×; V3 yields 5.7×.

Cross-domain transfer and OOD behavior. Figure 4(b) meta-trains on the 5 vision tasks and tests on the 4 NLP tasks without any NLP meta-training. CA-HPO reaches 2.9× average speedup against Random Search, against 5.4× when meta-trained in domain, and stays above HEBO (2.2×) and the meta-learning baselines, which degrade to 1.4 to 1.6× under domain shift. In an extreme-shift stress test (CIFAR-10 to ImageNet, which changes task, resolution, and model capacity together) the speedup drops to 1.3×, close to Bayesian optimization (Appendix G). The reliability threshold $\tau_{\text{dist}} = 0.3$ was selected on meta-validation and not retuned for this experiment; values of 0.1 and 0.5 give 2.4× and 2.7× (Appendix C.6).

Component ablation. Figure 4(c) decomposes the contribution of each component. Removing $\mathcal{L}_{\text{inv}}$ requires 118% more trials, the largest single drop and consistent with the role of the invariance term in Theorem 3.1. Removing gradient features increases trials by 95%, removing the reliability check by 67%, and replacing iTransformer with LSTM by 43%.

6 Related Work

Bayesian and multi-fidelity HPO. Classical HPO uses random search [Bergstra and Bengio, 2012], GP-based BO [Snoek et al., 2012, Srinivas et al., 2010, Shahriari et al., 2016], and multi-fidelity strategies [Jamieson and Talwalkar, 2016, Falkner et al., 2018, Klein et al., 2017, Li et al., 2020]. PriorBand [Mallik et al., 2023] combines expert priors with multi-fidelity scheduling, and LAMDA [Li et al., 2026a] learns a prior over promising regions from low-fidelity data before a

second search phase. HEBO [Cowen-Rivers et al., 2022] sets a strong robust baseline. Multi-task and transfer BO [Swersky et al., 2013, Perrone et al., 2018, Salinas et al., 2020] share information across tasks via task kernels. CA-HPO belongs to the family of low-fidelity meta-surrogates: its probes are short-horizon learning curves and its predictor is a meta-learned surrogate. It differs in that transfer is mediated by a task representation constrained to be configuration-invariant within each task, and Table 2 shows that this constraint, rather than the low-fidelity signal alone, accounts for most of the gain.

Neural meta-HPO. OptFormer [Chen et al., 2022] and ifBO [Rakotoarison et al., 2024, Adriaensen et al., 2023, Ding et al., 2025, Li and Zhao, 2026b] learn surrogates from optimization histories or learning curves. LLM-driven methods [Liu et al., 2024a, 2025, Yang et al., 2024, Xi et al., 2025] reason about hyperparameters from textual context. Most relevant to us are meta-feature-aware methods such as Zero-Shot AutoML [Öztürk et al., 2022], which uses dataset descriptors, and Quick-Tune [Pineda Arango et al., 2024], whose surrogate also conditions on learning curves. CA-HPO differs in that its task descriptor is a learned summary of short training trajectories that is constrained to be invariant across configurations of the same task, rather than an unconstrained curve or metadata embedding.

Invariance and representation learning. Invariant Risk Minimization [Arjovsky et al., 2019] and related distributional invariance methods [Zhou et al., 2025] enforce predictor invariance across environments; domain generalization methods enforce feature invariance across domains. Our invariance axis is different: invariance of the representation across λ for fixed t, with λ playing the role of an intervention. In the variant comparison of Section 5.3, this within-task constraint performs best among the three designs evaluated.

Causal representation learning. Recent CRL work [Khemakhem et al., 2020, Lippe et al., 2022, Brehmer et al., 2022, Zhang et al., 2023, Zheng et al., 2022] establishes identifiability of latent factors under various structural assumptions. Our framework is deliberately weaker: we do not identify any latent factor, but define the target through a constrained optimization problem and analyze the penalized estimator with standard tools; the causal diagram serves only as motivation.

7 Conclusion

We presented CA-HPO, a transfer HPO framework built on a representation that is required to be invariant across hyperparameter configurations of the same task and trained to predict performance. Defining it through a constrained optimization problem lets the analysis rest on standard statistical learning tools rather than identifiability assumptions. We proved a finite-penalty invariance bound with recovery in the limit, a task-level excess-risk bound, and a counterfactual error bound that ties the probe budget to the invariance penalty. Empirically, CA-HPO attains the highest accuracy at the lowest search cost on Llama-3-8B fine-tuning and a 5.7 times warm-start speedup over random search on twelve classical benchmarks, ahead of multi-fidelity and meta-feature-aware baselines, with matched ablations attributing most of the gain to the invariance constraint. Future work includes extending the framework to multi-objective HPO and to search spaces that include architecture or optimizer family.

Limitations. CA-HPO rests on four operational premises that bound its applicability. (i) Intrinsic task properties must manifest within the first T steps of training; this fails for tasks with very long warmup phases and for non-stationary tasks where the relevant invariants drift over the curriculum, in which case periodic re-sensing is needed and erodes the speedup (Appendix G). (ii) Meta-training requires a sufficiently diverse task pool and a one-off offline cost of about 488 GPU-hours, which is amortized only over roughly 16 or more comparable tasks; under heavy domain shift the speedup degrades toward that of Bayesian optimization ($1.3\times$ on the CIFAR-to-ImageNet stress test), and the reliability check does not protect the first recommendation or certify unseen regions of Λ. (iii) The encoder and predictor have about 163K parameters; width is not the bottleneck on CIFAR-10, but we have not shown that this capacity suffices for LLM fine-tuning dynamics in general or for unfamiliar hyperparameter regions. (iv) When the search space has only a few candidates, evaluating all of them needs only a few more full runs than CA-HPO and is preferable.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grants 62606363, 42550187, and 62476191. The authors declare no competing interests.

References

Steven Adriaensen, Herilalaina Rakotoarison, Samuel Müller, and Frank Hutter. Efficient Bayesian learning curve extrapolation using prior-data fitted networks. In *NeurIPS*, 2023.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

Johann Brehmer, Pim De Haan, Phillip Lippe, and Taco Cohen. Weakly supervised causal representation learning. In *NeurIPS*, volume 35, pages 38319–38331, 2022.

Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc' Aurelio Ranzato, Sagi Perel, and Nando de Freitas. Towards learning universal hyperparameter optimizers with transformers. In *NeurIPS*, volume 35, pages 32053–32068, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

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

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free Dolly: Introducing the world's first truly open instruction-tuned LLM. Databricks blog, 2023.

Alexander I Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, et al. HEBO: Pushing the limits of sample-efficient hyper-parameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

Yanna Ding, Zijie Huang, Xiao Shou, Yihang Guo, Yizhou Sun, and Jianxi Gao. Architecture-aware learning curve extrapolation via graph ordinary differential equation. In *AAAI*, pages 16289–16297, 2025.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *ICML*, pages 1437–1446. PMLR, 2018.

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.

Arthur Gretton, Kenji Fukumizu, Choon Hui Teo, Le Song, Bernhard Schölkopf, and Alex J Smola. A kernel statistical test of independence. In *NeurIPS*, volume 20, pages 585–592, 2007.

David Harrison and Daniel L Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

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

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *AISTATS*, pages 240–248, 2016.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. In *ICLR*, 2017.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. In *EMNLP-IJCNLP*, pages 2567–2577, 2019.

Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvärinen. Variational autoencoders and nonlinear ICA: A unifying framework. In *AISTATS*, pages 2207–2217. PMLR, 2020.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *AISTATS*, pages 528–536. PMLR, 2017.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. OpenAssistant conversations: Democratizing large language model alignment. In *NeurIPS Datasets and Benchmarks Track*, pages 47669–47681, 2023.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Fan Li, Shengbo Wang, and Ke Li. LAMDA: Two-phase HPO via learning prior from low-fidelity data. In *AAAI*, volume 40, pages 37018–37026, 2026a.

Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. In *MLSys*, volume 2, pages 230–246, 2020.

Mengyang Li and Pinlong Zhao. DABO: Difficulty-aware Bayesian optimization with diffusion-learned priors. In *CVPR*, pages 6126–6135, 2026a.

Mengyang Li and Pinlong Zhao. Difficulty-aware learning curve extrapolation. In *AAAI*, pages 23021–23029, 2026b.

Mengyang Li, Xiaoling Zhou, and Ou Wu. Delving into the training dynamics for image classification. *IEEE Transactions on Image Processing*, 34:6783–6798, 2025.

Mengyang Li, Jingwen Wang, and Pinlong Zhao. What do LLMs learn first? Asymmetric learning dynamics of input complexity and output ambiguity in preference alignment. In *ACL*, pages 17373–17388, 2026b.

Xin Li and Dan Roth. Learning question classifiers. In *COLING*, pages 1–7, 2002.

Phillip Lippe, Sara Magliacane, Sindy Löwe, Yuki M. Asano, Taco Cohen, and Stratis Gavves. CITRIS: Causal identifiability from temporal intervened sequences. In *ICML*, pages 13557–13603. PMLR, 2022.

Siyi Liu, Chen Gao, and Yong Li. AgentHPO: Large language model agent for hyper-parameter optimization. In *CPAL*, pages 1146–1169, 2025.

Tennison Liu, Nicolás Astorga, Nabeel Seedat, and Mihaela van der Schaar. Large language models to enhance Bayesian optimization. In *ICLR*, 2024a.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *ICLR*, 2024b.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150, 2011.

Neeratyoy Mallik, Edward Bergman, Carl Hvarfner, Danny Stoll, Maciej Janowski, Marius Lindauer, Luigi Nardi, and Frank Hutter. PriorBand: Practical hyperparameter optimization in the age of deep learning. In *NeurIPS*, volume 36, pages 7377–7391, 2023.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 978-0-262-01825-8.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *ICLR*, 2023.

Ekrem Öztürk, Fabio Ferreira, Hadi S. Jomaa, Lars Schmidt-Thieme, Josif Grabocka, and Frank Hutter. Zero-shot AutoML with pretrained models. In *ICML*, pages 17138–17155, 2022.

R Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297, 1997.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *NeurIPS*, volume 31, pages 6845–6855, 2018.

Sebastian Pineda Arango, Fabio Ferreira, Arlind Kadra, Frank Hutter, and Josif Grabocka. Quick-tune: Quickly learning which pretrained model to finetune and how. In *ICLR*, pages 13013–13032, 2024.

Herilalaina Rakotoarison, Steven Adriaensen, Neeratyoy Mallik, Samir Garibov, Eddie Bergman, and Frank Hutter. In-context freeze-thaw Bayesian optimization for hyperparameter optimization. In *ICML*, volume 235, pages 41982–42008. PMLR, 2024.

David Salinas, Huibin Shen, and Valerio Perrone. A quantile-based approach for hyperparameter transfer learning. In *ICML*, pages 8438–8448, 2020.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

Steve Smale and Ding-Xuan Zhou. Learning theory estimates via integral operators and their approximations. *Constructive Approximation*, 26(2):153–172, 2007.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *NeurIPS*, volume 25, pages 2951–2959, 2012.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, 2013.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, pages 1015–1022, 2010.

Kevin Swersky, Jasper Snoek, and Ryan P. Adams. Multi-task Bayesian optimization. In *NeurIPS*, pages 2004–2012, 2013.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *ICLR*, 2024.

Jiaqi Zhang, Kristjan Greenewald, Chandler Squires, Akash Srivastava, Karthikeyan Shanmugam, and Caroline Uhler. Identifiability guarantees for causal disentanglement from soft interventions. In *NeurIPS*, volume 36, pages 50254–50292, 2023.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *NeurIPS*, volume 28, pages 649–657, 2015.

Yujia Zheng, Ignavier Ng, and Kun Zhang. On the identifiability of nonlinear ICA: Sparsity and beyond. In *NeurIPS*, volume 35, pages 16411–16422, 2022.

Xiaoling Zhou, Mingjie Zhang, Zhemg Lee, Yuncheng Hua, Chengli Xing, Wei Ye, Flora Salim, and Shikun Zhang. Boosting resilience of large language models through causality-driven robust optimization. In *NeurIPS*, volume 38, pages 55821–55845, 2025.

Xiaoling Zhou, Mingjie Zhang, Zhemg Lee, Wei Ye, and Shikun Zhang. LeLoRA: Learnable low-rank adaptation of large language models. In *ACL*, pages 35255–35273, 2026.

A Proofs

A.1 Properties of the Penalty

Lemma A.1. *For every encoder $\mathcal{E}$ with $\mathbb{E}\|\mathcal{E}(\mathbf{X})\|_2^2 < \infty$: (i) $P(\mathcal{E}) = \mathbb{E}_t[s_t^2(\mathcal{E})]$; (ii) $V(\mathcal{E}) \le P(\mathcal{E})$; (iii) $P(\mathcal{E}) = 0$ if and only if $\mathcal{E}$ is task-determined.*

Proof. Fix t and let $a = \mathcal{E}(\mathbf{X})$, $a' = \mathcal{E}(\mathbf{X}')$ be the outputs of two independent runs of t, with mean $\bar{a} = \mathbb{E}[a \mid t]$. Expanding the square and using independence, $\frac{1}{2}\mathbb{E}\|a - a'\|_2^2 = \mathbb{E}\|a - \bar{a}\|_2^2 - \langle \mathbb{E}[a - \bar{a}], \mathbb{E}[a' - \bar{a}] \rangle = s_t^2(\mathcal{E})$, and taking $\mathbb{E}_t$ gives (i). For (ii), the law of total variance over λ gives $s_t^2(\mathcal{E}) = \mathbb{E}_\lambda \|\mathbb{E}[a \mid t, \lambda] - \bar{a}\|_2^2 + \mathbb{E}_\lambda \mathbb{E}[\|a - \mathbb{E}[a \mid t, \lambda]\|_2^2 \mid t, \lambda]$; the first term is $\mathrm{Var}_\lambda(\mathbb{E}[a \mid t, \lambda])$ and the second is non-negative, so taking $\mathbb{E}_t$ gives $V(\mathcal{E}) \le P(\mathcal{E})$. For (iii), $P(\mathcal{E}) = 0$ if and only if $s_t^2(\mathcal{E}) = 0$ for $p(\mathcal{T})$-almost every t, that is, $\mathcal{E}(\mathbf{X}) = \mathbb{E}[\mathcal{E}(\mathbf{X}) \mid t]$ almost surely, which is the definition of task-determined with $u_{\mathcal{E}}(t) = \mathbb{E}[\mathcal{E}(\mathbf{X}) \mid t]$. □

A.2 Proof of Theorem 3.1

Proof. (a) Let $(\mathcal{E}, f)$ be any pair with $\mathcal{E}$ task-determined. By Lemma A.1(iii), $P(\mathcal{E}) = 0$, so optimality of $(\mathcal{E}_\beta, f_\beta)$ gives

$$R(\mathcal{E}_\beta, f_\beta) + \beta P(\mathcal{E}_\beta) \;\le\; \mathcal{L}_\beta(\mathcal{E}, f) \;=\; R(\mathcal{E}, f). \tag{8}$$

Taking the infimum over such pairs yields $R(\mathcal{E}_\beta, f_\beta) + \beta P(\mathcal{E}_\beta) \le R^*_{\mathrm{inv}}$. Since $P \ge 0$, $R(\mathcal{E}_\beta, f_\beta) \le R^*_{\mathrm{inv}}$. Since $R(\mathcal{E}_\beta, f_\beta) \ge R_0$, $\beta P(\mathcal{E}_\beta) \le R^*_{\mathrm{inv}} - R_0$. Lemma A.1(ii) gives $V(\mathcal{E}_\beta) \le P(\mathcal{E}_\beta)$.

(b) Let $\beta_n \to \infty$ and let $(\mathcal{E}_{\beta_n}, f_{\beta_n})$ converge to $(\mathcal{E}_\infty, f_\infty)$ in the given topology; compactness guarantees that such limit points exist. Lower semicontinuity of P and part (a) give $P(\mathcal{E}_\infty) \le \liminf_n P(\mathcal{E}_{\beta_n}) \le \liminf_n (R^*_{\mathrm{inv}} - R_0)/\beta_n = 0$, so $\mathcal{E}_\infty$ is task-determined by Lemma A.1(iii). Lower semicontinuity of R and part (a) give $R(\mathcal{E}_\infty, f_\infty) \le \liminf_n R(\mathcal{E}_{\beta_n}, f_{\beta_n}) \le R^*_{\mathrm{inv}}$. Because $\mathcal{E}_\infty$ is task-determined, also $R(\mathcal{E}_\infty, f_\infty) \ge R^*_{\mathrm{inv}}$. Hence $R(\mathcal{E}_\infty, f_\infty) = R^*_{\mathrm{inv}}$, the infimum in Definition 2.1 is attained, and $(\mathcal{E}_\infty, f_\infty)$ is an invariant predictive pair. □

A.3 Proof of Theorem 3.3

Proof. The block loss is

$$\ell_\beta(\mathcal{E}, f; Z_n) = \frac{1}{m} \sum_j \big(f(\mathcal{E}(\mathbf{X}_{nj}), \lambda_{nj}) - y_{nj}\big)^2 + \frac{\beta}{m(m-1)} \sum_{j \ne j'} \tfrac{1}{2}\|\mathcal{E}(\mathbf{X}_{nj}) - \mathcal{E}(\mathbf{X}_{nj'})\|_2^2. \tag{9}$$

Unbiasedness: given t_n, the runs in Z_n are independent, so the first sum in ℓ_β has expectation $R(\mathcal{E}, f)$ and each pair $j \ne j'$ in the second sum has expectation $\frac{1}{2}\mathbb{E}[\|\mathcal{E}(\mathbf{X}) - \mathcal{E}(\mathbf{X}')\|_2^2 \mid t_n]$; averaging over t_n gives $\mathbb{E}[\ell_\beta(\mathcal{E}, f; Z_n)] = \mathcal{L}_\beta(\mathcal{E}, f)$.

Range: under Assumption 3.2, $(f - y)^2 \le 4B^2$ and $\frac{1}{2}\|\mathcal{E}(\mathbf{X}) - \mathcal{E}(\mathbf{X}')\|_2^2 \le 2B_u^2$, so $\ell_\beta \in [0, M]$ with $M = 4B^2 + 2\beta B_u^2$.

Uniform deviation: the blocks $Z_1, \dots, Z_N$ are independent and identically distributed. Applying the Rademacher bound of Mohri et al. [2012, Chapter 3] to the class $\{\ell_\beta / M\}$ and rescaling, with probability at least $1 - \delta/2$, for all $(\mathcal{E}, f) \in \mathfrak{E} \times \mathfrak{F}$,

$$\mathcal{L}_\beta(\mathcal{E}, f) \;\le\; \hat{\mathcal{L}}_\beta(\mathcal{E}, f) + 2\mathfrak{R}_N + M\sqrt{\frac{\log(2/\delta)}{2N}}. \tag{10}$$

For the fixed pair $(\mathcal{E}_\beta, f_\beta)$, Hoeffding's inequality gives, with probability at least $1 - \delta/2$, $\hat{\mathcal{L}}_\beta(\mathcal{E}_\beta, f_\beta) \le \mathcal{L}_\beta(\mathcal{E}_\beta, f_\beta) + M\sqrt{\log(2/\delta)/(2N)}$. On the intersection of the two events, which has probability at least $1 - \delta$, and using $\hat{\mathcal{L}}_\beta(\hat{\mathcal{E}}, \hat{f}) \le \hat{\mathcal{L}}_\beta(\mathcal{E}_\beta, f_\beta)$,

$$\mathcal{L}_\beta(\hat{\mathcal{E}}, \hat{f}) \;\le\; \mathcal{L}_\beta(\mathcal{E}_\beta, f_\beta) + 2\mathfrak{R}_N + 2M\sqrt{\frac{\log(2/\delta)}{2N}} \;=\; \mathcal{L}_\beta(\mathcal{E}_\beta, f_\beta) + \varepsilon_N. \tag{11}$$

Consequences: by the first display in the proof of Theorem 3.1, $\mathcal{L}_\beta(\mathcal{E}_\beta, f_\beta) \le R^*_{\mathrm{inv}}$, so $R(\hat{\mathcal{E}}, \hat{f}) + \beta P(\hat{\mathcal{E}}) \le R^*_{\mathrm{inv}} + \varepsilon_N$. Dropping $\beta P \ge 0$ gives the risk bound, and using $R(\hat{\mathcal{E}}, \hat{f}) \ge R_0$ gives $P(\hat{\mathcal{E}}) \le (R^*_{\mathrm{inv}} - R_0 + \varepsilon_N)/\beta$. □

A.4 Proof of Theorem 3.4

Proof. The pair $(\hat{\mathcal{E}}, \hat{f})$ is fixed and independent of the probes. Let $\xi_i = \hat{\mathcal{E}}(\mathbf{X}_i) - \bar{u}_t$ for the k probe runs. Conditional on t, the ξ_i are independent and identically distributed, $\mathbb{E}\xi_i = 0$, $\|\xi_i\|_2 \le 2B_u$, and $\mathbb{E}\|\xi_i\|_2^2 = s_t^2$. The Bernstein inequality for bounded random vectors in a Hilbert space [Smale and Zhou, 2007, Lemma 2] gives, with probability at least $1 - \delta$,

$$\|\hat{u}_t - \bar{u}_t\|_2 \;=\; \Big\|\frac{1}{k}\sum_{i=1}^{k}\xi_i\Big\|_2 \;\le\; \frac{4B_u \log(2/\delta)}{k} + s_t\sqrt{\frac{2\log(2/\delta)}{k}}. \tag{12}$$

On this event, for every $\lambda \in \Lambda$,

$$|\hat{f}(\hat{u}_t, \lambda) - y(t, \lambda)| \;\le\; |\hat{f}(\hat{u}_t, \lambda) - \hat{f}(\bar{u}_t, \lambda)| + |\hat{f}(\bar{u}_t, \lambda) - \mu(t, \lambda)| + |\mu(t, \lambda) - y(t, \lambda)|. \tag{13}$$

The first term is at most $L_f\|\hat{u}_t - \bar{u}_t\|_2$ by the Lipschitz condition, which holds uniformly in λ; the second is at most ϵ_t by definition. This gives (4). For (5), each of the two terms in the vector bound is at most $\varepsilon/2$ when $k \ge 8s_t^2 \log(2/\delta)/\varepsilon^2$ and $k \ge 8B_u \log(2/\delta)/\varepsilon$, respectively. □

A.5 Lower Bound on Representation Variability (No-Collapse)

We give the no-collapse result that was referenced in Section 3. The argument formalizes the intuition that distinguishable tasks must lead to distinguishable representations.

Proposition A.2 (Lower bound on representation variability)**.** *Define the task separation* $\mathcal{D} = \inf_{t \ne t'} \sup_{\lambda \in \Lambda} |\mathbb{E}[y(t, \lambda)] - \mathbb{E}[y(t', \lambda)]|$*; let* $\hat{u}_t$ *be a task-level representation (for example* $\bar{u}_t$*), and assume the predictor* $\hat{f}$ *is* L_f*-Lipschitz in its first argument, that its prediction error satisfies* $|\hat{f}(\hat{u}_t, \lambda) - \mathbb{E}[y(t, \lambda)]| \le \epsilon$ *uniformly over* t *and* λ*, and that two independent draws from* $p(\mathcal{T})$ *differ with probability at least* $1/2$*. Then*

$$\mathrm{Var}_t\big(\hat{u}_t\big) \;\ge\; \frac{1}{4L_f^2}\,(\mathcal{D} - 2\epsilon)_+^2. \tag{14}$$

Proof. Pick distinct tasks $t \ne t'$ and, for any $\eta > 0$, a configuration λ^* with $|\mathbb{E}[y(t, \lambda^*)] - \mathbb{E}[y(t', \lambda^*)]| \ge \mathcal{D} - \eta$. By the triangle inequality,

$$\mathcal{D} \le \eta + \big|\mathbb{E}[y(t, \lambda^*)] - \hat{f}(\hat{u}_t, \lambda^*)\big| + \big|\hat{f}(\hat{u}_t, \lambda^*) - \hat{f}(\hat{u}_{t'}, \lambda^*)\big| + \big|\hat{f}(\hat{u}_{t'}, \lambda^*) - \mathbb{E}[y(t', \lambda^*)]\big| \tag{15}$$

$$\le \eta + \epsilon + L_f\|\hat{u}_t - \hat{u}_{t'}\|_2 + \epsilon. \tag{16}$$

Letting $\eta \to 0$ gives $\|\hat{u}_t - \hat{u}_{t'}\|_2 \ge (\mathcal{D} - 2\epsilon)_+/L_f$. Using the identity $\mathrm{Var}_t(\hat{u}_t) = \frac{1}{2}\mathbb{E}_{t,t'}[\|\hat{u}_t - \hat{u}_{t'}\|_2^2]$ for independent t, t', the lower bound on each pairwise distance with $t \ne t'$, and $\Pr(t \ne t') \ge 1/2$,

$$\mathrm{Var}_t(\hat{u}_t) \ge \frac{(\mathcal{D} - 2\epsilon)_+^2}{4L_f^2}. \tag{17}$$

□

B Algorithm Details

To minimize per-probe overhead in the LLM setting, the base 8B model is held in GPU memory and only the LoRA adapters are swapped between probes; with sequence length 256 and batch size 4 during micro-sensing, each probe takes ≈ 40s on a single A40, so the entire phase ($k = 9$) finishes in under 6 minutes. Full evaluations in Phase 2 use the standard fine-tuning configuration of the target task.

C Extended Experimental Results

C.1 Per-Task Results on Classical Benchmarks

Table 3 reports per-task final performance and time-to-target for Random Search, HEBO, and CA-HPO, together with the strongest other baseline on each task; the Best/Tied/Other column of Table 1 is computed from it.

Algorithm 1 CA-HPO Inference

Require: Target task t, hyperparameter space Λ, evaluation budget B
Require: Pre-trained $\{\mathcal{E}, G_X, f\}$
Require: Micro-sensing length $T = 100$, initial probes $k_0 = 9$, probe cap $k_{\max}$, thresholds $\tau_{\text{var}} = 0.1$, $\tau_{\text{dist}} = 0.3$
Require: Normalization ϕ: log-transform scale-type coordinates (learning rate, weight decay), scale continuous coordinates to $[0, 1]$, one-hot encode categorical ones
Ensure: Optimized configuration λ^*

Phase 1: Micro-sensing (encoder frozen)
Sample $\{\lambda_i\}_{i=1}^{k_0} \subset \Lambda$ via Latin Hypercube Sampling; $k \leftarrow k_0$
for $i = 1$ **to** k **do**
  Run T gradient steps with λ_i, record $\mathbf{X}_i = [\mathcal{L}, \|\nabla\|_2, \mathrm{Var}(\nabla), \|\theta\|_2, \eta]_{1:T}$; $\hat{u}_i \leftarrow \mathcal{E}(\mathbf{X}_i)$
end for
$\bar{u}_t \leftarrow k^{-1} \sum_{i=1}^{k} \hat{u}_i$, $\sigma_u \leftarrow \sqrt{k^{-1} \sum_{i=1}^{k} \|\hat{u}_i - \bar{u}_t\|^2}$
while $\sigma_u > \tau_{\text{var}}$ **and** $k + 3 \leq k_{\max}$ **do**
  Run 3 more probes as above, $k \leftarrow k + 3$, recompute $\bar{u}_t$ and σ_u
end while
Phase 2: Counterfactual selection
Initialize observation set $\mathcal{D} \leftarrow \emptyset$
while $|\mathcal{D}| < B$ **do**
  $\lambda_{\text{cf}} \leftarrow \arg\max_\lambda f(\bar{u}_t, \lambda)$ via gradient ascent (Gumbel-Softmax for categorical components)
  if $\mathcal{D} = \emptyset$ **or** $\min_{(\lambda,y)\in\mathcal{D}} \|\phi(\lambda_{\text{cf}}) - \phi(\lambda)\|_2 \leq \tau_{\text{dist}} \cdot \mathrm{diam}(\phi(\Lambda))$ **then**
    $\lambda_{\text{next}} \leftarrow \lambda_{\text{cf}}$ ▷ Use counterfactual prediction
  else
    $\lambda_{\text{next}} \leftarrow \arg\max_\lambda \mathrm{EI}(\lambda; \mathrm{GP}(\mathcal{D}))$ ▷ Bayesian fallback
  end if
  Run full training with λ_{next}, observe y_{next}; $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\lambda_{\text{next}}, y_{\text{next}})\}$
  Refine f on $\mathcal{D}$ for 5 epochs with $\mathcal{E}$ frozen
end while
return $\arg\max_{(\lambda,y)\in\mathcal{D}} y$

Two patterns are worth noting in the per-task numbers. First, the speedup over Random Search, computed from the Time columns, ranges from 4.5× on IMDB to 7.5× on Boston Housing, with the three tabular regression tasks at 6.0 to 7.5× and the vision tasks at 5.0 to 6.8×. Second, the only loss is on IMDB (92.5% versus 92.8% for AgentHPO), which is also the task with the lowest HSIC p-value in Table 4; as discussed in Section 5.3, we treat this as an association rather than an established mechanism. On IMDB, CA-HPO still reaches the time-to-target threshold in 1.4 hours, against 3.5 hours for HEBO and 6.3 hours for Random Search. On TREC, ifBO matches CA-HPO's final accuracy (97.6%). Across tasks the dominant gain is in time-to-target rather than in the accuracy ceiling, which is the relevant metric whenever per-evaluation cost is the bottleneck.

Note on Boston Housing. The Boston Housing data include a feature constructed under a racially motivated assumption, and scikit-learn has deprecated the dataset for this reason. We use it only as a generic regression benchmark, note this limitation here, and do not recommend it for future HPO benchmarks.

C.2 Per-Task Independence Tests

Table 4 lists the task-wise HSIC permutation p-values summarized in Section 5.3.

C.3 Cross-Domain Transfer

Table 5 expands Figure 4(b) into per-task numbers. Models are meta-trained on the 5 vision tasks and evaluated on the 4 NLP tasks without any NLP meta-training.

Two observations follow from this table. First, the online speedups of the transformer-based meta-learners (OptFormer, ifBO) decrease to 1.4 to 1.6× under the evaluated domain shift, while CA-HPO

Table 3: Complete results on classical benchmarks. Performance is test accuracy (%) for classification and negative RMSE for regression. Time is wall-clock hours until a method first reaches the time-to-target threshold during search, which for classification is 95% of the best accuracy observed across methods. For the three regression tasks the threshold was applied to the scores monitored during search, not to the final RMSE reported here; Time and Perf. are therefore not directly comparable on these tasks.

	Random		HEBO		CA-HPO		Best Baseline	
Task	Perf.	Time	Perf.	Time	Perf.	Time	Method	Perf.
MNIST	99.2	1.5h	99.4	0.8h	**99.5**	**0.3h**	OptFormer	99.4
F-MNIST	93.1	2.1h	93.8	1.1h	**94.2**	**0.4h**	ifBO	94.0
CIFAR-10	92.5	8.2h	94.1	4.3h	**94.6**	**1.2h**	OptFormer	94.3
CIFAR-100	74.2	12.5h	76.8	6.8h	**77.4**	**2.1h**	AgentHPO	77.1
SVHN	96.1	4.5h	96.8	2.1h	**97.1**	**0.7h**	HEBO	96.8
AG News	93.2	5.8h	94.1	2.9h	**94.5**	**1.0h**	OptFormer	94.3
SST-2	92.1	4.2h	93.4	2.1h	**93.8**	**0.8h**	LLAMBO	93.6
IMDB	91.5	6.3h	92.4	3.5h	92.5	1.4h	**AgentHPO**	**92.8**
TREC	96.8	3.1h	97.4	1.4h	**97.6**	**0.5h**	ifBO	**97.6**
Boston	−3.21	0.15h	−2.89	0.08h	**−2.72**	**0.02h**	HEBO	−2.89
California	−0.48	0.25h	−0.42	0.12h	**−0.39**	**0.04h**	OptFormer	−0.41
Diabetes	−54.2	0.12h	−51.8	0.06h	**−50.1**	**0.02h**	ifBO	−51.2

Table 4: Task-wise HSIC permutation tests of independence between per-probe encodings and hyperparameter configurations, with 200 samples per task and 1,000 permutations. At the unadjusted 0.05 level, independence is not rejected on 11 of 12 tasks. Non-rejection does not establish independence.

Task	HSIC p	Task	HSIC p	Task	HSIC p
MNIST	0.41	AG News	0.27	Boston	0.52
F-MNIST	0.33	SST-2	0.31	California	0.49
CIFAR-10	0.18	IMDB	**0.04**	Diabetes	0.46
CIFAR-100	0.11	TREC	0.24	SVHN	0.22

keeps 2.9×. A plausible reading, consistent with the controlled study in Appendix D, is that the invariance constraint pushes $\hat{u}$ toward optimization-landscape properties rather than input-space statistics.

Second, the cross-domain to in-domain ratio (2.9× vs 5.4×, a speedup factor about 54% as large) indicates how much of the advantage depends on in-domain meta-training. Measured as gain above 1×, about 43% survives without any in-domain meta-training, and this transferable part is already enough to stay ahead of HEBO (2.9× vs 2.2×).

C.4 Encoder Choice

Table 6 compares iTransformer, Set Transformer, and LSTM encoders on the classical benchmarks.

The encoder must summarize a short ($T = 100$), narrow ($D = 5$) multivariate trajectory into $\hat{u}$. In our ablation, the tested Set Transformer and LSTM baselines yield lower speedups (4.8× and 4.1×) and higher within-task variability of the representation ($\sigma_u = 0.061$ and 0.078) than iTransformer ($\sigma_u = 0.042$). We report this as an empirical comparison of our specific implementations; it does not establish which architectural property is responsible.

C.5 Sensitivity Analyses

We examine sensitivity to two key hyperparameters of the framework itself: the micro-sensing duration T (how many steps each probe runs) and the representation dimension d_u (the size of $\hat{u}$).

At $T = 20$ both stability and prediction quality are poor, and they improve steadily up to $T = 100$. At $T = 100$ prediction R^2 reaches 0.82. Beyond $T = 200$ the marginal gain is small (0.84 to 0.85)

Table 5: Cross-domain transfer. Speedup is relative to Random Search.

Method	AG News	SST-2	IMDB	TREC	Avg. Speedup
Random	1.0×	1.0×	1.0×	1.0×	1.0×
HEBO	2.1×	2.3×	1.9×	2.4×	2.2×
OptFormer	1.4×	1.6×	1.2×	1.5×	1.4×
ifBO	1.6×	1.8×	1.4×	1.7×	1.6×
CA-HPO (cross-domain)	2.8×	3.1×	2.2×	3.4×	2.9×
CA-HPO (in-domain)	5.8×	5.2×	4.5×	6.2×	5.4×

Table 6: Encoder ablation on classical benchmarks (averaged).

Encoder	Avg. Speedup	Stability σ_u
iTransformer (default)	5.7×	0.042
Set Transformer	4.8×	0.061
LSTM	4.1×	0.078

while overhead grows linearly. We use $T = 100$ as the default because it sits at the knee of this curve on CIFAR-10.

Table 8 reveals a less obvious tradeoff. Increasing d_u improves the fidelity of $\hat{u}$ as a task descriptor (the R^2 column climbs), but it also tends to increase the within-task spread s_t, since s_t^2 sums over the d_u coordinates, and by (5) a larger s_t requires more probes. The two effects compete: larger d_u reduces residual prediction error but requires more probes, which raises the cold-start cost. The speedup column reports the net effect, which is non-monotonic: it peaks at $d_u = 16$ (5.9×), with $d_u = 8$ close behind (5.7×) and $d_u = 32$ already declining (5.8×). At $d_u = 4$, the bottleneck is representational expressiveness rather than probe budget, and the speedup drops sharply to 3.2×. The probe counts in this table follow the rule $k = d_u + 1$ used for the sweep. We use $d_u = 8$ for classical benchmarks and $d_u = 16$ for LLM tasks. For the LLM tasks the adaptive rule of Section 4.3 starts from $k_0 = 9$ probes and never added probes, since σ_u stayed below τ_{var}; a direct sweep of k at $d_u = 16$ on the LLM tasks would give stronger evidence that nine probes suffice there. The non-monotonicity is consistent with the trade-off in Theorem 3.4 between representation quality and probe cost.

C.6 Sensitivity to Loss Weights, Reliability Threshold, and Predictor Width

Table 9 varies the two loss weights of (6) on the 12 classical benchmarks (5 seeds); R^2 is the prediction quality from $\hat{u}$ on held-out configurations. The defaults $\beta_1 = 1$ and $\beta_2 = 0.5$ were selected on meta-validation. Results near the defaults are numerically close, and we do not read meaning into their small differences. The informative row is $\beta_1 = 0$: without the penalty the representation drifts across configurations, independence is not rejected on only 3 of 12 tasks at the unadjusted 0.05 level, and the speedup falls to the level of a plain meta-surrogate. Removing the reconstruction term ($\beta_2 = 0$) costs less (5.2×).

Table 10 (left) varies the reliability threshold τ_{dist} in the vision-to-NLP transfer experiment, averaged over the four NLP test tasks and five seeds; the default was not retuned on these tasks. A small threshold falls back to Bayesian optimization too often, a large one rarely falls back, and the default gives the best trade-off in this sweep. The right block varies the predictor width on CIFAR-10. Width is not the bottleneck in this in-distribution experiment; this does not establish that the same capacity suffices for LLM fine-tuning or for unfamiliar regions of Λ.

D Controlled Synthetic Study

Construction. Each synthetic task is a stochastic quadratic optimization problem. A task variable z_t, fixed before training, sets the Hessian eigenspectrum and the distribution of gradient noise; the hyperparameters are step size and momentum. Training dynamics are recorded with the same $D = 5$ statistics as in the main experiments, and the encoder, predictor, and representation dimension

Table 7: Sensitivity to micro-sensing duration on CIFAR-10.

Duration T	Stability σ_u	Prediction R^2	Overhead (min)
20	0.18	0.61	0.8
50	0.12	0.73	2.0
100	0.08	0.82	4.0
200	0.07	0.84	8.0
500	0.06	0.85	20.0

Table 8: Sensitivity to representation dimension on CIFAR-10.

d_u	Prediction R^2	Speedup	Probes used	Overhead (min)
4	0.66	3.2×	5	2.2
8	0.82	5.7×	9	4.0
16	0.84	5.9×	17	7.5
32	0.85	5.8×	33	14.5

($d_u = 8$) are as in Appendix F. We meta-train on 1,000 tasks and evaluate on 200 disjoint tasks over five seeds.

Recovery. A linear probe fitted on held-out tasks recovers z_t from $\hat{u}$ with $R^2 = 0.93$. This compares the learned representation with a known ground truth, which the HSIC test of Section 5.3 cannot do.

Falsification. We then add an interaction between task and configuration that depends on λ, so that within the implemented representation and predictor class there is a genuine trade-off between invariance and prediction. The probe R^2 falls to 0.41 and counterfactual prediction error rises by a factor of 2.8. Predictive performance under the invariance constraint thus deteriorates when the price of invariance within the implemented model classes is high. The result does not show that no useful invariant representation exists for unrestricted function classes, and we make no such claim.

E Probe Budget

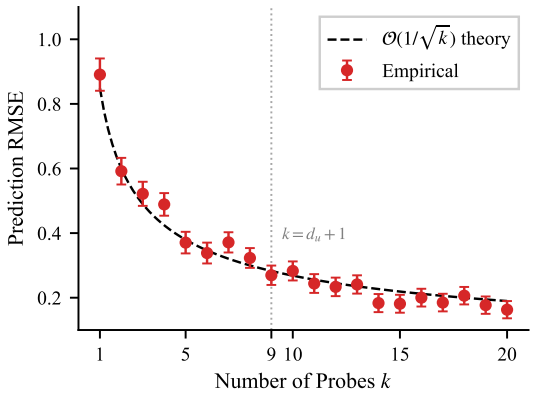

Figure 5: Counterfactual prediction error versus number of probes k. The error decreases at a rate close to $1/\sqrt{k}$ and stabilizes around $k = d_u + 1$.

Figure 5 gives an empirical check on term (I) of Theorem 3.4, whose leading part scales as $s_t/\sqrt{k}$. A curve of the form $a/\sqrt{k}$ fits the empirical prediction error well. The curve flattens around $k = d_u + 1 = 9$, beyond which more probes still help but with diminishing returns; this knee motivates the default $k_0 = 9$ in Section 4.3.

Table 9: Sensitivity to the invariance weight β_1 (left) and the reconstruction weight β_2 (right), 12 classical benchmarks.

β_1	σ_u	HSIC non-rejections	Speedup
0	0.34	3/12	2.6×
0.5	0.06	10/12	5.4×
1	0.042	11/12	5.7×
5	0.038	11/12	5.6×
50	0.037	11/12	5.4×

β_2	Held-out R^2	Speedup
0	0.78	5.2×
0.5	0.82	5.7×
1	0.82	5.6×

Table 10: Left: reliability threshold in cross-domain transfer. Right: predictor width on CIFAR-10.

τ_{dist}	Speedup	Fallback rate
0.1	2.4×	41%
0.3	2.9×	18%
0.5	2.7×	7%

Predictor size	Held-out R^2	Speedup
18K	0.82	5.7×
72K	0.83	5.7×
290K	0.83	5.6×

F Implementation Details

Encoder $\mathcal{E}$. Input $\mathbf{X}^{(T)} \in \mathbb{R}^{100\times 5}$ passes through a 1D convolutional embedding (kernel size 5, 32 channels) along the time dimension, layer normalization, and ReLU. Sinusoidal positional encodings are added before two iTransformer layers (4 heads, $d_{\text{model}} = 64$, FFN dim 256, GELU, dropout 0.1). Mean pooling over the D variate tokens followed by a linear projection produces $\hat{u} \in \mathbb{R}^{d_u}$. Total parameters: ≈145K. Default $d_u = 8$ for classical benchmarks, $d_u = 16$ for LLM tasks.

Generator G_X. Categorical components of λ are embedded with dimension 8; continuous components are normalized as in Section 4.3 and linearly projected. Concatenated with $\hat{u}$, mapped through MLP ($d_u + d_\lambda \to 128 \to 64$, ReLU, dropout 0.1), and decoded by a 2-layer iTransformer with tanh activation scaled to per-feature ranges.

Predictor f. Linear $(d_u + d_\lambda) \to 128$, BatchNorm, ReLU, Dropout 0.2, Linear $128 \to 1$, with sigmoid output for classification and identity for regression. ≈18K parameters.

Meta-training. AdamW, learning rate 10^{-3}, weight decay 10^{-4}, batch size 32, gradient clipping at norm 1.0. Linear warmup (5 epochs) and cosine annealing. Early stopping with patience 10. Gaussian noise ($\sigma = 0.01$) added to micro-dynamics as light data augmentation.

Counterfactual optimization. Adam, learning rate 10^{-2}, 100 steps, 8 random restarts. Gumbel-Softmax temperature 0.5 for categorical components.

Compute. All experiments on NVIDIA A40 48GB GPUs. Foundation model experiments use one GPU; classical benchmarks use 4 GPUs in parallel for baseline evaluations. PyTorch 2.0.1, CUDA 11.8. Seeds $\{42, 123, 456, 789, 1011\}$.

Meta-training data and offline cost. 5,000 training trajectories in aggregate, with 10 hyperparameter configurations per meta-training task drawn from the joint search space via Sobol sequences. Collecting the corpus costs 480 GPU-hours and training the encoder and predictor about 8 GPU-hours, for a one-off offline cost of about 488 GPU-hours.

G Failure Cases

CA-HPO is built on two operational assumptions: that the task is constant during the optimization process (so that $u^*(t)$ is well-defined), and that the configuration space is large enough that probing is worthwhile compared with exhaustive search. Both assumptions can fail in practice. The following cases describe regimes where the framework's gains erode or disappear, and how the reliability check in Algorithm 1 behaves in each.

Extreme distribution shift. Meta-training on CIFAR-10 (32×32, ResNet-18) and testing on ImageNet [Deng et al., 2009] (224×224, ResNet-50) reduces speedup from 5.7× to 1.3×. This experiment changes task, input resolution, and model capacity together, so it is a stress test under distribution shift rather than an isolated study of any one factor. In this regime the counterfactual candidates after the first evaluation are frequently far from the evaluated configurations, the reliability check falls back to Bayesian optimization, and performance degrades to roughly that of Bayesian optimization. The first counterfactual recommendation is not protected by the check. We recommend including in meta-training at least one task family that resembles the deployment regime in scale and architecture.

Non-stationary tasks. Continual learning settings violate the constancy of the task. In a curriculum experiment on CIFAR-100 with progressive difficulty, the initial $\hat{u}_t$ becomes stale after the curriculum stage transitions: the optimal learning rate during the easy phase is roughly 3× higher than during the hard phase, but $\hat{u}_t$ was estimated under the easy phase and so the counterfactual surrogate's recommendation persists past its useful range. Periodic re-sensing at stage transitions restores accuracy, at the price of another kT training steps of overhead for each re-sensing. A more principled fix would be to model the non-stationarity directly, for example by treating the task as a slowly-varying $u^*(t, s)$ where s is the curriculum stage, and predicting u^* at each new stage from the history of past representations; we leave this to future work.

Very small search spaces. When the configuration space has only a handful of candidates, evaluating all of them costs only a few more full runs than CA-HPO typically uses (five on GSM8K), and it returns the best candidate without relying on meta-training, so the framework offers little benefit. CA-HPO is intended for medium-to-high-dimensional mixed spaces (such as the 6-dimensional LLM search space) where exhaustive enumeration is infeasible.

H Computational Cost

We break down CA-HPO's wall-clock time on GSM8K to make the dominant cost transparent. The breakdown also clarifies which components of the framework are actually the source of the speedup, in case future work seeks to remove or improve them.

Table 11: Wall-clock breakdown for CA-HPO on GSM8K. Comparable HEBO requires ≈ 36 hours; Random Search exceeds 48 hours.

Operation	Time (min)	Percentage
Micro-sensing (9 probes × 100 steps)	6.0	2.5%
LoRA adapter swapping (9 switches)	0.5	0.2%
Representation extraction (9 forward passes)	0.1	<0.1%
Counterfactual optimization	0.1	<0.1%
Full training evaluations (5 runs × 45 min)	225.0	94.9%
Predictor refinement (5 epochs)	5.3	2.2%
Total	**237**	**100%**

The breakdown gives a precise account of where the speedup comes from. Full training evaluations dominate the total at 94.9%, as one would expect on an LLM task where each run is expensive. Everything that is unique to CA-HPO (micro-sensing, representation extraction, counterfactual optimization, predictor refinement) accounts for under 5.1% of the wall-clock time combined. The framework therefore does not introduce a new computational bottleneck; instead, it reduces the number of full evaluations required, from approximately 48 for HEBO and 64 for Random Search to 5 for CA-HPO.

This decomposition has two practical consequences. First, the online speedup is bounded by the ratio of full-evaluation counts (about 10 to 13 times on this task, against HEBO and Random Search), and additional engineering of the encoder or surrogate cannot push the speedup substantially higher unless the per-evaluation cost itself is reduced (for example, by using fewer epochs per full evaluation, which would amount to a multi-fidelity extension of the framework). Second, this breakdown excludes the one-off offline cost of about 488 GPU-hours (Appendix F). With 3.95 GPU-hours per CA-HPO search

against 36 for HEBO, the offline cost breaks even after $488/(36 - 3.95) \approx 16$ GSM8K-equivalent tasks, assuming the corpus is reusable and future tasks have comparable cost. For a single cold-start task CA-HPO is more expensive than HEBO.

I Extended Interpretability

This section provides two visualization-based diagnostics that probe the geometry of the learned representation $\hat{u}$ and the kind of signal that micro-sensing actually extracts from training dynamics. Neither is a formal validation of any theorem; both are meant to give the reader an intuitive sense of what has been learned.

I.1 t-SNE Projection of Task Representations

Figure 6 shows a t-SNE projection of $\hat{u}$ across the 12 classical benchmarks. Tasks form clear clusters despite hyperparameter variation. This is a qualitative diagnostic of within-task invariance; whether the representation is sufficient to predict performance is assessed by held-out prediction, not by the clustering.

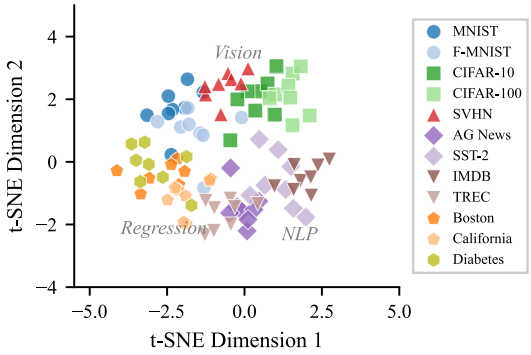

Figure 6: t-SNE projection of task representations (perplexity 30). Each point is one (task, λ) pair; clustering by task is consistent with within-task invariance.

Points from the same task stay together across configurations, and the three tabular regression tasks occupy a region of their own. Because t-SNE does not preserve global distances, we do not interpret the overall layout beyond these qualitative observations.

I.2 Gradient Variance as a Task Signature

Figure 7 plots gradient variance trajectories over the first 100 steps for MNIST, F-MNIST, and CIFAR-10. Each task exhibits a distinct pattern (rapid decay for MNIST, moderate oscillation for F-MNIST, sustained variance for CIFAR-10), and the patterns persist across a 10-fold change in learning rate, supporting the use of gradient variance as a task-discriminative feature.

The persistence under learning-rate change is the empirical content of the invariance constraint. Changing the learning rate directly changes update magnitudes and can alter subsequent gradient statistics through the optimization trajectory. The figure shows that the absolute scale of the variance shifts with the learning rate, but the relative ordering of the three tasks (MNIST below F-MNIST below CIFAR-10) is preserved, as is the qualitative shape of each curve (rapid early decay vs sustained plateau). A plausible reading is that the encoder, trained with $\mathcal{L}_{\text{inv}}$, relies on such ordering and shape rather than on absolute magnitudes, which would be consistent with the stability in Figure 4(a).

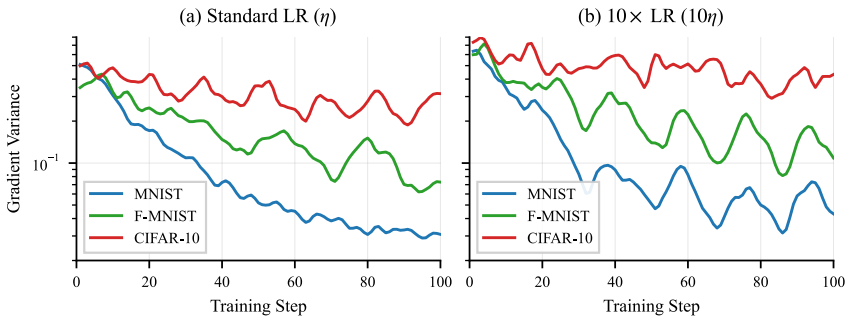

Figure 7: Gradient variance trajectories under standard (left) and 10× elevated (right) learning rates. Distinct task signatures persist across learning-rate changes.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the three contributions (constrained definition of the target, three theoretical guarantees, complete instantiation with empirical gains), all of which are substantiated in Sections 3 and 5. Speedups are stated per track and relative to a named baseline, as warm-start results, the offline cost is reported in Section 5.2, and the scope of the framework is stated in Section 2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 (Limitations and Conclusion) lists four premises that bound the framework's applicability, including offline cost, surrogate capacity, and the limited role of the reliability check; Appendix G discusses three concrete failure regimes, and Appendix D reports a controlled case where the invariance premise fails.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Each theorem states its conditions, with Assumption 3.2 collecting the boundedness and Lipschitz conditions. Complete proofs of Theorems 3.1, 3.3, and 3.4, Lemma A.1, and Proposition A.2 are given in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Architecture details, training hyperparameters, seeds, and meta-training data composition are documented in Section 4 and Appendix F. Algorithm 1 provides the full inference pseudocode.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often

one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: All datasets used (GSM8K, PubMedQA, ARC, MMLU, MNIST, CIFAR, etc.) are public benchmarks and are properly cited. Code is not included with the paper; reproduction details, including hyperparameters, architecture, seeds, and the inference pseudocode, are in Appendices B and F.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

Justification: Training and inference settings, optimizer choices, learning rates, batch sizes, and seeds are documented in Section 5 (Setup) and Appendix F; baseline configurations follow original papers' recommended settings.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: All main results are averaged over 5 seeds; error bars in Figure 3 show ± 1 standard deviation. Task-wise HSIC permutation tests in Section 5 report p-values from 1,000 permutations at the unadjusted 0.05 level, and the per-dimension IMDB tests are reported with Holm correction.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: GPU type (NVIDIA A40 48GB), per-task wall-clock times, and the one-off offline cost (480 GPU-hours for corpus collection plus about 8 GPU-hours for model training) are reported in Section 5.2 and Appendix F, with an online cost breakdown in Appendix H.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses only publicly available benchmarks and does not involve human subjects, sensitive data, or deployment risks; we have reviewed and conform to the NeurIPS Code of Ethics. The Boston Housing dataset contains a feature constructed under a racially motivated assumption; we use it only as a generic regression benchmark, disclose this in Appendix C, and do not recommend it for future benchmarks.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The work is a methodological contribution to hyperparameter optimization, with no foreseeable direct societal impact beyond reducing the compute and energy cost of repeated model tuning once the offline meta-training cost is amortized.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release new datasets or pretrained models with misuse risk; the contribution is an HPO method operating on user-supplied training trajectories.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and models are cited with their original references in Section 5. Licenses as stated by the official release pages or dataset cards: Llama-3-8B (Meta Llama 3 Community License); GSM8K, PubMedQA, MMLU, and Fashion-MNIST (MIT); ARC (CC BY-SA 4.0); Dolly-15k (CC BY-SA 3.0); OpenAssistant OASST1 (Apache-2.0); Alpaca data (CC BY-NC 4.0); BERT-base (Apache-2.0); ResNet implementations from torchvision (BSD-3-Clause); SVHN (non-commercial use only); ImageNet (ImageNet terms of access, non-commercial research and education). For MNIST, CIFAR-10/100, AG News, SST-2, IMDB, TREC, California Housing, Diabetes, and Boston Housing we did not find an explicit license on the original release pages; we use them only for non-commercial research and follow the terms stated by their providers where available.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: We do not release new datasets, models, or code. The CA-HPO method is documented in Section 4 and Appendices B and F.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects, so IRB review does not apply.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are evaluation targets (Llama-3-8B fine-tuning is one of the benchmarks), not a component of the proposed methodology. The CA-HPO method itself contains no LLM in its pipeline.

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.