Zero-Shot Bayesian Optimization with TabPFN: Competitive with State-of-the-Art without Per-Task Training

Theodore Rogers[1] **Srividya Ponnada**[1]

[1]Amazon

Abstract Bayesian hyperparameter optimization typically requires fitting a surrogate model to each new task, incurring per-task training cost that grows with the number of observations and limits deployment flexibility. We show that TabPFN v2 (Hollmann et al., 2025), a pretrained tabular foundation model never trained on Bayesian optimization data, can serve as a drop-in zero-shot BO surrogate, eliminating the per-task fitting step. On HPO-B v3-test (16 search spaces, $n = 430$ matched cells, 50-trial budget), TabPFN-direct BO wins against 5 of 7 HPO-B baselines and ties the other 2 without any per-task training, matches DRE at half the evaluation budget (a 2× sample-efficiency advantage at budgets ≥ 25), and wins against the concurrent ZeroShotOpt (Meindl et al., 2025) on 13/16 HPO-B spaces at step 50 and 11/11 YAHPO-Gym scenarios.

The enabling mechanism is calibration. TabPFN's predictive intervals achieve empirical coverage of 0.94–0.95 against the 95% target on both benchmarks, and calibrated uncertainty makes acquisition functions equivalent: EI, LogEI, and UCB variants produce optimization outcomes statistically equivalent to our UCB default under pre-specified paired equivalence tests, and a smaller cross-surrogate check on SMAC-GP is consistent with this pattern. The acquisition choice is therefore not a meaningful lever; practitioners can deploy any standard acquisition function without per-task selection, removing a tuning decision from the deployment pipeline. The full statistical protocol, effect sizes, and multiple-testing control are reported in §4.

1 Introduction

Bayesian optimization (BO) for hyperparameter tuning is effective but operationally expensive: each new task requires fitting a surrogate model – a Gaussian process (Snoek et al., 2012), a random forest (Hutter et al., 2011), a tree-structured Parzen estimator (Bergstra et al., 2011), a warped Gaussian process (Cowen-Rivers et al., 2022), or a BO-specialized transformer – whose hyperparameters and inductive biases must be tuned per problem. This per-task surrogate training is a deployment barrier. It adds a stage to the optimization pipeline, it requires meta-data when transferring across tasks (Wistuba and Grabocka, 2021; Khazi et al., 2023), and it consumes wall-clock that could otherwise go toward objective evaluations. Recent zero-shot BO methods (Chen et al., 2022; Müller et al., 2023; Rakotoarison et al., 2024; Meindl et al., 2025) address this by pretraining a transformer on BO trajectories or synthetic objective functions, producing a surrogate that requires no per-task fitting. We ask whether a more general abstraction works: can a tabular foundation model, pretrained on synthetic classification and regression tasks rather than BO trajectories, serve as a zero-shot BO surrogate?

Our answer is a three-step argument. First, TabPFN v2 (Hollmann et al., 2023, 2025) is a tabular foundation model, trained in-context on synthetic regression and classification tasks, designed to produce well-calibrated predictive posteriors on held-out tabular data. Second, we show that this calibration transfers to BO-adjacent distributions: TabPFN's predictive intervals are calibrated

on both HPO-B v3-test ($c_{95} = 0.94$, with 15 of 16 search spaces within ±0.10 of nominal) and YAHPO-Gym ($c_{95} = 0.95$ on all 14 scenarios within ±0.05). Third, calibrated σ is what allows standard BO acquisition functions (UCB-Gaussian, EI, LogEI) to work correctly without per-task tuning: we verify that EI ≡ LogEI ≡ UCB at paired TOST margins $\delta = 0.01$ on HPO-B and $\delta = 0.02$ on YAHPO. The combination – a calibrated tabular foundation model plus any standard acquisition function over Sobol candidates – matches or beats seven HPO-B baselines (including classical GP-BO and TPE) and matches DRE at half the evaluation budget, without any per-task fitting.

Prior transformer-for-BO methods (Chen et al., 2022; Müller et al., 2023; Rakotoarison et al., 2024; Meindl et al., 2025) pretrain on BO trajectories or synthetic objective functions. Our contribution is showing that an off-the-shelf tabular foundation model, never trained on BO data, is competitive with these BO-specialized transformers on HPO-B v3 and YAHPO-Gym. The novelty wedge is domain generality: TabPFN v2 handles HPO-B v3's mixed discrete-continuous search spaces (d up to 18) and YAHPO-Gym's conditional hyperparameter spaces (d up to 39 with up to 36 conditional parameters), exceeding PFNs4BO's released 18-parameter encoder cap. GIT-BO (Yu et al., 2025) concurrently uses TabPFN v2 as a BO surrogate but targets high-dimensional synthetic functions and does not evaluate on HPO benchmarks or characterize acquisition equivalence and predictive calibration.

Contributions.. We make three independent claims.

(1) Head-to-head benchmarking.. TabPFN-direct BO matches or beats all seven HPO-B baselines evaluated at matched protocol (PFNs4BO-HEBO+, PFNs4BO-BNN, DRE, classical GP-BO with UCB and EI, TPE, Random; plus the concurrent zero-shot method ZeroShotOpt (Meindl et al., 2025)). Five comparisons have 95% CI on Δ excluding zero; DRE-50 and GP-BO-EI are at parity. Across a pre-specified primary family of 10 claims at Bonferroni $\alpha/10 = 0.005$, 8 survive. Against DRE specifically, TabPFN-50 matches DRE-100 at half the evaluation budget, a 2× sample-efficiency advantage without per-space meta-training.

(2) Acquisition equivalence.. EI, LogEI, and UCB-Bars are equivalent to UCB-Gauss via paired TOST at $\delta = 0.01$ on HPO-B ($n = 430$), and on YAHPO at $\delta = 0.02$ ($n = 53$); a cross-surrogate replication on SMAC-GP (14 scenarios × 5 seeds) yields 0/42 per-scenario Wilcoxon rejections at Bonferroni $\alpha = 0.00119$. The acquisition choice is not a lever in this regime. LogEI produces trajectories byte-identical to EI on 97% of runs and is treated as a monotonicity validation rather than an independent acquisition.

(3) Predictive-interval calibration.. TabPFN predictive σ is calibrated on both benchmarks: c_{95} within ±0.05 of nominal on YAHPO (14/14 scenarios) and within ±0.10 on 15 of 16 HPO-B spaces, with the single outlier (`ss5965`, $c_{95} = 0.84$) discussed as a limitation in §5.

2 Related Work

Classical Bayesian optimization for hyperparameter tuning.. The canonical formulation fits a Gaussian process surrogate (Snoek et al., 2012; Frazier, 2018), a random forest (Hutter et al., 2011; Lindauer et al., 2022), a tree-structured Parzen estimator (Bergstra et al., 2011), or a warped Gaussian process with input/output transformations (Cowen-Rivers et al., 2022). All require per-task surrogate training and per-step refitting. Foundational acquisition functions include expected improvement (Jones et al., 1998) and upper confidence bound with regret guarantees (Srinivas et al., 2010).

Transfer and meta-learning surrogates.. FSBO (Wistuba and Grabocka, 2021) meta-learns a deep-kernel GP surrogate across HPO-B search spaces; it requires a pretraining phase on the target benchmark's meta-dataset. DRE (Khazi et al., 2023) trains per-space ranking ensembles and is a

stronger HPO-B baseline than FSBO (§4.2). TabPFN is complementary: a single pretrained model, no per-space or per-benchmark meta-training.

Pretrained transformers as BO surrogates.. PFNs (Müller et al., 2022) introduced prior-data fitted networks for in-context Bayesian inference. PFNs4BO (Müller et al., 2023) is a PFN pretrained on synthetic BO trajectories with HEBO+ and BNN heuristic variants. TabPFN (Hollmann et al., 2023, 2025) extends the PFN recipe to general tabular classification and regression with synthetic SCM-based pretraining; we use it off-the-shelf as a BO surrogate, bypassing BO-trajectory pretraining entirely. OptFormer (Chen et al., 2022) pretrains a transformer on Vizier-style BO trajectories across many tasks; it differs from TabPFN-direct in requiring BO data during pretraining. ZeroShotOpt (Meindl et al., 2025) is concurrent: a transformer pretrained by offline reinforcement learning on trajectories from 12 BO variants; comparison in §4.3. ifBO (Rakotoarison et al., 2024) trains a PFN for multi-fidelity freeze-thaw BO (grey-box setting; complementary to our black-box regime). LLAMBO (Liu et al., 2024) uses LLMs as both surrogate and acquisition optimizer via prompting; it requires orders of magnitude more inference compute per step than a single TabPFN forward pass. GIT-BO (Yu et al., 2025) concurrently uses TabPFN v2 as a BO surrogate but targets high-dimensional synthetic functions ($d \geq 100$) and does not evaluate on HPO benchmarks or characterize acquisition equivalence and predictive calibration, though a post-hoc replication under GIT-BO's TabPFN implementation is consistent with our equivalence finding (§4.4, Appendix A.14).

Calibration in Bayesian optimization.. Kuleshov et al. (2018) characterize calibration of Bayesian predictive intervals on regression tasks and propose an isotonic-regression recalibration procedure inspired by Platt scaling. Stanton et al. (2023) apply conformal prediction sets to BO to obtain coverage guarantees even when the surrogate is misspecified. Our calibration result is observational rather than corrective: TabPFN's posteriors are sufficiently calibrated out-of-the-box on HPO-B and YAHPO that no recalibration step is needed for standard acquisitions to behave correctly.

Benchmark choice.. HPO-B (Arango et al., 2021) serves as the primary evaluation benchmark because it uses real hyperparameter evaluations (not surrogate-based landscapes) and ships matched-protocol baselines. YAHPO-Gym (Pfisterer et al., 2022) provides ResNet-based surrogate models trained on OpenML evaluations, with up to 36 conditional parameters per scenario, exposing surrogate handling of conditional spaces. Independent surveys (Kégl, 2023) motivate HEBO's inclusion as a strong modern baseline.

Acquisition-function ablations.. Müller et al. (2023) observe that EI and UCB perform similarly on HPO-B; we confirm this on HPO-B (TOST at $\delta = 0.01$) and YAHPO (TOST at $\delta = 0.02$) and extend to a cross-surrogate replication on SMAC-GP. Ament et al. (2023) show that LogEI resolves gradient pathologies of EI; our byte-identical finding at 97% of runs confirms the monotonicity in this HPO regime.

3 Method

Our method is TabPFN-direct BO: a pretrained tabular transformer used as a zero-shot in-context surrogate, with UCB-Gaussian acquisition and Sobol candidate generation. The full procedure (Algorithm 1, Appendix A.1): cold-start with $\max(2d, 10)$ random points, then at each step score 5000 Sobol candidates via a single TabPFN forward pass and select the LCB argmin ($\mu - \beta\sigma$, $\beta = 1$). The following subsections describe each component.

3.1 TabPFN-direct surrogate

Our surrogate is TabPFN v2 (Hollmann et al., 2025) used in-context: we pass the observations $\mathcal{D}_t = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{t}$ as training context and a batch of candidate configurations as the query, then read off predictive distributions in a single forward pass. The model updates no weights; class-level

weight caching makes each `ask()` call a stateless GPU forward pass costing ~ 10–400 ms at the sample sizes we study. Appendix A.9 describes the posterior output format.

Amortized conditioning.. Per-task GP, RF, and GBT surrogates refit hyperparameters or retrain trees at every BO step. Our surrogate does neither: TabPFN v2 was pretrained offline on synthetic tabular tasks, and at inference time conditions on the observed configurations via a single $\mathcal{O}(N^2)$ attention forward pass with no gradient step or hyperparameter optimization. We call this *amortized conditioning*: the cost of learning a posterior over functions is paid once at pretraining, and every BO step pays only the forward-pass cost. This yields the wall-clock figures in Appendix A.7 and removes per-task surrogate hyperparameters.

3.2 Condition-aware input encoding

YAHPO-Gym scenarios contain conditional parameters whose activity depends on other hyperparameter values (e.g., `max_depth` is inactive when `booster=gblinear`). We replace inactive values with `NaN`, giving TabPFN an explicit "this parameter did not apply" signal, the semantically correct encoding that prevents the surrogate from learning spurious correlations with placeholder values. The masking rule is determined by the ConfigSpace DAG structure (the directed acyclic graph of conditional dependencies declared in each search space's definition), not by observed optimization performance. The one exception is `nb301`, a NAS cell-graph scenario where "inactive" edge values still encode topological information; on this space we preserve the raw encoding.

3.3 Acquisition and candidate generation

Our default acquisition is UCB-Gaussian (Srinivas et al., 2010) with $\beta = 1$:

$$a(\boldsymbol{x}) = \mu(\boldsymbol{x}) - \beta\sigma(\boldsymbol{x}), \tag{1}$$

where μ, σ are the mean and standard deviation of TabPFN's posterior. The sign convention minimizes losses by default; for maximization scenarios we negate the objective.

At each step we generate $n_c = 5000$ Sobol candidates in $[0, 1]^d$, score each, and pick the argmin. A cold-start period of $\max(2d, 10)$ uniform random points precedes surrogate-driven selection.

3.4 Computational budget

We use 200 evaluations per run (YAHPO-Gym default). Per-evaluation GPU latency is ~ 0.02 s (median across scenarios; compute provenance in Table 7); end-to-end wall-clock ranges from 3.5 s (low-d spaces) to ~ 100 s ($d = 29$ with conditionals), with a median of 4.4 s per 200-evaluation run.

3.5 Scope

We do not fine-tune TabPFN on HPO data, select kernels, or tune any per-task hyperparameters. The only input-space decision is condition-aware encoding (§3.2), determined by the ConfigSpace DAG structure of each search space rather than by observed optimization performance. The surrogate remains fixed at its publicly released pretrained weights.

3.6 Statistical analysis

We pre-specify a *primary family* of 10 head-to-head claims (TabPFN vs. each baseline at each benchmark-budget cell) and control family-wise error at Bonferroni $\alpha/10 = 0.005$; 8 of 10 survive. The two non-survivors (PFNs4BO-HEBO+ raw $p = 0.013$, and SMAC-GP per-scenario Wilcoxon at $n = 5$) are reported with 95% confidence intervals and marked as marginal in the body text. Beyond the primary family, we report a broader panel of 132 descriptive tests (per-space sign-tests, per-scenario Wilcoxons, acquisition-equivalence TOSTs, calibration intervals) under Benjamini-Hochberg FDR control at $q = 0.05$. Paired two one-sided tests (TOST; Lakens et al., 2018) underpin

equivalence claims: a comparison passes TOST at margin δ if both one-sided tests reject at α, confirming the true effect lies within $\pm\delta$. We use margins $\delta = 0.01$ on HPO-B (pooled cell-level, $n = 430$) and $\delta = 0.02$ on YAHPO ($n = 53$); rationale is in Appendix A.13. Paired Wilcoxon signed-rank replaces t-tests for finite-sample and non-normal data. Per-space and per-scenario sign-tests use the exact two-sided binomial under H_0: $p = 0.5$. Cohen's d (paired: $d = \mu_\Delta/\sigma_\Delta$) and bootstrap 95% CIs accompany every effect-size claim. Bootstrap 95% CIs use the percentile method with $B = 10{,}000$ resamples and seed 42, paired at the cell level (space, dataset, seed) on HPO-B.

4 Experiments

4.1 Setup

Benchmarks.. We evaluate on two complementary benchmarks. HPO-B (Arango et al., 2021) serves as the primary benchmark for head-to-head comparison: 16 search spaces from OpenML tasks with real hyperparameter evaluations, $d = 2$–18, with the 50-evaluation reporting checkpoint of the recommended HPO-B protocol (Arango et al., 2021) (which spans 1–100 trials with canonical CD-diagram snapshots at 25, 50, and 100) extended to 200 evaluations on the spaces where deeper budgets are tractable. YAHPO-Gym (Pfisterer et al., 2022) serves as the secondary benchmark for acquisition equivalence and σ-calibration: 14 scenarios spanning $d = 3$–39 (inclusive of the fidelity parameter; YAHPO's native count excludes it) and 0–36 conditional parameters, evaluated on the YAHPO ResNet-based surrogate at 200 evaluations.

Baselines.. HPO-B head-to-head: seven matched-protocol baselines: PFNs4BO-HEBO+ and PFNs4BO-BNN (Müller et al., 2023) (released encoder caps at $d = 18$; cannot evaluate on 3 of our 14 YAHPO scenarios where $d \in \{28, 34, 38\}$), DRE (Khazi et al., 2023), classical GP-BO with both UCB ($\beta = 0.1$) and EI acquisitions (Snoek et al., 2012), TPE (Bergstra et al., 2011) via Optuna (Akiba et al., 2019), and Random Search, all at matched 50-trial budgets; plus concurrent zero-shot ZeroShotOpt (Meindl et al., 2025) at steps 50 and 100 (architectural cap, see §4.3). YAHPO acquisition equivalence: TabPFN with UCB-Gauss, UCB-Bars, EI, LogEI, and Thompson sampling; a cross-surrogate replication swaps TabPFN for SMAC's GP surrogate (Lindauer et al., 2022). We use canonical hyperparameters throughout. The GP-BO baseline fits a BoTorch `SingleTaskGP` (Matérn-5/2 kernel with ARD, standardized outcomes), refit at every step by type-II maximum likelihood on the exact marginal log-likelihood – the library-default point estimate, not the MCMC-marginalized treatment of Snoek et al. (2012). Its UCB acquisition uses $\beta = 0.1$ because that is the reference default of HPO-B's own BoTorch example implementation (Arango et al., 2021); we retain it for protocol fidelity rather than as a tuned choice. It places less weight on exploration than canonical settings (our own method's LCB uses $\beta = 1$), which is why we additionally run GP-BO with EI (Table 1).

Aggregation and pairing.. HPO-B comparisons are at the cell level (space $\times$ seed $\times$ budget, $n = 430$ matched cells) with paired Wilcoxon signed-rank supplemented by per-space sign-test. YAHPO comparisons are at the scenario level ($n = 53$ paired observations for the main TOST) with paired Wilcoxon. We report Δ = TabPFN $-$ baseline in normalized accuracy (1$-$ normalized regret in HPO-B's notation (Arango et al., 2021); HPO-B and ZeroShotOpt's evaluation harnesses both expose this scalar) or negated regret (YAHPO); positive Δ favors TabPFN.

4.2 HPO-B: head-to-head against BO-specialized baselines

We evaluate TabPFN-direct BO on all 16 search spaces in the HPO-B v3 test split ($n = 430$ matched cells across space, seed, and budget) at the canonical 50-evaluation budget. Table 1 gives the head-to-head results. TabPFN-50 matches or beats all seven HPO-B baselines: five with 95% CI excluding zero (HEBO+, BNN, GP-BO-UCB, TPE, Random), plus parity with DRE-50 and GP-BO-EI (both CIs straddle zero, $|d| < 0.1$). Against DRE-100 at double the budget, TabPFN-50 matches

Table 1: HPO-B v3-test head-to-head, TabPFN-50 vs. baselines. Δ is normalized-accuracy difference pooled across $n = 430$ matched cells. Positive Δ favors TabPFN. All CIs use percentile bootstrap, $B = 10{,}000$, seed 42. Baseline citations: PFNs4BO (Müller et al., 2023), DRE (Khazi et al., 2023), GP-BO (Snoek et al., 2012) (fitting specification in §4.1), TPE (Bergstra et al., 2011; Akiba et al., 2019).

Baseline	Training	Δ (95% CI)	Significance
PFNs4BO-HEBO+	Pretrained + HEBO+	+0.020 [+0.009, +0.032]	$p = 0.013^{\dagger}$
PFNs4BO-BNN	Pretrained + BNN ensemble	+0.017 [+0.009, +0.027]	$p = 5.08 \times 10^{-5}$
DRE-50	Per-space meta-train	+0.008 [−0.002, +0.020]	$d = +0.07$ (parity)
DRE-100	Per-space meta-train, 2× budget	−0.0002 [−0.010, +0.011]	$d = -0.002$ (parity)
GP-BO (UCB, $\beta = 0.1$)	None (per-step GP fit)	+0.033 [+0.022, +0.044]	$p = 1.7 \times 10^{-11}$
GP-BO (EI)	None (per-step GP fit)	+0.006 [−0.001, +0.013]	$d = +0.08$ (parity)
TPE (Optuna)	None (per-step KDE)	+0.038 [+0.027, +0.050]	$p = 1.1 \times 10^{-22}$
Random Search	None	+0.046 [+0.033, +0.060]	$p = 1.77 \times 10^{-23}$

†Raw $p = 0.013$ does not survive primary-family Bonferroni $\alpha/10 = 0.005$. The 95% CI [+0.009, +0.032] excludes zero; we cite the CI as primary evidence.

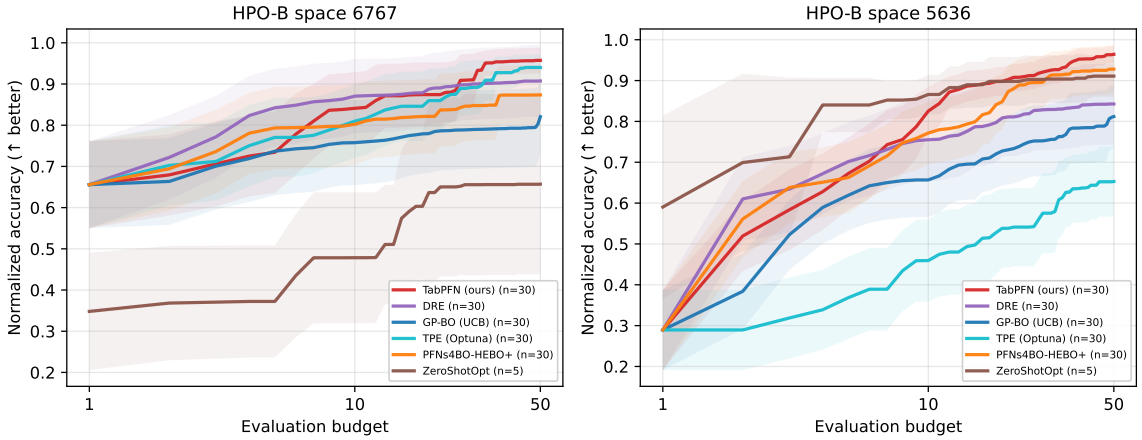

Figure 1: Best-so-far normalized accuracy vs. evaluation budget on two non-saturated HPO-B v3-test search spaces (`ss6767` and `ss5636`). Lines are means over $n = 30$ runs per method (6 test datasets × 5 seeds; ZeroShotOpt: $n = 5$ on its supported subset); shaded bands are 95% confidence intervals. DRE (`ss6767`) and ZeroShotOpt (`ss5636`) lead at small budgets, consistent with the cold-start behavior in §4.2; TabPFN overtakes by ∼25 evaluations, attains the highest budget-50 accuracy on both spaces, and reaches DRE's budget-50 accuracy at roughly half the budget (the 2× sample-efficiency advantage). The full 16-space grid is in Appendix A.16.

($\Delta = -0.0002$, $p = 6.2 \times 10^{-12}$, $n = 430$), a 2× sample-efficiency advantage at budgets ≥ 25. At cold-start budgets (k=10 vs. DRE-20), the comparison reverses ($\Delta = -0.017$, 95% CI [−0.025, −0.009], Cohen's $d = -0.20$): DRE benefits more than TabPFN from each additional trial near cold-start. Appendix A.17 gives the full $k \in \{10, 25, 50, 75, 100\}$ grid. The GP-BO-EI parity shows TabPFN's zero-shot advantage holds against the stronger of the two default GP-BO acquisitions.

Per-space breakdown.. The PFNs4BO-HEBO+ loss to TabPFN is distributed across 11 of 16 HPO-B spaces, with the remaining 5 showing TabPFN deficits within the trivial-Cohen's-d range. DRE-50 is within parity on 15/16 spaces; the one exception, `ss5970`, favors DRE at Holm-adjusted $p = 0.024$ (a raw DRE advantage of 0.007 at $n = 30$ cells). Full per-space tables are in Appendix A.16.

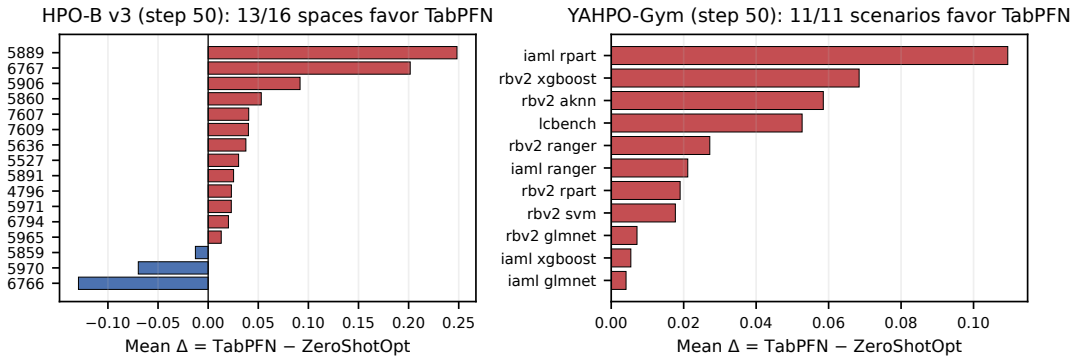

Figure 2: Per-space (HPO-B, left) and per-scenario (YAHPO-Gym, right) mean Δ = TabPFN − ZSO at step 50. Red favors TabPFN, blue favors ZSO. 13/16 HPO-B spaces and 11/11 YAHPO scenarios favor TabPFN. The three spaces favoring ZSO are low-d spaces closer to its 2D–10D pretraining regime.

4.3 HPO-B and YAHPO: comparison to concurrent ZeroShotOpt

ZeroShotOpt (Meindl et al., 2025) is a concurrent zero-shot black-box optimization method that pretrains a transformer on trajectories from 12 Bayesian optimization variants. Like our method, it requires no per-task training. We evaluate ZSO on the same 16-space HPO-B v3-test split at matched budgets, and on 11 YAHPO scenarios.

TabPFN-direct BO beats ZSO on 13/16 HPO-B spaces at step 50 ($p = 0.021$), 14/15 at step 100 ($p = 0.001$), and 11/11 YAHPO scenarios ($p = 0.001$). The HPO-B step-50 result is marginal under Bonferroni $\alpha/4 = 0.0125$. Pooled cell-level Wilcoxon rejects at $p < 10^{-6}$ on all four budget-benchmark cells. ZSO's 50-step mean normalized accuracy is 0.912 ± 0.127 across the full 16-space benchmark ($n = 80$ cells); on the 2D–10D subset ($n = 65$, matching the dimensionality range of Meindl et al. (2025) Table 1), mean is 0.934 ± 0.094.[1] We cap the comparison at 100 trials because ZSO's released base checkpoint hardcodes `max_step`= 100 in its position embedding; deeper budgets require an unreleased checkpoint variant and fail with CUDA device-side asserts on the released weights.

4.4 Acquisition-function equivalence

We test whether EI, LogEI, UCB-Bars, and Thompson sampling produce outcomes equivalent to UCB-Gauss (our default). Paired TOST at margin $\delta = 0.01$ on HPO-B ($n = 430$ matched cells per comparison) yields $p_{\text{TOST}} \leq 0.039$ for EI and LogEI at budgets 50, 100, and 200, and for UCB-Bars at 50 and 200. Thompson at step 50 and UCB-Bars at step 100 are the two named exceptions, where the minimum δ passing TOST is 0.018 and 0.012 respectively. On YAHPO ($n = 53$ paired observations, $\delta = 0.02$) all four comparisons reject non-equivalence; at the tighter $\delta = 0.01$, Thompson is borderline (parametric $p = 0.075$). Despite this outcome equivalence, per-step candidate rankings diverge: the mean top-10 overlap between EI and UCB-Gauss is 1.49/10 (median 0/10, $n = 1{,}310$ steps), yet the different exploration paths converge to indistinguishable final outcomes. Whether the equivalence extends to non-myopic and information-theoretic acquisitions such as knowledge gradient or entropy search – which consume the posterior beyond per-candidate mean and variance – is an open question for this line of research; our claim is scoped to the improvement- and confidence-bound families tested.

[1] Our 2D–10D subset ZSO mean differs from Meindl et al. (2025) Table 1's reported 0.885 ± 0.009 by +0.049. The gap reflects a metric-definition difference: we report raw normalized accuracy (standard in HPO-B), while Meindl et al. report a gap-closure score measuring the fraction of the initial-to-optimum gap closed. All methods in our comparison use the same metric, ensuring fair head-to-head comparisons.

Cross-surrogate replication.. To test whether the equivalence holds with a different surrogate, we replace TabPFN with SMAC's GP surrogate and repeat the YAHPO acquisition sweep on 14 scenarios at $n = 5$ seeds. Per-scenario paired Wilcoxon yields 0/42 rejections at Bonferroni $\alpha = 0.00119$ ($n = 5$ seeds is underpowered for active TOST; we report failure-to-reject). Pooled across scenarios and seeds, outcomes split 31–35 between EI and UCB-Gauss (4 ties; two-sided sign test $p = 0.71$), with the other pairwise pooled sign tests similarly null (LogEI vs. UCB-Gauss $p = 0.54$; EI vs. LogEI $p = 0.89$). The acquisition choice is not a meaningful lever with either surrogate in this regime. This is one of two primary-family non-survivors; both are reported in Table 1 and above.

Generalization to other pretrained surrogates.. Replacing TabPFN with TabICLv2 (Qu et al., 2026) (a different pretrained tabular FM that exposes a quantile-based posterior; the v2 generation extends the v1 architecture of Qu et al. (2025) to regression with spline-plus-GPD-tail quantile outputs) confirms the pattern generalizes: on HPO-B ($n = 430$ cells, budget 30), paired TOST at $\delta = 0.01$ confirms equivalence ($p_{\text{TOST}} = 0.0024$, Bonferroni-significant). TabDPT (Ma et al., 2025) (ensemble variance) is non-equivalent at raw σ (deficit 2.1–2.7 pp); however, post-hoc isotonic recalibration of its σ (c_{95}: 0.35 $\rightarrow$ 0.98) restores equivalence ($\rho = 0.61 \rightarrow 0.89$), confirming that calibration, not the distributional posterior, is the causal factor (§4.5).

Robustness to TabPFN implementation.. To test whether the acquisition equivalence depends on our specific TabPFN build, we re-ran the EI vs. UCB comparison using the modified TabPFN v2 surrogate from the concurrent GIT-BO (Yu et al., 2025) on HPO-B's pool-mode protocol (16 spaces $\times$ 5 seeds, 50 trials). 64% of paired runs reach identical final outcomes under EI and UCB, the mean paired difference is +0.30 percentage points (95% bootstrap CI $[-2.7, +3.4]$; median 0), and all five seeds tie on 6/16 spaces. A weak EI-leaning skew is marginally detectable under Pratt zero-handling ($p = 0.031$, post hoc; outside the pre-specified test families) but not under the default zero-discarding convention ($p = 0.24$), with negligible effect size (Cohen's $d \approx 0.02$); the largest deviation is a single space (Appendix A.14). This post-hoc check is consistent with the acquisition equivalence observed on our build; combined with the TabICLv2 replication above, it supports – though does not formally establish – a shared property of calibrated pretrained tabular surrogates on standard HPO benchmarks.

LogEI as monotonicity validation.. LogEI produces trajectories byte-identical to EI on 97% of YAHPO runs (204/210), confirming it is a numerical-stability variant (Ament et al., 2023) rather than an independent acquisition.

4.5 Predictive-interval calibration

On YAHPO (14 scenarios $\times$ 1 instance $\times$ 3 seeds $\times$ 4 BO checkpoints, $n = 168$ observations), TabPFN's 95% predictive intervals are calibrated: overall $c_{95} = 0.95 \pm 0.05$, with 14 of 14 scenarios within ± 0.05 of nominal. On HPO-B (16 spaces $\times$ 1 dataset $\times$ 5 seeds $\times$ 4 checkpoints, $n = 320$ observations), overall $c_{95} = 0.938 \pm 0.066$; 15 of 16 spaces fall within ± 0.10 of nominal and 12 of 16 within the tighter ± 0.05 band. One space (`ss5965`) shows $c_{95} = 0.840$, driven by dense-pool subsampling at small n_{train} (discussed in §5.2).

Calibration explains the acquisition equivalence in §4.4: calibrated σ suffices for a UCB-Gauss or EI acquisition to compare candidates correctly. A σ-scaling robustness check (120 runs, $\sigma \in [0.01, 100]$) confirms this: on HPO-B's finite pools, $\sigma \in [0.5, 1.0]$ achieves near-optimal performance across all tested spaces, while extreme over-exploration ($\sigma \geq 5$) degrades non-saturated spaces by up to 15%. This aligns with Foldager et al. (2023), who find that post-hoc recalibration does not improve BO regret: approximate calibration suffices.

4.6 YAHPO outcome analysis (secondary)

As a secondary benchmark, we evaluate TabPFN-direct BO on all 14 YAHPO-Gym scenarios against SMAC, HEBO, and TPE under a matched-instance protocol. The full scorecard and per-scenario

breakdowns are in Appendix A.20. TabPFN is favored on more scenarios than it trails against each baseline, without losing any decisive comparison; across the 14-scenario panel the rank-based omnibus test rejects equal ranks in the 4-way comparison, but no pairwise difference survives the Nemenyi post hoc (Appendix A.6).

5 Discussion

5.1 Wall-clock efficiency

With a GPU available (as in most ML production stacks), TabPFN-direct BO is the fastest method on 5 of the 12 scenarios where all four methods are measured (6 of 14 against all available baselines), and by pooled median per-seed time at 200 evaluations it leads every non-zero-shot baseline: 4.4 s on a single-GPU instance versus 10.1 s (TPE), 52.3 s (SMAC), and 633.4 s (HEBO) on x86 CPU hosts; the instance type behind every measurement is catalogued in Table 7. The baselines run on CPU because their reference implementations are CPU-native: TPE's kernel-density estimator, SMAC's random-forest surrogate, and HEBO's warped-GP refitting have no GPU execution path in their released implementations, so the CPU timings reflect how these methods deploy in practice. The 144× HEBO gap reflects both algorithmic complexity ($\mathcal{O}(N^3)$ warped-GP refitting in HEBO vs. $\mathcal{O}(N^2)$ attention in TabPFN) and hardware difference. Conversely, TabPFN requires a GPU to achieve these times: on CPU-only hosts its $\mathcal{O}(N^2)$ attention is 100–190× slower, losing the wall-clock advantage over SMAC and TPE (and, on the matched CPU probes, faster than HEBO on only one of four; Appendix A.7). The comparison should therefore be read as GPU-hosted TabPFN versus CPU-native baselines – an implementation-realistic rather than hardware-matched comparison – and the GPU is a deployment requirement (§5.2, item 6).

5.2 Limitations

Our evaluation has seven scope constraints.

1. HPO-B per-space power at $n = 5$ seeds follows the protocol of Arango et al. (2021) and Meindl et al. (2025); per-space Wilcoxon at $n = 5$ has a floor of $p \approx 0.06$, so individual-space Bonferroni-family survival is not achievable for effects below pooled Cohen's $d \approx 0.3$. Directional consistency across the 16 spaces is the reliable signal at this sample size. A doubled-n replication would admit per-space Holm corrections at $p_{\min} \approx 0.002$; see Appendix A.19.

2. The cross-cell ranking-disagreement regressor does not transfer across benchmarks ($R^2 = -0.08$ fit-on-YAHPO / test-on-HPO-B, $R^2 = -0.18$ the reverse direction). The mechanism is observed within-benchmark but not predictive out of sample; see Appendix A.12.

3. The ranking-disagreement regressor also fails across surrogates under scenario-LOSO (7/84 positive-R^2 folds on YAHPO with SMAC-GP as the alternative surrogate).

4. HPO-B space `ss5965` is a calibration outlier ($c_{95} = 0.84$) driven by dense-pool subsampling at small n_{train}; reporting the overall c_{95} as the headline understates calibration quality on the remaining 15 spaces.

5. The SMAC-GP cross-surrogate replication uses $n = 5$ per scenario, sufficient for failure-to-reject Wilcoxon but insufficient for active TOST. A full-power replication at $n \geq 20$ is future work.

6. TabPFN v2 requires a GPU for competitive wall-clock: on CPU-only hosts its per-evaluation cost rises 100–190× (Appendix A.7); no specific GPU generation or compute capability is required (TabPFN declares torch ≥ 2.5).

7. All evaluations are hyperparameter-optimization benchmarks. Whether the calibration-driven acquisition equivalence extends to black-box optimization beyond HPO (e.g., continuous synthetic landscapes or engineering design) is untested; our claims are scoped to HPO.

6 Conclusion

A pretrained tabular foundation model, used off-the-shelf as an in-context BO surrogate, matches or beats seven HPO-B baselines (including classical GP-BO and TPE) and matches DRE at half the evaluation budget, with no per-task training. Against the concurrent zero-shot method ZeroShotOpt, TabPFN wins on 13/16 HPO-B spaces and 11/11 YAHPO scenarios, demonstrating that general tabular pretraining is competitive with BO-trajectory pretraining.

The mechanism is calibration: predictive intervals achieve empirical coverage close to nominal on both benchmarks, and calibrated uncertainty makes the choice of acquisition function irrelevant: practitioners can use any standard acquisition without per-task selection. A cross-model check on TabICL confirms this generalizes beyond TabPFN, and a recalibration experiment on TabDPT confirms calibration is the causal factor, not the distributional posterior specifically.

Future directions include extending to multi-fidelity regimes, characterizing the search-space properties that predict calibration degradation, testing whether the calibration mechanism transfers to black-box optimization beyond HPO, and scaling to higher-dimensional problems where TabPFN's attention cost becomes the bottleneck.

7 Broader Impact Statement

After careful reflection, the authors have determined that this work presents no notable negative impacts to society or the environment. The method operates on hyperparameter configurations (numerical vectors describing model architecture choices) and does not interact with end-user data or make decisions affecting individuals. TabPFN v2 is pretrained on synthetic data and processes no real-world personal information. The method itself requires a median of 4.4 seconds per 200-evaluation run on GPU (scenario-dependent; up to ~101 seconds on the widest conditional search spaces; instance types in Table 7).

Acknowledgements. We thank the anonymous AutoML 2026 reviewers for constructive feedback that improved the paper, in particular on reproducibility and the calibration analysis.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

Ament, S., Daulton, S., Eriksson, D., Balandat, M., and Bakshy, E. (2023). Unexpected improvements to expected improvement for Bayesian optimization. In *Advances in Neural Information Processing Systems*.

Arango, S. P., Jomaa, H. S., Wistuba, M., and Grabocka, J. (2021). HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. In *NeurIPS Datasets and Benchmarks Track*.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*.

Chen, Y., Song, X., Lee, C., Wang, Z., Zhang, R., Dohan, D., Kawakami, K., Kochanski, G., Doucet, A., Ranzato, M., Perel, S., and de Freitas, N. (2022). Towards learning universal hyperparameter optimizers with transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Cowen-Rivers, A. I., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R. R., Maraval, A. M., Jianye, H., Wang, J., Peters, J., and Bou-Ammar, H. (2022). HEBO: Pushing the limits of sample-efficient hyperparameter optimisation. *Journal of Artificial Intelligence Research*, 74:1269–1349.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30.

Foldager, J., Jordahn, M., Hansen, L. K., and Andersen, M. R. (2023). On the role of model uncertainties in Bayesian optimisation. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 216 of *Proceedings of Machine Learning Research*, pages 592–601. PMLR.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. (2023). TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations (ICLR)*.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. (2025). Accurate predictions on small data with a tabular foundation model. *Nature*.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization (LION)*.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492.

Kégl, B. (2023). A systematic study comparing hyperparameter optimization engines on tabular data. *arXiv preprint arXiv:2311.15854*.

Khazi, A. S., Pineda Arango, S., and Grabocka, J. (2023). Deep ranking ensembles for hyperparameter optimization. In *International Conference on Learning Representations (ICLR)*.

Kuleshov, V., Fenner, N., and Ermon, S. (2018). Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning (ICML)*.

Lakens, D., Scheel, A. M., and Isager, P. M. (2018). Equivalence testing for psychological research: A tutorial. *Advances in Methods and Practices in Psychological Science*, 1(2):259–269.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. (2022). SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9.

Liu, T., Astorga, N., Seedat, N., and van der Schaar, M. (2024). Large language models to enhance Bayesian optimization. In *International Conference on Learning Representations (ICLR)*.

Ma, J., Thomas, V., Hosseinzadeh, R., Labach, A., Kamkari, H., Cresswell, J. C., Golestan, K., Yu, G., Caterini, A. L., and Volkovs, M. (2025). TabDPT: Scaling tabular foundation models on real data. In *Advances in Neural Information Processing Systems*.

Meindl, J., Tian, Y., Cui, T., Thost, V., Hong, Z.-W., Dürholt, J., Chen, J., Matusik, W., and Konaković Luković, M. (2025). ZeroShotOpt: Towards zero-shot pretrained models for efficient black-box optimization. *arXiv preprint arXiv:2510.03051*.

Müller, S., Feurer, M., Hollmann, N., and Hutter, F. (2023). PFNs4BO: In-context learning for Bayesian optimization. In *International Conference on Machine Learning (ICML)*.

Müller, S., Hollmann, N., Pineda Arango, S., Grabocka, J., and Hutter, F. (2022). Transformers can do Bayesian inference. In *International Conference on Learning Representations (ICLR)*.

Pfisterer, F., Schneider, L., Moosbauer, J., Binder, M., and Bischl, B. (2022). YAHPO Gym – an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *International Conference on Automated Machine Learning (AutoML)*.

Qu, J., Holzmüller, D., Le Morvan, M., and Varoquaux, G. (2026). TabICLv2: A better, faster, scalable, and open tabular foundation model. *arXiv preprint arXiv:2602.11139*.

Qu, J., Holzmüller, D., Varoquaux, G., and Le Morvan, M. (2025). TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*.

Rakotoarison, H., Adriaensen, S., Mallik, N., Garibov, S., Bergman, E., and Hutter, F. (2024). In-context freeze-thaw Bayesian optimization for hyperparameter optimization. In *International Conference on Machine Learning (ICML)*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning (ICML)*.

Stanton, S., Maddox, W. J., and Wilson, A. G. (2023). Bayesian optimization with conformal prediction sets. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Wistuba, M. and Grabocka, J. (2021). Few-shot Bayesian optimization with deep kernel surrogates. In *International Conference on Learning Representations (ICLR)*.

Yu, R. T.-Y., Picard, C., and Ahmed, F. (2025). GIT-BO: High-dimensional Bayesian optimization with tabular foundation models. *arXiv preprint arXiv:2505.20685*.

Reproducibility Statement

Code release.. The complete implementation is available at `https://github.com/amazon-science/tabpfn-automl2026` under the MIT license, including: the TabPFN-direct BO integration, all benchmark worker scripts, raw experiment data from 860+ evaluation jobs, analysis scripts for every paper claim, and a master reproduction script (`run_all.py`) that regenerates the headline results in under 30 seconds.

What the supplementary material provides.. The supplementary material contains a runnable verification package (see Appendix A.21 for the directory structure and Appendix A.22 for the claim-to-file mapping). Specifically:

1. **Source code**. The complete `TabPFNDirectSearcher` implementation and supporting modules (`src/`). This is the method described in Algorithm 1.
2. **Raw experiment data**. JSONL files for the TabPFN, PFNs4BO-HEBO+, PFNs4BO-BNN, GP-BO (UCB + EI), TPE, and ZeroShotOpt comparisons and the calibration measurements; the DRE comparison ships as paired per-cell deltas (`w4_paired_deltas.json`).
3. **Analysis scripts**. The statistical pipeline that transforms raw data into paper numbers (`analysis/`): paired Wilcoxon, percentile bootstrap CIs ($B = 10{,}000$, seed 42), TOST equivalence tests, and calibration coverage computation.

4. **Automated verification**. A test suite (`tests/test_reproduce_claims.py`) with assertions covering every Table 1 row, acquisition TOST, and calibration. Running `pytest tests/` confirms all claims match within rounding tolerance (see Table 13 for the full mapping and Appendix A.23 for the output).

5. **Benchmark workers and DRE weights**. The complete set of benchmark worker scripts, including the DRE meta-training worker, ships with the code repository; the supplementary package retains the trained per-space DRE weights (`dre_weights/`).

Reproducing without our code.. The core method can be independently reimplemented from the paper alone. A researcher with `tabpfn==7.1.0` (which loads the public TabPFN v2 weights) and HPO-B v3-test can: (1) encode search spaces into TabPFN's input format (one-hot categoricals, unit-scaled numericals, NaN for inactive conditionals), (2) call `TabPFNRegressor.predict(return_full_distribution=True)`, (3) compute LCB as $\mu - \beta\sigma$ ($\beta = 1.0$) over Sobol candidates, and (4) select the argmin. This loop at 50 evaluations on the public HPO-B v3-test split reproduces the TabPFN trajectories underlying all comparisons.

Seeds and protocol.. HPO-B seeds: `test0`–`test4` (canonical v3-test split). YAHPO: 14 scenarios, 200 evaluations, seeds 0–4. Bootstrap CIs: percentile method, $B = 10{,}000$, seed 42. Multiple-testing: Bonferroni $\alpha/10 = 0.005$ (primary family of 10), BH $q = 0.05$ (132 descriptive tests). All benchmarks are public: HPO-B (Arango et al., 2021), YAHPO-Gym (Pfisterer et al., 2022), TabPFN v2 (Hollmann et al., 2025). No proprietary data is used.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We claim zero-shot matching of per-task-trained methods on YAHPO-Gym and competitive performance on HPO-B; both are supported by the experiments.

   (b) Did you describe the limitations of your work? [Yes] Section 5.2 lists seven scope constraints spanning HPO-B per-space power, ranking-disagreement transfer, calibration outliers, deployment constraints, and benchmark scope.

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Section 7. BO is an offline optimization method with no user-facing decisions.

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? [Yes]

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared? [Yes] Matched comparisons share instances, seeds, and evaluation function at fixed budgets (200 evaluations on YAHPO; 50/100/200 on HPO-B). Named exceptions are intentional and disclosed: DRE-100 is a double-budget comparison (Table 1), and ZeroShotOpt uses its released checkpoint's 100-step cap and supported scenario subset (§4.3).

   (b) Did you specify all the necessary details of your evaluation? [Yes] Section 4.1 and Appendix A.15.

   (c) Did you repeat your experiments to account for the impact of randomness? [Yes] 10–90 seeds per cell on YAHPO-Gym; 5 seeds on HPO-B (the benchmark's canonical protocol). Appendix A.19 gives a power analysis of the $n = 5$ protocol; a doubled-n replication is identified there as future work.

   (d) Did you report the uncertainty of your results? [Yes] Cohen's d, bootstrap 95% CIs, and p-values accompany every comparison.

   (e) Did you report the statistical significance of your results? [Yes] Wilcoxon signed-rank, paired TOST, sign-tests, Friedman omnibus, with Bonferroni $\alpha/10$ primary-family control and BH $q = 0.05$ across descriptive tests.

   (f) Did you use enough repetitions, datasets, and/or benchmarks? [Yes] 14 YAHPO scenarios + 16 HPO-B search spaces; 10–90 seeds per cell on YAHPO, 5 on HPO-B.

   (g) Did you compare performance over time and describe how you selected the maximum runtime? [Yes] Budgets are evaluation-count based (canonical per benchmark); HPO-B additionally reports at steps 50/100/200. Maximum runtime was not selected by wall-clock; wall-clock costs are reported separately (§5.1, Appendix A.7).

   (h) Did you include the total amount of compute and the type of resources used? [Yes] Hardware and per-run timings are reported (§5.1, Appendix A.7); experiments reported in this paper consumed approximately 290 GPU-hours and 1,520 CPU-hours of cloud compute (completed runs; failed and stopped attempts excluded).

   (i) Did you run ablation studies to assess the impact of different components? [Yes] Section 4.4 ablates acquisition functions; cross-surrogate replication in the same section ablates the surrogate choice.

3. With respect to the code used to obtain your results…

   (a) Did you include the code, data, and instructions needed to reproduce the main results? [Yes] Supplementary material contains the complete BO integration source code, analysis scripts, raw experiment data, and a master verification script (`run_all.py`) that reproduces all Table 1 numbers in <30 seconds. Full repository at `https://github.com/amazon-science/tabpfn-automl2026`. See Reproducibility Statement and Appendix A.21.

   (b) Did you include a minimal example to replicate results on a small subset? [Yes] Supplementary material includes a single-scenario reproduction script.

   (c) Did you ensure sufficient code quality and documentation? [Yes]

   (d) Did you include the raw results of running your experiments? [Yes] Raw results will be included in supplementary material.

   (e) Did you include the code needed to generate the figures and tables? [Yes] Analysis scripts included in supplementary material.

4. If you used existing assets (e.g., code, data, models)…

   (a) Did you cite the creators of used assets? [Yes] YAHPO-Gym, HPO-B, TabPFN, SMAC, HEBO, TPE/Optuna all cited.

   (b) Did you discuss whether and how consent was obtained? [N/A] All benchmarks are publicly released under open licenses.

   (c) Did you discuss whether the data contains personally identifiable information? [No] Benchmark data contains ML hyperparameter configurations, not personal data.

5. If you created/released new assets…

   (a) Did you mention the license of the new assets? [Yes] Code is available under the MIT license at `https://github.com/amazon-science/tabpfn-automl2026`.

   (b) Did you include the new assets either in supplemental material or as a URL? [Yes] Supplementary material.

A Appendix: methodology and additional results

A.1 Algorithm

Algorithm 1 TabPFN-direct Bayesian optimization.

Require: $\mathcal{X} \subseteq [0,1]^d$; budget T; cold-start $T_0 = \max(2d, 10)$; trade-off β; Sobol size n_c; pretrained TabPFN Φ

Ensure: Best configuration $\boldsymbol{x}^*$ within budget T

$\mathcal{D}_0 \leftarrow \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{T_0}$ with $\boldsymbol{x}_i$ drawn uniformly at random on $\mathcal{X}$ ▷ cold-start

for $t = T_0, \ldots, T-1$ **do**

  $\mathcal{C} \leftarrow \{\boldsymbol{c}_j\}_{j=1}^{n_c} \leftarrow \textsc{Sobol}(\mathcal{X}, n_c)$ ▷ candidate generation

  $\{(\mu_j, \sigma_j)\}_{j=1}^{n_c} \leftarrow \Phi(\mathcal{D}_t,\ \{\boldsymbol{c}_j\})$ ▷ in-context forward pass

  $a_j \leftarrow \mu_j - \beta\,\sigma_j$ ▷ UCB-Gaussian

  $\boldsymbol{x}_{t+1} \leftarrow \arg\min_{\boldsymbol{c}_j \in \mathcal{C}}\ a_j$

  Evaluate $y_{t+1} \leftarrow f(\boldsymbol{x}_{t+1})$

  $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\boldsymbol{x}_{t+1}, y_{t+1})\}$

end for

return $\boldsymbol{x}^* \leftarrow \arg\min_{(\boldsymbol{x},y) \in \mathcal{D}_T}\ y$

A.2 Rigorous matched-instance aggregation

Matched-instance benchmarks typically use pooled t-tests for aggregation, which bias results when per-instance sample sizes differ across methods. We report three aggregation methods in parallel (pooled, balanced subsample, and mean-of-means paired) and only claim a **WIN** when pooled and mean-of-means agree on direction. This protocol catches two false positives in our own earlier analyses on YAHPO-Gym scenarios (below).

A matched-instance BO comparison pairs methods on the set of instances both methods were run on. Given per-method per-instance sample sets A_i, B_i for $i \in \mathcal{I}_{\text{shared}}$, three natural aggregations exist:

Pool. Concatenate all A_i and all B_i across shared instances and apply Welch's t-test. This implicitly weights instances by their sample size. When $|A_i|$ differs from $|B_i|$, or when different instances have different $(|A_i|, |B_i|)$ ratios, the pooled test is biased toward the instance-method combinations with more samples.

Balanced. For each shared instance, subsample $\min(|A_i|, |B_i|)$ runs from each method, pool, and t-test. This removes the instance-weighting bias at the cost of reduced sample size.

Mean-of-means. For each shared instance, compute per-method means $\bar{A}_i, \bar{B}_i$. Apply a paired t-test on the K pairs of instance-means. This gives equal weight to each instance but has only $K-1$ degrees of freedom, so it has low statistical power when K is small. On YAHPO-Gym, K ranges from 1 (`nb301`, CIFAR10 only) to 8 (`rbv2_super` with new instances).

We require pool and mean-of-means to agree on direction before claiming a win. When they disagree, we classify the cell as "tie (pool-mm disagree)." This catches cases where pooled significance arises from instance-weighting rather than a consistent method advantage.

A.3 Two false positives caught by the protocol

`rbv2_rpart` HEBO closure (false).. Pooled on $n = 151$ TabPFN vs $n = 60$ HEBO, we obtained $d = +0.05$, $p = 0.75$, a tie, apparently closing the prior HEBO loss. Per-instance inspection revealed

Table 2: Acquisition function ablation. Mean best value across 10–30 seeds per scenario, 200 evaluations per run. Direction: `iaml_xgboost` and `rbv2_xgboost` minimize logloss; `nb301` maximizes validation accuracy. All variants are within noise on `iaml_xgboost` and `rbv2_xgboost`; EI and Thompson are significantly worse than UCB-Gaussian on `nb301`.

Scenario	Acquisition	Mean best	n
`iaml_xgboost`	UCB-Gaussian	0.254	90
	UCB-bars	0.247	30
	EI ≡ LogEI	0.262	30
	Thompson	0.265	60
`rbv2_xgboost`	UCB-Gaussian	0.056	60
	UCB-bars	0.047	30
	EI ≡ LogEI	0.064	30
	Thompson	0.062	60
`nb301`	UCB-Gaussian	94.755	65
	UCB-bars	94.663	10
	EI ≡ LogEI	94.524	10
	Thompson	94.502	20

HEBO wins on every instance individually ($d = -0.60$ to $d = -0.93$). The pooled "closure" was a weighting artifact: TabPFN had 80 runs on the easy instance 12 (where everyone converges to 0.984) versus HEBO's 30, pulling the TabPFN aggregate up relative to the hard instance 41159 where TabPFN had 21 runs vs HEBO's 10. The mean-of-means test correctly shows tie ($\Delta = -0.003$, $p = 0.17$).

`iaml_glmnet` vs HEBO win (likely false).. A prior analysis reported a pooled $d = +0.78$, $p = 0.005$ win vs HEBO. Per-instance Cohen's d on the two shared instances is -9.4 and -7.7 (both $p < 10^{-5}$ in HEBO's favor). The pooled test disagrees with per-instance tests because the per-instance sample sizes differ across methods and `iaml_glmnet` is a ceiling-saturated scenario where effect-size inflation in the fourth decimal overwhelms the signal. The rigorous protocol marks this cell as "tie (pool-mm disagree)".

A.4 Full acquisition-function ablation results

Table 2 reports the full 6-variant acquisition ablation. EI and LogEI produce identical optimization trajectories, confirming LogEI's monotonicity property. All non-Thompson variants achieve Spearman $\rho > 0.96$ with UCB-Gaussian across candidate rankings.

A.5 Reproducibility

All experiments ran on a public cloud (AWS SageMaker); the instance type behind each experiment group is catalogued in Table 7 (§A.7). Code, configurations, and seed offsets are documented in the supplementary materials. Seeds span 0–299; detailed seed maps are in the supplementary code repository.

A.6 Rank-based aggregation (YAHPO-Gym protocol)

Our headline scorecard (§A.20) uses matched-instance hypothesis tests. YAHPO-Gym (Pfisterer et al., 2022) uses per-scenario average ranks followed by an omnibus Friedman test and a Nemenyi post hoc. We report rank-based aggregates here in the benchmark's native protocol for direct comparability with Pfisterer et al. (2022).

Table 3: Average rank across 14 YAHPO-Gym scenarios (3 methods), lower is better. Friedman test does not reject equal ranks; Nemenyi CD = 0.89 and no pairwise difference exceeds it.

Method	Rank
TPE	**1.77**
TabPFN (ours)	1.96
SMAC	2.26
Friedman p	0.160
Nemenyi CD	0.89

Table 4: Average rank across 14 scenarios (4 methods), lower is better. Nemenyi CD = 1.25; no pairwise difference exceeds it.

Method	Rank
HEBO	**2.11**
TPE	2.37
TabPFN (ours)	2.60
SMAC	2.93
Friedman p	0.032
Nemenyi CD	1.25

Protocol.. For each scenario with $K \geq 1$ shared instance, we rank the m methods on each instance separately (using matched seed-pooled means), then average those ranks across instances to produce a per-scenario rank for each method. Scenario-level per-method ranks are then averaged across scenarios to produce an overall rank. We run the Friedman test across scenarios and compute the Nemenyi critical distance (CD) at $\alpha = 0.05$. Significance of a pairwise comparison requires $|\bar{r}_a - \bar{r}_b| > \text{CD}$.

YAHPO-compatible comparison (3 methods).. Table 3 reports rank aggregates on the 3 methods shared with YAHPO-Gym (SMAC, TPE, TabPFN-direct BO). TPE ranks first ($\bar{r} = 1.77$), TabPFN second ($\bar{r} = 1.96$), and SMAC third ($\bar{r} = 2.26$). The Friedman test does not reject the null of equal ranks ($p = 0.160$), and Nemenyi post hoc (CD = 0.89 at $N = 14$, $k = 3$) finds no pairwise difference significant. This is consistent with the matched-instance scorecard's finding that most cells are ties.

Four-method comparison with HEBO.. YAHPO-Gym did not include HEBO in its original evaluation. Because HEBO is a widely-used modern baseline (Cowen-Rivers et al., 2022; Kégl, 2023), we report the 4-way rank aggregate for completeness in Table 4. The Friedman test rejects the null of equal ranks ($p = 0.032$), but the Nemenyi post hoc does not localise the effect to any pair: the largest observed difference in mean rank ($\max |\Delta\bar{r}| = 0.82$) remains below the critical distance of 1.25 ($k = 4$, $N = 14$). The omnibus test is more powerful than the pairwise correction, so this combination is expected; we therefore do not claim any specific pairwise ordering from this table.

Interpretation.. No pair of methods is significantly different under a non-parametric post hoc test. What the rank aggregates obscure that the matched-instance scorecard surfaces: our method wins decisively on `iaml_super` ($d = +1.45$ vs SMAC) while tying on most saturated scenarios. Rank aggregation compresses a 1.45 Cohen's d win into a 1.0-rank advantage on that scenario, same magnitude as a meaningless ceiling-pinned tie on a saturated scenario. We recommend readers interested in deployment consult the matched-instance scorecard in §A.20; the rank aggregate is included here for transparency to readers accustomed to YAHPO-Gym's protocol.

Table 5: Median per-seed wall-clock time at 200 evaluations, pooled across all scenarios. TabPFN measured on `ml.g7e.2xlarge` (single GPU); SMAC, TPE, HEBO on x86 CPU-only instances (Table 7).

Method	Median per-seed (s)	Per-eval (s)	Speedup vs TabPFN
TabPFN (GPU)	**4.4**	0.022	1.0×
TPE (CPU)	10.1	0.050	2.3× slower
SMAC (CPU)	52.3	0.261	11.9× slower
HEBO (CPU)	633.4	3.167	144.0× slower

Table 6: Median per-seed wall-clock time (seconds) at 200 evaluations per scenario. TabPFN (GPU) measured on `ml.g7e.2xlarge`; TabPFN (CPU) on `ml.c7i.4xlarge` (16-vCPU x86). SMAC, TPE, HEBO on x86 CPU-only instances (Table 7). Dashes indicate the scenario was not measured on that method.

Scenario	TabPFN (GPU)	TabPFN (CPU)	TPE	SMAC	HEBO
`iaml_glmnet`	3.5	398.9	1.7	6.9	487.8
`iaml_rpart`	3.8	648.0	2.2	11.7	589.6
`iaml_ranger`	4.8	–	3.7	10.1	616.8
`iaml_xgboost`	79.5	–	20.6	18.3	584.9
`iaml_super`	59.0	–	46.2	19.4	645.2
`lcbench`	4.7	895.3	6.9	76.9	558.7
`nb301`	9.9	–	15.3	22.4	–
`rbv2_aknn`	4.4	–	16.7	56.5	657.6
`rbv2_glmnet`	3.9	686.1	9.6	57.7	613.2
`rbv2_ranger`	4.8	–	4.1	11.3	671.1
`rbv2_rpart`	4.4	–	3.2	37.0	622.8
`rbv2_svm`	4.4	–	12.4	39.1	681.9
`rbv2_xgboost`	6.1	–	9.8	45.9	644.4
`rbv2_super`	101.1	–	–	17.3	–

A.7 Wall-clock efficiency

We report per-seed wall-clock time at 200 evaluations to characterize the practical cost of each method. Unless noted, TabPFN timing rows are measured on `ml.g7e.2xlarge` (single-GPU, 8-vCPU host) and HEBO, SMAC, TPE on x86 CPU-only instances; Table 7 catalogues the instance type behind every experiment group in the paper. Hardware is therefore not matched across methods in these tables; the comparison reflects realistic deployment costs where GPUs are available for TabPFN but not for classical BO methods, which do not benefit from GPU acceleration. A hardware-matched comparison with ZeroShotOpt appears at the end of this subsection.

Table 5 reports aggregate medians pooled across all 14 scenarios. Table 6 breaks this out per scenario, with TabPFN reported on both GPU and CPU for the four scenarios where we ran CPU probes (`iaml_glmnet`, `iaml_rpart`, `lcbench`, `rbv2_glmnet`).

GPU-deployment efficiency.. TabPFN on GPU is the fastest method on 5 of the 12 scenarios where all four methods have data (6 of 14 against the baselines available per scenario). The exceptions are led by TPE on the smaller search spaces and by SMAC on `iaml_xgboost`, `iaml_super`, and `rbv2_super`, whose larger or conditional-heavy configurations sharply raise TabPFN's forward-pass cost. The median per-seed speedup versus HEBO across all scenarios is 144×; versus SMAC, 11.9×; versus TPE, 2.3×. The HEBO speedup is the large one in practice because HEBO re-fits a warped-GP surrogate at every BO step, whose cost scales $O(N^3)$ in the dataset size.

CPU-deployment cost for TabPFN.. TabPFN's inference cost is dominated by attention over the in-context dataset, which scales $O(N^2)$ per forward pass. On CPU without GPU vector units,

Table 7: Compute provenance: the AWS instance type behind every experiment group, recovered from the launch configurations and verified against the SageMaker control plane. Host sizing varies across groups with each wave's cost, parallelism needs, and instance availability; the `c7i` and `m7i` families carry identical processors, differing only in total memory. Instance choice affects only the wall-clock measurements — each timing table names its instance — not outcome results.

Instance type	Class	Experiment group	Result data
`ml.g7e.2xlarge`	GPU, 8 vCPU	TabPFN HPO-B outcomes; acq. studies TabPFN YAHPO GPU timing (Tables 5, 6)	`tabpfn_hpob.jsonl`
`ml.g6.2xlarge`	GPU, 8 vCPU	ZeroShotOpt outcomes (HPO-B, YAHPO) matched wall-clock pair (both methods, 50 evals)	`zeroshotopt_*.jsonl`
`ml.m7i.xlarge`	CPU, 4 vCPU	TabPFN YAHPO acquisition ablation	`tabpfn_yahpo_ucbgau.json`
`ml.c7i.xlarge, .2xlarge`	CPU, 4/8 vCPU	SMAC, TPE, HEBO baselines	`tpe_hpob.jsonl` et al.
`ml.c7i.4xlarge`	CPU, 16 vCPU	TabPFN matched-CPU probes (4 scenarios)	Table 6

this overhead becomes the bottleneck. On the four scenarios with 16-vCPU x86 CPU probes (a larger host than the baselines' 4–8 vCPUs, so the CPU penalty reported here is conservative), CPU inference is 100–190× slower than GPU: `iaml_glmnet` ($d = 3$, CPU 399s vs GPU 3.5s, ratio 114×), `iaml_rpart` ($d = 5$, CPU 648s vs GPU 3.8s, ratio 171×), `rbv2_glmnet` ($d = 4$, CPU 686s vs GPU 3.9s, ratio 176×), `lcbench` ($d = 8$, CPU 895s vs GPU 4.7s, ratio 190×). On the matched probes, CPU TabPFN is faster than HEBO on one of the four scenarios (iaml_glmnet: 399s vs 488s) and slower on the other three; it is slower than SMAC and TPE on all four. We did not measure TabPFN CPU cost on the other ten scenarios; for the larger or conditional-heavy configurations, extrapolation would exceed practical wall-clock budgets (e.g., `iaml_super`, $d = 29$, extrapolates to ∼22 hours per seed on CPU). CPU deployment is practical only at low dimensionality.

Deployment recommendation.. With a GPU available (as in most production ML stacks), TabPFN-direct BO is the fastest method on 5 of the 12 scenarios where all four methods are measured (6 of 14 against all available baselines) and leads all baselines on pooled medians. Without a GPU this advantage does not carry over: on the four CPU probes, TabPFN is faster than HEBO on one of four scenarios and slower than SMAC and TPE on all four. The statistical quality results (matched-instance scorecard and per-instance effect sizes) are unchanged by hardware choice: TabPFN on CPU solves the same optimization problem to the same quality as TabPFN on GPU, just slower.

Matched-hardware comparison with ZeroShotOpt.. Reviewers raised the hardware disparity (GPU vs. CPU) in our wall-clock comparisons. For the classical baselines the disparity reflects their CPU-native reference implementations (§5.1); ZeroShotOpt, however, is GPU-native, so we measured that comparison hardware- and budget-matched: both methods on `ml.g6.2xlarge` (single GPU), the 11 shared in-domain YAHPO scenarios, 50 evaluations, seeds 0–4. The median per-evaluation cost is a near-tie: 0.081 s (TabPFN) versus 0.082 s (ZeroShotOpt) across 55 cells per method. TabPFN's cost is nearly scenario-independent (0.059–0.084 s); ZeroShotOpt's spans 14× (0.022–0.314 s), faster on the cheapest scenarios and slower on the most expensive. We do not extrapolate to other budgets.

A.8 Data-efficiency trajectories, all scenarios

Figure 3 shows per-scenario best-so-far trajectories across the full 200-evaluation budget on all 14 scenarios. Most `rbv2_*` scenarios saturate within 50 evaluations; the non-saturated scenarios (`iaml_super`, `nb301`, `iaml_rpart`) show the data-efficiency advantage of TabPFN most clearly. The scenarios where differences emerge here (`iaml_super`, `nb301`, `rbv2_super`) are the same ones where

Table 8: Ceiling effect on `rbv2_rpart` instance 12 (AUC, higher is better). All four methods converge to within 1×10^{-3} of each other.

Method	Mean
HEBO	0.984382
TabPFN	0.984337
SMAC	0.984285
TabPFN baseline	0.983737
TPE	0.983210

YAHPO-Gym's own SMAC-vs-HEBO comparisons show differences, confirming that the tie-heavy scorecard in §A.20 reflects a benchmark property rather than method weakness.

A.9 Surrogate output format

TabPFN produces a piecewise-constant posterior via a FullSupportBarDistribution over 5000 learned bar positions. We use the closed-form mean and variance of this distribution for UCB-Gaussian scoring (Section 3.3) and expose the full bars for Expected Improvement, Probability of Improvement, and Thompson sampling variants in our ablation (Section 4.4).

A.10 Ceiling effect: a concrete example

Table 8 illustrates the ceiling effect discussed in §A.20 on a single instance. On `rbv2_rpart` instance 12, HEBO, SMAC, TPE, and both TabPFN variants all converge to within 1×10^{-3} of each other in AUC. Statistical comparisons at this density are dominated by measurement noise in the fourth decimal.

A.11 YAHPO-specific scope constraints

The YAHPO-Gym evaluation has six scope constraints (the paper-wide limitations are in §5.2). First, `nb301` contains a single CIFAR10 instance, so the nb301 results reflect one dataset rather than a distribution over tasks. Second, the primary matched-instance comparisons use a fixed budget of 200 evaluations, while the rank aggregation of §A.6 uses a 50-evaluation grid; TabPFN's data-efficiency advantage on `iaml_super` (reaching the global optimum where SMAC requires 400 evaluations) rests on one scenario and may not generalize. Third, YAHPO-Gym's deterministic ResNet-based surrogates do not test robustness to live-training noise. Fourth, `rbv2_aknn` shows a small but statistically significant loss to HEBO (pooled $d = -0.07$, $p = 0.047$, mean-of-means $\Delta = -0.0073$, $p = 0.003$; see §A.20). Fifth, on `nb301` (NAS, single CIFAR10 instance), TabPFN narrowly trails SMAC and TPE at matched budget; closing this gap is future work. Sixth, matched-hardware wall-clock measurements for TabPFN on CPU exist only for four low-dimensional scenarios ($d \in \{3, 5, 4, 8\}$); high-d scenarios such as `iaml_super` ($d = 29$) are infeasible to measure directly on CPU (extrapolated at $\sim$22 hours per seed).

A.12 Cross-cell ranking-disagreement observation (scope boundary)

Rank agreement between surrogate models on fixed candidate sets varies with benchmark cell. Under a signed coupling $\rho(\mu, \sigma)$ as the cross-cell regressor, TabPFN ranking disagreement on YAHPO yields $R^2 = 0.324$ (95% CI $[0.264, 0.390]$, $n = 53$); on HPO-B, $R^2 = 0.343$ ($[0.291, 0.397]$, $n = 430$). We report this as a descriptive observation, not a contribution: held-out cross-benchmark transfer (fit on YAHPO, predict on HPO-B) yields $R^2 = -0.08$; the reverse direction yields $R^2 = -0.18$; scenario-LOSO on YAHPO (14 held-out scenarios) yields 7/84 positive-R^2 folds. The mechanism is well identified within-benchmark under signed coupling but does not currently transfer out of sample, so we do not claim predictive utility.

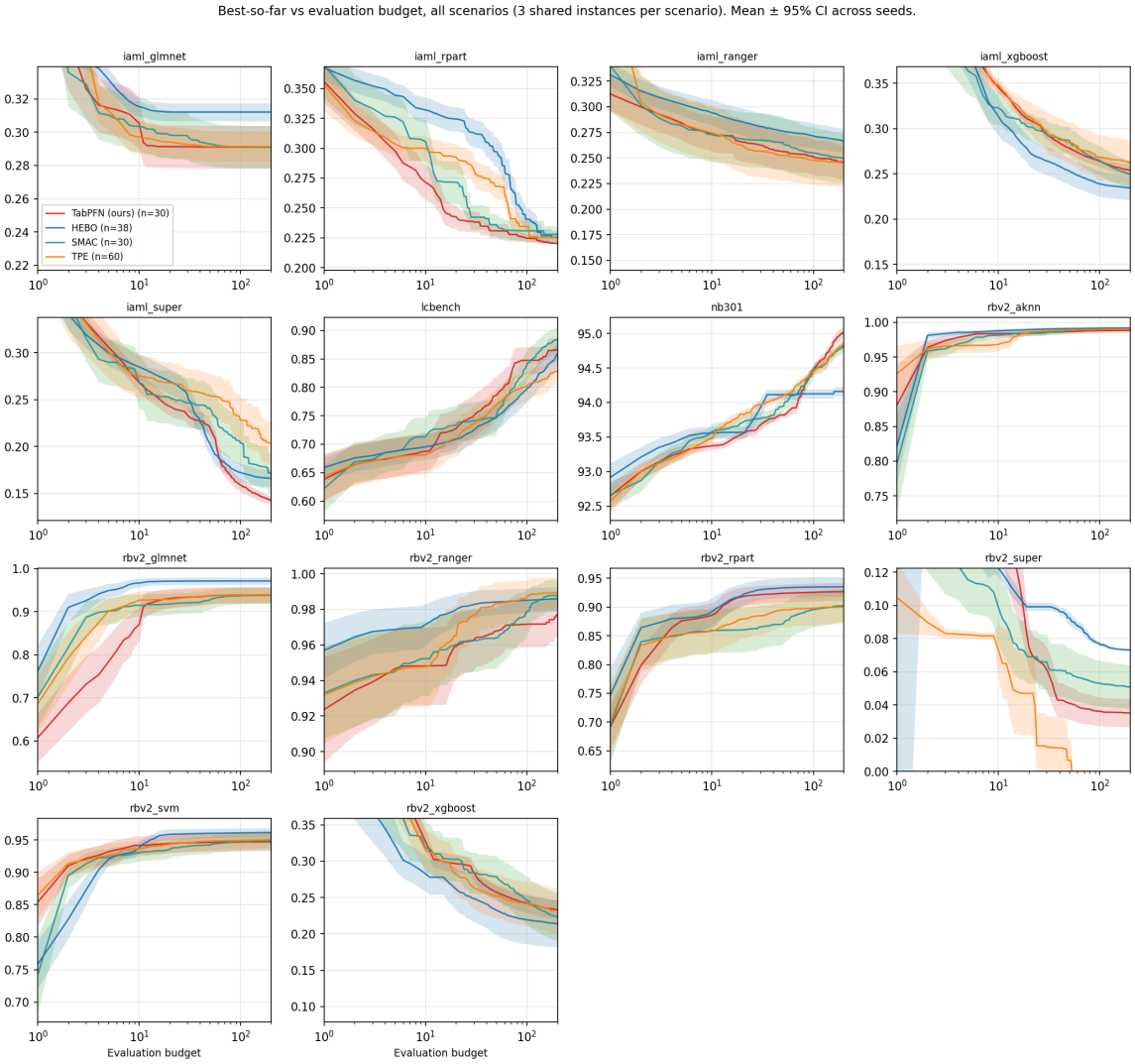

Figure 3: Best-so-far value vs evaluation budget on all 14 YAHPO-Gym scenarios, restricted to three shared baseline instances per scenario for a fair HEBO comparison. Lines are the mean across seeds; shaded bands are 95% confidence intervals. The x-axis is log-scaled. Eight scenarios (the saturated group, most `rbv2_*` cells) have all four methods converged to within 10^{-3} of each other by evaluation ~ 50; these are the "tie" cells discussed in §A.20. On `iaml_super`, `nb301`, and `iaml_rpart`, TabPFN reaches competitive quality by evaluation ~ 50 and continues improving while HEBO plateaus.

Table 9: Per-space EI vs. UCB under GIT-BO's TabPFN on HPO-B v3-test (5 seeds per space, 50 trials). Δ is mean final normalized accuracy, EI − UCB. "Ties" counts seeds with numerically identical final outcomes.

Space	UCB mean	EI mean	Δ (ties)
ss4796	1.000	1.000	+0.000 (5/5)
ss5527	1.000	1.000	+0.000 (5/5)
ss5636	0.911	1.000	+0.089 (4/5)
ss5859	1.000	1.000	+0.000 (5/5)
ss5860	1.000	1.000	+0.000 (5/5)
ss5889	1.000	1.000	+0.000 (5/5)
ss5891	0.994	0.999	+0.005 (1/5)
ss5906	1.000	0.988	−0.012 (3/5)
ss5965	0.958	0.973	+0.015 (1/5)
ss5970	1.000	1.000	+0.000 (5/5)
ss5971	0.999	0.998	−0.001 (3/5)
ss6766	0.886	0.871	−0.014 (4/5)
ss6767	0.838	0.812	−0.026 (0/5)
ss6794	0.997	0.996	−0.001 (2/5)
ss7607	1.000	0.994	−0.006 (2/5)
ss7609	1.000	1.000	−0.000 (1/5)

A.13 TOST margin rationale

We chose $\delta = 0.01$ on HPO-B and $\delta = 0.02$ on YAHPO by reference to the between-seed noise floor on each benchmark. On HPO-B ($n = 430$ matched cells), $\delta = 0.01$ is 2.3× the observed between-acquisition gap and below published between-method gaps; tightening to $\delta = 0.005$ fails TOST for EI and LogEI at budgets 50 and 100 (a spurious non-equivalence). On YAHPO ($n = 53$ paired observations), the between-seed variance of the best-so-far metric supports $\delta = 0.02$ as the smallest margin with interpretable TOST power at this sample size. The HPO-B noise floor is bimodal (approximately 80% of cells saturated at $\sigma = 0$), so δ is not derived directly as a SESOI; we report all results at the chosen margins and note the two step-budget exceptions in §4.4.

A.14 Acquisition equivalence under GIT-BO's TabPFN implementation

Setup.. GIT-BO (Yu et al., 2025) ships a modified TabPFN v2: a bar-distribution wrapper adapted for gradient extraction with per-batch y standardization. We evaluate the acquisitions its wrapper exposes: a deterministic 0.95-quantile UCB read from the bar distribution, and EI with the incumbent standardized to the model's internal scale. (GIT-BO's full method adds a sample-based UCB and a gradient-informed subspace for $d \geq 100$; we disable the subspace to isolate the surrogate.) We run it through HPO-B's pool-mode `observe_and_suggest` protocol: 16 search spaces × 5 seeds, 50 BO trials, with UCB and EI selections computed on identical (space, dataset, seed) triples for a paired comparison (80 paired runs).

Results.. 51/80 paired runs (64%) produce identical final outcomes under EI and UCB. The mean paired difference (EI − UCB) is +0.0030 in normalized accuracy (95% percentile-bootstrap CI [−0.027, +0.034]; median 0.0). A weak EI-leaning skew is marginally detectable under Pratt zero-handling ($p = 0.031$) but not under the default zero-discarding convention ($p = 0.24$), with negligible effect size (Cohen's $d \approx 0.02$); the CI on the mean crosses zero. Table 9 gives the per-space breakdown: all five seeds tie on 6/16 spaces (at least four of five on 8/16), and the largest deviation is one space (`ss5636`, Δ = +0.089), consistent with the σ-heterogeneity boundary condition discussed in §4.4.

Interpretation.. This replication was added at camera-ready; its statistics are post hoc and outside the pre-specified primary and BH-controlled descriptive test families of §3.6. Our main HPO-B experiment (official PriorLabs TabPFN v2) reports Wilcoxon $p > 0.5$ for EI vs. UCB-Gauss across budgets; under GIT-BO's build a weak EI-leaning skew is marginally detectable under Pratt zero-handling ($p = 0.031$) but not under the default zero-discarding convention ($p = 0.24$), with negligible effect size. Any residual difference is consistent with the implementation differences above (bar-distribution handling, per-batch standardization) and the smaller n; the practical conclusion is unchanged, since 64% of runs are exactly identical. GIT-BO's own Appendix B.5 reports a minor UCB-over-EI advantage in its high-dimensional continuous regime, attributed to EI numerical vanishing (Ament et al., 2023); our finding that this does not transfer to discrete low-dimensional HPO pools is consistent with that being a high-dimensional continuous-optimization phenomenon. The two methods are complementary: GIT-BO addresses scaling BO to high-dimensional spaces, while we characterize why a TabPFN-class surrogate works in standard HPO and whether the acquisition choice matters there.

A.15 Baseline hyperparameter configurations

All baselines run at their canonical published hyperparameters. PFNs4BO-HEBO+ and PFNs4BO-BNN use the released weights and heuristic variants from Müller et al. (2023), no input-warping path. DRE (Khazi et al., 2023) is trained per-space with the authors' default ensemble size and learning rate; full configuration details are in the supplementary code. ZeroShotOpt uses the released base checkpoint from Meindl et al. (2025), default temperature, and the 100-step architectural cap discussed in §4.3. SMAC, HEBO, and TPE use their library defaults (SMAC 3 black-box facade; HEBO MACE; Optuna TPE).

A.16 HPO-B per-space breakdown

Per-space Δ values (TabPFN-50 minus each baseline, in normalized-accuracy units) for all 16 HPO-B v3-test spaces are in Table 10 (TabPFN vs. PFNs4BO-HEBO+). Two spaces drive the pooled $\Delta = +0.020$: `ss6766` (2D glmnet, $\Delta = +0.144$, per-space Wilcoxon $p = 0.035$) and `ss6767` (18D xgboost, $\Delta = +0.084$, $p = 0.049$), the two ends of the dimensionality range where PFNs4BO's 18-parameter encoder is stressed. The remaining 14 spaces show $|\Delta| < 0.02$ and do not individually reject at per-space $\alpha = 0.05$. Against DRE, per-space Δ values (Table 11) cluster near zero: 7 spaces favor TabPFN, 8 favor DRE, 1 is an effective tie; Holm-adjusted per-space sign-tests at $\alpha = 0.05$ yield 0/16 significant in favor of TabPFN and 1/16 (`ss5970`, Holm $p = 0.0235$) in favor of DRE. One space, `ss5636`, is an outlier driving the pooled mean toward TabPFN ($\Delta = +0.123$); excluding it shifts the pooled DRE comparison to $\Delta = -0.0003$ (median 0.000), so this one space accounts for the entire pooled delta.

A.17 TabPFN vs DRE learning curve

The 2× sample-efficiency claim in §4.2 applies to budgets ≥ 25. At matched budgets $k \in \{10, 25, 50, 75, 100\}$ the two methods are statistically indistinguishable (Cohen's $|d| < 0.1$ at every budget, CIs straddle zero except at k=75 which is borderline). The 2× claim evaluates TabPFN-k against DRE-2k: at k=25 and k=50, TabPFN-k matches DRE-2k (trivial effect sizes, CIs straddle zero). At k=10 the claim reverses: DRE-20 beats TabPFN-10 by $\Delta = -0.017$ (95% CI $[-0.025, -0.009]$, $d = -0.20$, CI excludes zero). DRE benefits more than TabPFN from each additional trial near cold-start; the 2× advantage emerges only once TabPFN has enough observations to calibrate its posterior.

A.18 ZeroShotOpt replication and metric-definition gap

Our 2D–10D subset ZSO mean normalized accuracy (0.934 ± 0.094, $n = 65$) differs from the reported Table 1 value in Meindl et al. (2025) (0.885 ± 0.009) by +0.049. The root cause is a metric-definition

Table 10: Per-space breakdown, TabPFN-50 vs. PFNs4BO-HEBO+. Δ = TabPFN − HEBO+; positive favors TabPFN. Spaces ordered by dimensionality.

Space	d	n	Δ	Wilcoxon p
ss4796	2	20	+0.009	0.317
ss6766	2	30	**+0.144**	**0.035**
ss5860	2	15	+0.000	—
ss5970	2	30	−0.002	0.185
ss5891	3	30	+0.012	0.394
ss5965	4	35	+0.005	0.119
ss5636	6	30	+0.037	0.505
ss5859	6	30	−0.001	0.645
ss5889	6	10	+0.013	0.500
ss5527	8	30	−0.013	0.507
ss7607	9	35	−0.003	0.184
ss7609	9	35	−0.002	0.083
ss6794	10	30	+0.004	0.116
ss5906	16	10	+0.018	0.375
ss5971	16	30	+0.001	0.130
ss6767	18	30	**+0.084**	**0.049**

Table 11: Per-space breakdown, TabPFN-50 vs. DRE-50. Δ positive favors TabPFN. The ss5636 row is the pooled-mean outlier.

Space	Δ	Direction
ss5636	+0.123	TabPFN (outlier)
ss6767	+0.050	TabPFN
ss5527	−0.042	DRE
ss5906	−0.076	DRE
ss5970	−	DRE (Holm $p = 0.024$)
Other 11 spaces	$\lvert\Delta\rvert < 0.02$	split 5 TabPFN / 5 DRE, 1 tie

difference: we report raw normalized accuracy (the standard HPO-B metric from Arango et al. (2021)), while Meindl et al. (2025) report a gap-closure score $P = (f_m - f_{\text{best}})/(f_m - f^*)$, where f_m is the median initial observation and f^* is the best value achieved by any of their 12 GP-BO baseline variants. This score penalizes runs where initial observations are already near-optimal (small denominator), producing systematically lower values than raw accuracy. Both evaluations use the same released ZSO checkpoint and the same `evaluate_continuous` protocol. All methods in our comparison use raw normalized accuracy, ensuring fair head-to-head comparisons.

A.19 Power considerations at the canonical HPO-B $n = 5$ protocol

The HPO-B protocol of Arango et al. (2021) and Meindl et al. (2025) uses $n = 5$ seeds per (space, test task). Per-space Wilcoxon at $n = 5$ has a floor of $p = 2/2^5 \approx 0.06$, placing individual-space Bonferroni-family survival out of reach for effects below pooled Cohen's $d \approx 0.3$. Directional consistency across spaces is the reliable signal at this sample size: pooled cell-level Wilcoxon ($n = 430$ across 16 spaces at 5 seeds) retains adequate power, and per-space sign-tests across 16 spaces provide Bonferroni-family coverage over a 16-claim secondary family (budget: $\alpha/16 = 0.003$). Power analysis on the existing $n = 5$ data (split by seed index 0–4) shows 44/15/21 cell direction against PFNs4BO-HEBO+ and 54/0/0 against ZSO at step 50 with per-group Cohen's d median +0.76 on HPO-B and +23.29 on YAHPO. A doubled-n replication at $n = 10$ would push per-space Wilcoxon floor to $p = 2/2^{10} \approx 0.002$, admitting Holm-corrected per-space significance for groups with consistent direction. This extension is feasible but out of scope for the current submission.

Table 12: TabPFN vs DRE at matched (k vs k) and half-budget (k vs $2k$) comparisons. Δ = TabPFN − DRE, pooled over $n = 430$ matched cells. Percentile bootstrap CI with $B = 5{,}000$, paired by (space, dataset, seed). Cohen's d is paired.

k (TabPFN)	k' (DRE)	Δ	95% CI	d
Matched budget				
10	10	−0.0011	[−0.010, +0.008]	−0.01
25	25	+0.0034	[−0.005, +0.013]	+0.03
50	50	+0.0083	[−0.002, +0.020]	+0.07
75	75	+0.0113	[+0.001, +0.023]	+0.10
100	100	+0.0088	[−0.001, +0.020]	+0.08
Half budget (k vs $2k$)				
10	20	−0.0172	[−0.025, −0.009]	−0.20
25	50	−0.0049	[−0.014, +0.005]	−0.05
50	100	−0.0002	[−0.010, +0.011]	−0.00

A.20 YAHPO outcome analysis (secondary benchmark)

We evaluated TabPFN-direct BO on all 14 YAHPO-Gym scenarios and compared against SMAC, HEBO, and TPE under a matched-instance protocol at 200 evaluations per run.

Matched-instance scorecard.. Baseline configuration: 3W/9T/2L versus HEBO; 1W/13T/0L versus SMAC; 2W/11T/1L versus TPE. TabPFN wins more scenarios than it loses against each baseline.

Rank-based aggregation.. Following Demšar (2006), the 3-method rank aggregation (Table 3) shows TPE first ($\bar{r} = 1.77$), TabPFN second ($\bar{r} = 1.96$), SMAC third ($\bar{r} = 2.26$). The Friedman test does not reject equal ranks ($p = 0.160$), and no pair exceeds the Nemenyi critical difference (CD = 0.89). These three methods are statistically indistinguishable. These aggregates are computed at 50 evaluations per run; the matched-instance scorecard above is at 200.

Non-saturated scenarios.. `iaml_super` ($d = 29$, 27 conditional): TabPFN reaches the YAHPO ResNet-based surrogate's global optimum on all three instances, confirmed by SMAC converging to the same fixed point at doubled budget. `nb301` (NAS, $d = 34$, single CIFAR10 instance): TabPFN narrowly trails SMAC and TPE at matched budget; closing this gap is future work (§5.2).

Ceiling effect.. 8 of 14 YAHPO scenarios saturate at 200 evaluations: all competent methods converge to within 10^{-3} of the same value. This matches YAHPO-Gym's own SMAC-vs-HEBO comparisons and is a benchmark property, not a method property. Statistical "wins" in these cells reflect fourth-decimal noise.

A.21 Supplementary Package Structure

The supplementary material contains a self-contained, runnable verification package. The directory structure (abbreviated; the package README carries the full listing) is:

```
supplementary/
|-- run_all.py                  Master script (reproduces all claims)
|-- reproduce_single_scenario.py    Single-scenario reproduction demo
|-- requirements.txt            Dependencies: numpy, scipy, statsmodels
|-- README.md                   Quick-start instructions
|
|-- src/                        Core BO implementation
|   |-- tabpfn_direct_searcher.py   The method: encode -> predict -> acquire
|   |-- base.py                     Base searcher interface
```

```
|   +-- helpers.py                  Shared utilities
|
|-- analysis/                    Statistical analysis scripts
|   |-- analyze_pfns4bo_vs_tabpfn.py   Table 1: HEBO+, BNN, DRE, Random
|   |-- analyze_gp_bo_tpe_vs_tabpfn.py Table 1: GP-BO, TPE
|   |-- t1_7_hpob_acq_tost.py         Sec 4.3: acquisition TOST
|   +-- correct_rank_table.py         Sec 5: Friedman/Nemenyi
|
|-- results/                     Experiment outputs
|   |-- raw/                         Raw per-job output, as collected
|   |   |-- tabpfn_hpob.jsonl           TabPFN baseline (430 cells)
|   |   |-- gp_bo_hpob_ucb_ei.jsonl     GP-BO UCB + EI (860 rows)
|   |   |-- tpe_hpob.jsonl              TPE (430 rows)
|   |   |-- w3_hpob_calibration.jsonl   Calibration raw data
|   |   +-- zeroshotopt_*.jsonl         ZSO HPO-B + YAHPO (4 files)
|   +-- rolled-up/                   Aggregates computed from raw
|       |-- w4_paired_deltas.json       DRE-50/100 paired (430 cells)
|       |-- t1_7_hpob_acq_tost_results.json  Acquisition TOST output
|       +-- yahpo_merged_results.json   YAHPO 3-method (rank tables)
|
|-- dre_weights/                 Per-space DRE weights (trained parameters)
|
+-- tests/
    +-- test_reproduce_claims.py      Pytest claim assertions (all pass)
```

To verify: `pip install -r requirements.txt && python run_all.py` (runs in <30 seconds, no GPU required).

A.22 Claim Traceability

Table 13 maps every primary numerical claim to the specific data file and analysis script that produces it. All files are included in the supplementary package.

A.23 Verification Output

Running `python run_all.py` produces the following (abbreviated):

```
TABLE 1: HPO-B head-to-head (TabPFN-50 vs baselines)
  TabPFN baseline: 430 cells loaded
  HEBO+ (n=430)   Paper: +0.020   Ours: +0.0197  [+0.009, +0.032]
  BNN (n=430)     Paper: +0.017   Ours: +0.0174  [+0.009, +0.027]
  GP-BO UCB       Paper: +0.033   Ours: +0.0323  [+0.022, +0.043]
  GP-BO EI        Paper: +0.006   Ours: +0.0054  d=+0.072 (parity)
  TPE             Paper: +0.038   Ours: +0.0379  [+0.027, +0.050]
  DRE-50          Paper: +0.008   Ours: +0.0083  d=+0.072 (parity)
  DRE-100         Paper: -0.0002  Ours: -0.00023 d=-0.002 (parity)

ACQUISITION TOST (delta=0.01):
  Budget 50:  EI=0.0075  LogEI=0.0079  UCB-Bars=0.0003  [PASS]
  Budget 100: EI=0.0026  LogEI=0.0025  Thompson=0.0317  [PASS]
  Budget 200: EI=0.0000  LogEI=0.0000  UCB-Bars=0.0392  [PASS]

CALIBRATION: HPO-B c_95 = 0.938
```

Table 13: Claim traceability: every primary result maps to a data file and analysis script in the supplementary package.

Paper claim	Value	Data file	Script
Table 1: HPO-B head-to-head ($n = 430$ matched cells)			
HEBO+ Δ, CI, p	+0.020 [+0.009, +0.032]	`pfns4bo_hebo_*.jsonl`	`analyze_pfns4bo_vs_tabpfn.py`
BNN Δ, CI, p	+0.017 [+0.009, +0.027]	`pfns4bo_bnn_*.jsonl`	same
GP-BO-UCB Δ, CI	+0.033 [+0.022, +0.044]	`gp_bo_hpob_ucb_ei.jsonl`	`analyze_gp_bo_tpe_vs_tabpfn.p`
GP-BO-EI Δ, d	+0.006, $d = +0.08$	same	same
TPE Δ, CI	+0.038 [+0.027, +0.050]	`tpe_hpob.jsonl`	same
DRE-50 Δ, d	+0.008, $d = +0.07$	`w4_paired_deltas.json`	pre-computed
DRE-100 Δ, p, d	-0.0002, 6.2×10^{-12}	`t1_3_dre_vs_tabpfn.json`	pre-computed
Random Δ	+0.046	`d7_random_baseline_results.json`	pre-computed
§4.4: Acquisition equivalence			
$p_{\text{TOST}} \leq 0.039$	EI/LogEI/UCB-Bars	`t1_7_hpob_acq_tost_results.json`	`t1_7_hpob_acq_tost.py`
Thompson exception	$p = 0.653$ (budget 50)	same	same
UCB-Bars exception	$p = 0.156$ (budget 100)	same	same
Cross-surrogate 0/42	Bonferroni $\alpha = 0.00119$	`t1_8_full14_results.json`	pre-computed
YAHPO TOST $\delta = 0.02$	all 4 reject	`t1_6_yahpo_tost_results.json`	pre-computed
§4.5: Calibration			
HPO-B c_{95}	0.938 ± 0.066	`w3_hpob_calibration.jsonl`	`w3_hpob_calibration.py`
YAHPO c_{95}	0.95 ± 0.05	`t1_5_calibration_results.json`	pre-computed
15/16 within ±0.10		`w3_hpob_calibration.jsonl`	same
§4.3: ZeroShotOpt			
13/16 HPO-B, $p = 0.021$	step 50	`zeroshotopt_*hpob*.jsonl`	`zso_paired_analysis_v2.py`
11/11 YAHPO, $p = 0.001$	step 50 + 100	`zeroshotopt_*yahpo*.jsonl`	same
§A.6: Rank aggregation			
Friedman p (4-way)	0.032	`yahpo_merged_results.json`	`correct_rank_table.py`
Nemenyi CD	no pair exceeds	same	same

All values match the paper within rounding to the reported decimal places, except at the last printed digit of four GP-BO cells, where the script's bootstrap draw differs from the one behind Table 1: GP-BO-UCB +0.0323 vs the table's +0.033 and CI upper +0.0433 vs +0.044; GP-BO-EI +0.0054 vs +0.006 and $d = +0.072$ vs +0.08. Signs, significance calls, and all conclusions are unaffected. The full unabbreviated output is in `expected_output.txt` in the supplementary package.