Automatic Termination for Hyperparameter Optimization

Anonymous[1]

[1]Anonymous Institution

Abstract Bayesian optimization (BO) is a widely popular approach for the hyperparameter optimization (HPO) in machine learning. At its core, BO iteratively evaluates promising configurations until a user-defined budget, such as wall-clock time or number of iterations, is exhausted. While the final performance after tuning heavily depends on the provided budget, it is hard to pre-specify an optimal value in advance. In this work, we propose an effective and intuitive termination criterion for BO that automatically stops the procedure if it is sufficiently close to the global optimum. Our key insight is that the discrepancy between the true objective (predictive performance on test data) and the computable target (validation performance) suggests stopping once the suboptimality in optimizing the target is dominated by the statistical estimation error. Across an extensive range of real-world HPO problems, we show that our termination criterion achieves better test performance compared to existing baselines, such as stopping when the probability of improvement drops below a fixed threshold. We also provide evidence that these baselines are, compared to our method, sensitive to the choices of their own hyperparameters. Additionally, we find that overfitting may occur in the context of HPO, which is arguably an overlooked problem in the literature, and show how our termination criterion helps to mitigate this phenomenon on both small and large datasets.

1 Introduction

While the performance of machine learning algorithms crucially depends on their hyperparameters, setting them correctly is typically a tedious and expensive task. Hyperparameter optimization (HPO) emerged as a new sub-field in machine learning that tries to automatically determine how to configure a machine learning model. One of the most successful strategies for HPO is Bayesian optimization (BO; Močkus, 1975; Chen et al., 2018; Snoek et al., 2012; Melis et al., 2018) - a powerful framework for sequentially optimizing a costly blackbox objective, that is the predictive performance of a model configured with certain hyperparameters. BO iteratively searches for a better predictive performance via (i) training a probabilistic model on the evaluations of the models performance and (ii) selecting the most promising next hyperparameter candidate.

In practice, the quality of the final solution found by BO heavily depends on a budget, such as the number of BO iterations or wall-clock time, which needs to be specified by a user in advance. If this budget is too small, BO might result into hyperparameters of a poor predictive performance. If the budget is too large, compute resources will be wasted or it may even result in *overfitting* as we will show in our experiments. Automatically terminating the sequential procedure of BO is a rather under-explored topic, in contrast to the more widely considered orthogonal direction of speeding up HPO via stopping the model training (Klein et al., 2017; Dai et al., 2019; Li et al., 2017).

A naive approach suggests terminating BO if the best found solution remains unchanged for some subsequent BO iterations. Another approach is to track the probability of improvement (Lorenz et al., 2016) or the expected improvement (Nguyen et al., 2017), and stop the optimization process once it falls below a given threshold. However, determining this threshold may in practice be less intuitive than setting the number of iterations or the wall-clock time. Instead of stopping BO completely, in McLeod et al. (2018), it is proposed to switch to local optimization when the global regret is smaller than a pre-defined target. This condition can also be used to terminate BO early, but it comes with additional complexity such as identifying a (convex) region for local optimization and again a predefined budget.

 © 2022 the authors, released under CC BY 4.0

In this work, we propose a simple and interpretable automatic termination criterion for BO. The criterion consists of two main ingredients: (i) a high-probability confidence bound on the regret (i.e., the difference of our current solution to the global optimum) and (ii) the termination threshold. The first already allows a user to specify a desired tolerance that defines *how accurate should the final solution be compared to the global optimum.* For the case when cross-validation is used, we recommend a threshold based on statistical properties of the cross-validation estimator. This threshold takes into account the irreducible discrepancy between the actual HPO objective (i.e., performance on new data) and the target function optimized via BO (i.e., the validation error). Our extensive empirical evaluation on a variety of HPO and neural architecture search (NAS) benchmarks suggests that our method is more robust and effective in maintaining the final solution quality than common baselines. We also surface overfitting effects in HPO on both small and large datasets, arguably an overlooked problem, and demonstrate that our termination criterion helps to mitigate it.

2 Background

Bayesian optimization (BO) refers to methods for gradient-free optimization of an objective $f : \Gamma \to \mathbb{R}$ in an iterative manner. At every step t, a learner selects an input $\gamma_t \in \Gamma$ and observes a noisy output

$$y_t \triangleq f(\gamma_t) + \varepsilon_t,$$

where ε_t is typically assumed to be i.i.d. (sub)-Gaussian noise with some variance σ_ε^2. The decision of the next input to evaluate depends on a probabilistic model, used to approximate the objective f, and an acquisition function, which determines the decision rule. A popular choice for the probabilistic model is a *Gaussian process (GP)*: $f \sim GP(\mu, \kappa)$ (Rasmussen and Williams, 2006), specified by some mean function $\mu : \Gamma \to \mathbb{R}$ and some kernel $\kappa : \Gamma \times \Gamma \to \mathbb{R}$. As observations $y_{1:t} = [y_1, \dots, y_t]^\top$ for the selected inputs $G_t = \{\gamma_1, \dots, \gamma_t\}$ are being collected, they are used to update the posterior belief of the model defined by the posterior mean $\mu_t(\gamma)$ and variance $\sigma_t^2(\gamma)$:

$$\mu_t(\gamma) = \kappa_t(\gamma)^T (K_t + \sigma_\varepsilon^2 I)^{-1} y_{1:t}, \qquad \sigma_t^2(\gamma) = \kappa(\gamma, \gamma) - \kappa_t(\gamma)^\top (K_t + \sigma_\varepsilon^2 I)^{-1} \kappa_t(\gamma), \tag{1}$$

where $(K_t)_{i,j} = \kappa(\gamma_i, \gamma_j)$ and $\kappa_t(\gamma)^T = [\kappa(\gamma_1, \gamma), \dots, \kappa(\gamma_t, \gamma)]^T$. The next input to query is determined by an *acquisition function* that aims to trade off exploration and exploitation. Common choices include probability of improvement (Kushner, 1963), entropy search (Hennig and Schuler, 2012) and GP upper-confidence bound (Srinivas et al., 2010) to name a few. The convergence of BO can be quantified via *(simple) regret*:

$$r_t := f(\gamma_t^*) - f(\gamma^*), \tag{2}$$

where γ^* is the global optimizer of f and $\gamma_t^* = \arg\min_{\gamma \in G_t} f(\gamma)$. Specifying adequate tolerance that defines how small the regret should be to terminate BO is of high importance as it determines both the quality and the cost of the solution. However, this criterion cannot be directly evaluated in practice, as the input γ^* and the optimum $f(\gamma^*)$ are not known.

Hyperparameter optimization (HPO) is a classical application for BO. Consider a supervised learning problem that requires to train a machine learning model (e.g., a neural network) $\mathcal{M}$ on some feature-response data points $\mathcal{D} = \{(\mathrm{x}_i, \mathrm{y}_i)\}_{i=1}^n$ sampled i.i.d. from some unknown data distribution P. The model is obtained by running a training algorithm (e.g., optimizing the weights of the neural network via SGD) on $\mathcal{D}$, both of which depend on *hyperparameters* γ (e.g., learning rates used, batch size, etc.). We use the notation $\mathcal{M}_\gamma(\mathrm{x}; \mathcal{D})$ to refer to the prediction that the model produced by $\mathcal{M}$ makes for an input x, when trained with hyperparameters γ on data $\mathcal{D}$. Given some loss function $\ell(\cdot, \cdot)$, the *population risk* of the model on unseen data points is given by the expected loss $\mathbb{E}_P[\ell(\mathrm{y}, \mathcal{M}_\gamma(\mathrm{x}, \mathcal{D}))]$. The main objective of HPO is to identify hyperparameters γ,

such that the resulting model minimizes the population risk:

$$f(\gamma) = \mathbb{E}_{(\mathrm{x},\mathrm{y})\sim P}\big[\ell\big(\mathrm{y}, \mathcal{M}_\gamma(\mathrm{x}, \mathcal{D})\big)\big], \qquad \gamma^* = \arg\min_{\gamma\in\Gamma} f(\gamma). \tag{3}$$

In practice, however, the population risk cannot be evaluated since P is unknown. Thus, typically, it is estimated on a separate finite validation set $\mathcal{D}_V$ drawn from the same distribution P. Practical HPO focuses on minimizing the *empirical estimator* $\hat{f}(\gamma)$ of the expected loss $f(\gamma)$ leading to the optimizer $\gamma^*_{\mathcal{D}}$:

$$\hat{f}(\gamma) = \frac{1}{|\mathcal{D}_V|}\sum_{(\mathrm{x}_i,\mathrm{y}_i)\in\mathcal{D}_V} \ell\big(\mathrm{y}_i, \mathcal{M}_\gamma(\mathrm{x}_i, \mathcal{D})\big), \qquad \gamma^*_{\mathcal{D}} = \arg\min_{\gamma\in\Gamma} \hat{f}(\gamma). \tag{4}$$

At its core, BO-based HPO sequentially evaluates the empirical estimator $\hat{f}(\gamma_t)$ for promising hyperparameters γ_t and terminates after some specified number of BO rounds, reporting the solution $\gamma^*_t = \arg\min_{\gamma\in G_t} \hat{f}(\gamma)$, where $G_t = \{\gamma_1, \ldots, \gamma_t\}$ are the solutions considered so far. We can define the simple regret for the reported solution w.r.t. the validation loss by

$$\hat{r}_t := \hat{f}(\gamma^*_t) - \hat{f}(\gamma^*_{\mathcal{D}}). \tag{5}$$

Inconsistency in the optimization objective. Importantly, the true HPO objective $f(\gamma)$ in Eq. (3) and the empirical surrogate $\hat{f}(\gamma)$ in Eq. (4) used for tuning by BO generally do not coincide. Therefore, existing BO approaches may yield sub-optimal solutions to the population risk minimization, even if they succeed in globally optimizing $\hat{f}(\gamma)$. This issue, however, is typically neglected in practical HPO. In contrast, we propose a termination condition for BO motivated by the discrepancy in the objectives.

3 Termination criterion for Hyperparameter Optimization

This section firstly motivates why early termination of HPO can be beneficial and then addresses the following two questions: (1) How to estimate the unknown simple regret and (2) What threshold of the simple regret can be used to stop HPO.

3.1 Motivation for the termination criterion

We start by analysing the effect of optimizing $\hat{f}$ in lieu of f. We observe that challenges in optimizing f are both due to the statistical error of the empirical BO objective $\hat{f}(\gamma)$ and the sub-optimality of the BO candidates, encoded in the simple regret $\hat{r}_t$. The key insight of the following proposition is that iteratively reducing $\hat{r}_t$ to 0 may not bring any benefits if the statistical error dominates.

Proposition 1. *Consider the expected loss f and its estimator $\hat{f}$ defined, respectively, in Eqs.* (3) *and* (4)*, and assume the statistical error of the estimator is bounded as $||\hat{f} - f||_\infty \leq \epsilon_{st}$ for some $\epsilon_{st} \geq 0$. Let γ^* and $\gamma^*_{\mathcal{D}}$ be their optimizers: $\gamma^* = \arg\min_{\gamma\in\Gamma} f(\gamma)$ and $\gamma^*_{\mathcal{D}} = \arg\min_{\gamma\in\Gamma} \hat{f}(\gamma)$. Let γ^*_t be some candidate solution to $\min_{\gamma\in\Gamma} \hat{f}(\gamma)$ with sub-optimality in function value $\hat{r}_t := \hat{f}(\gamma^*_t) - \hat{f}(\gamma^*_{\mathcal{D}})$. Then the gap in generalization performance $f(\gamma^*_t) - f(\gamma^*)$ can be bounded as follows:*

$$\begin{aligned} f(\gamma^*_t) - f(\gamma^*) &\leq \underbrace{f(\gamma^*_t) - \hat{f}(\gamma^*_t)}_{\leq\epsilon_{st}} + \underbrace{\hat{f}(\gamma^*_t) - \hat{f}(\gamma^*_{\mathcal{D}})}_{=\hat{r}_t} + \underbrace{\hat{f}(\gamma^*_{\mathcal{D}}) - \hat{f}(\gamma^*)}_{\leq 0} + \underbrace{\hat{f}(\gamma^*) - f(\gamma^*)}_{\leq\epsilon_{st}} \\ &\leq 2\epsilon_{st} + \hat{r}_t. \end{aligned}$$

Moreover, without further restrictions on f, $\hat{f}$, γ^*_t and γ^*, the upper bound is tight.

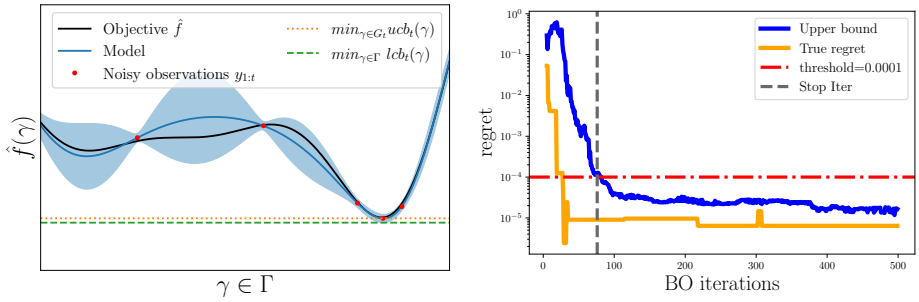


Figure 1: **Left**: Visualization of the upper bound for $\hat{r}_t$. The gap between green and orange lines is the estimate of the upper bound for $\hat{r}_t$. **Right**: Illustration of automated BO termination when tuning MLP on the `naval` dataset from HPO-Bench (Klein and Hutter, 2019) with the BORE optimizer (Tiao et al., 2021).

Proof: The components in the inequality can be bounded due to the following:

$$(1)\ f(\gamma_t^*) - \hat{f}(\gamma_t^*) \leq |f(\gamma_t^*) - \hat{f}(\gamma_t^*)| \leq \max_{\gamma \in \Gamma} |f(\gamma) - \hat{f}(\gamma)| = ||\hat{f} - f||_\infty \leq \epsilon_{st},$$

$$(2)\ \gamma_{\mathcal{D}}^* = \arg\min_{\gamma \in \Gamma} \hat{f}(\gamma) \quad \Rightarrow \quad \forall \gamma \in \Gamma : \hat{f}(\gamma_{\mathcal{D}}^*) - \hat{f}(\gamma) \leq 0 \quad \Rightarrow \quad \hat{f}(\gamma_{\mathcal{D}}^*) - \hat{f}(\gamma^*) \leq 0. \quad \blacksquare$$

The proposition bounds the sub-optimality of the target objective f in terms of the statistical error ϵ_{st} and the simple regret $\hat{r}_t$. This naturally suggests terminating HPO at a candidate γ_t^* for which the simple regret $\hat{r}_t$ is of the same magnitude as the statistical error ϵ_{st}, since further reduction in $\hat{r}_t$ may not improve notably the true objective. However, neither of the quantities ϵ_{st} and $\hat{r}_t$ are known.

Below, we propose a termination criterion that relies on estimates of both quantities. Firstly, we show how to use confidence bounds on $\hat{f}(\gamma)$ to obtain high probability upper bounds on the simple regret $\hat{r}_t$ (Srinivas et al., 2010; Ha et al., 2019). Secondly, we estimate the statistical error ϵ_{st} in the case of cross-validation (Stone, 1974; Geisser, 1975) where the model performance is defined as an average over several training-validation runs. To this end, we rely on the statistical characteristics (i.e., variance or bias) of such cross-validation-based estimator that are theoretically studied by Nadeau and Bengio (2003) and Bayle et al. (2020).

3.2 Building blocks of the termination criterion

Upper bound for the simple regret $\hat{r}_t$. The key idea behind bounding $\hat{r}_t$ is that, as long as the GP-based approximation of $\hat{f}(\cdot)$ is well-calibrated, we can use it to construct high-probability confidence bounds for $\hat{f}(\cdot)$. In particular, Srinivas et al. (2010) show that, as long as $\hat{f}$ has a bounded norm in the reproducing kernel Hilbert space (RKHS) associated with the covariance function κ used in the GP, $\hat{f}(\gamma)$ is bounded (with high probability) by lower and upper confidence bounds $\mathrm{lcb}_t(\gamma) = \mu_t(\gamma) - \sqrt{\beta_t}\sigma_t(\gamma)$ and $\mathrm{ucb}_t(\gamma) = \mu_t(\gamma) + \sqrt{\beta_t}\sigma_t(\gamma)$. Hereby, β_t is a parameter that ensures validity of the confidence bounds (see Appendix A.2.3 for practical discussion and ablation study).

Consequently, we can bound the unknown $\hat{f}(\gamma_t^*)$ and $\hat{f}(\gamma_{\mathcal{D}}^*)$ that define the sub-optimality $\hat{r}_t$:

$$\hat{r}_t = \hat{f}(\gamma_t^*) - \hat{f}(\gamma_{\mathcal{D}}^*) \leq \min_{\gamma \in G_t} \mathrm{ucb}_t(\gamma) - \min_{\gamma \in \Gamma} \mathrm{lcb}_t(\gamma) =: \bar{r}_t, \tag{6}$$

where the inequality for $\hat{f}(\gamma_t^*)$ is due to the definition of the reporting rule $\gamma_t^* = \arg\min_{\gamma \in G_t} \hat{f}(\gamma)$ over the evaluated points $G_t = \{\gamma_1, \ldots, \gamma_t\}$. We illustrate the idea with an example in Fig. 1.

Termination threshold. We showed how to control the optimization error via the (computable) regret upper bound $\bar{r}_t$ above. We now explain when to stop BO, i.e., how to choose some threshold

ϵ_{BO} and an iteration $T : \bar{r}_T \leq \epsilon_{BO}$. Following Proposition 1, we suggest setting ϵ_{BO} to be of similar magnitude as the statistical error ϵ_{st} of the empirical estimator (since smaller regret $\hat{r}_t$ is not beneficial when ϵ_{st} dominates). In case of cross-validation being used for HPO, one can estimate this statistical error ϵ_{st} and we further discuss how it can be done.

Cross-validation is the standard approach to compute an estimator $\hat{f}(\gamma)$ of the population risk. The data $\mathcal{D}$ is partitioned into k equal-sized sets $\mathcal{D}_1, \ldots, \mathcal{D}_k$ used for (a) training the model $\mathcal{M}_\gamma(\cdot; \mathcal{D}_{-i})$, where $\mathcal{D}_{-i} = \cup_{j \neq i} \mathcal{D}_i$ (i.e., training on all but the i-th fold), and (b) validating $\mathcal{M}_\gamma(\cdot; \mathcal{D}_i)$ on the i-th fold of the data. These two steps are repeated in a loop k times, and then the average over k validation results is computed.

The statistical error ϵ_{st} of an estimator can be characterised in terms of its variance $\mathrm{Var}\hat{f}(\gamma) = \mathbb{E}[(\hat{f}(\gamma) - \mathbb{E}\hat{f}(\gamma))^2]$ and bias $B(\gamma) = \mathbb{E}[\hat{f}(\gamma)] - f(\gamma)$, where the latter can be neglected in case of cross-validation (Bayle et al., 2020). Though the variance $\mathrm{Var}\hat{f}(\gamma)$ of the cross-validation estimate is generally unknown, Nadeau and Bengio (2003) propose an unbiased estimate for it. Specifically, for the sample variance (denoted as s^2_{cv}) of k-fold cross-validation, a simple post-correction technique to estimate the variance $\mathrm{Var}\hat{f}(\gamma)$ is

$$\mathrm{Var}\hat{f}(\gamma) \approx \left(\frac{1}{k} + \frac{|\mathcal{D}_i|}{|\mathcal{D}_{-i}|}\right) s^2_{\mathrm{cv}}(\gamma), \tag{7}$$

where $|\mathcal{D}_i|, |\mathcal{D}_{-i}|$ are the set sizes. For example, in the case of 10-fold cross-validation we have $\mathrm{Var}\hat{f}(\gamma) \approx 0.21 s^2_{\mathrm{cv}}(\gamma)$. We are now ready to propose our termination condition in the following.

Termination condition for BO. *Consider the setup of Proposition 1 where $\hat{f}(\cdot)$ is a cross-validation-based estimator being iteratively minimized by BO. Let γ^*_t the solution reported in round t, and $\bar{r}_t$ defined in Eq.* (6) *be the simple regret bound computed at each iteration t. Let the variance* $\mathrm{Var}\hat{f}(\gamma^*_t)$ *of the estimator $\hat{f}(\cdot)$ be approximated according to Eq.* (7)*. Then, BO is terminated once:*

$$\bar{r}_t < \sqrt{\mathrm{Var}\hat{f}(\gamma^*_t)}. \tag{8}$$

Intuitively, the termination is triggered once the maximum plausible improvement becomes less than the standard deviation of the estimate. This variance-based termination condition adapts to different algorithms or datasets and its computation comes with negligible computational cost on top of cross-validation. The pseudo-code for the criterion is summarised in Algorithm 1. If cross-validation cannot be used or is computationally prohibitive, the user can define the right-hand side of the termination condition. In this case, the upper bound on the left-hand side still has an intuitive interpretation: the user can set the threshold based on their desired solution accuracy. This case is demonstrated in Fig. 1, with an example of automatic termination for tuning an MLP.

4 Experiments

The main challenge of a termination criterion for HPO is to balance between reducing runtime and performance degradation. We thus study in experiments how the speed-up gained from different termination criterions affects the final test performance. To this end, we define two new metrics that account for the trade-off between resources saved and performance drop and provide a list of reasonable baselines. The code to reproduce the experimental results is publicly available.[1]

Baselines. We consider the following baselines:

- Näive convergence test controlled by a parameter i (referred as Conv-i): stopping BO if the *best* observed validation metric remains unchanged for i consecutive iterations. This condition heavily relies on pre-specified i and we consider common in practice values $i = \{10, 30, 50\}$.

[1] `https://anonymous.4open.science/r/BO-early-stopping-555E`

- Threshold for probability of improvement (PI; Lorenz et al., 2016): stopping BO once PI drops below a pre-defined threshold of $\{10^{-5}, 10^{-9}, 10^{-13}\}$.
- Threshold for expected improvement (EI) (PI; Nguyen et al., 2017): stopping BO once EI drops below a pre-defined threshold of $\{10^{-9}, 10^{-13}, 10^{-17}\}$.

  Empirically, we observe that all thresholds lead to significant speeds up but, at the same time, to a severe degradation of the final solution quality (see Appendix A.2 for full results).

Metrics. We measure the effectiveness of a termination criterion via two metrics quantifying (i) the change in test error on a held-out dataset and (ii) the time saved. Given a budget of T BO iterations, we compare the test error y_T after T iteration to the test error y_{es} after early stopping is triggered. For each experiment, we compute the *relative test error change* RYC as

$$\text{RYC} = \frac{y_T - y_{es}}{\max(y_T, y_{es})}. \tag{9}$$

This allows us to aggregate the results over different algorithms and datasets, as RYC $\in [-1, 1]$ and can be interpreted as follows: A positive RYC represents an improvement in the test error when applying early stopping, while a negative RYC indicates the opposite. Similarly, let the total training time for a predefined budget T be t_T and the total training time when early stopping is triggered be t_{es}. Then the *relative time change* RTC is defined as

$$\text{RTC} = \frac{t_T - t_{es}}{t_T} \tag{10}$$

indicates a reduction in total training time, RTC $\in [0, 1]$. While reducing training time is desirable, it should be noted that this can be achieved through any simple stopping criterion (e.g., consider interrupting HPO with a fixed probability after every iteration). In other words, the RTC is not a meaningful metric when decoupled from the RYC and, thus, both need to be considered in tandem.

How to select the data to estimate the bound? Since we are only interested in the upper bound of the simple regret, we conjecture that using only the top performing hyperparameter evaluations may improve the estimation quality. To validate this, we use results of BORE (Tiao et al., 2021) on the `naval` dataset from HPO-Bench (Klein and Hutter, 2019) where we can quantify the true regret (see Section 4.2). We compute the upper bound by Eq. (6) using three options: 100%, top 50% or top 20% of the hyperparameters evaluated so far and measure the distance to the true regret (see Fig. 2). From Fig. 2, fitting a surrogate model with all the hyperparameter evaluations poses a challenge for estimating the upper bound of the regret, which is aligned with recent findings on more efficient BO with local probabilistic model, especially for high-dimensional problems (Eriksson et al., 2019). Using the top 20% evaluations gives the best upper bound estimation quality in the median, at the cost of the most under-estimations of the true regret (2553). Our method would stop too early due to the under-estimation, thus negatively impacting the RYC score, as shown in Fig. 2. *As a result, we use the top 50% hyperparameters evaluations for the upper bound estimation throughout this paper.*

4.1 Hyperparameter optimization with cross-validation

We use a Matérn 5/2 kernel for the GP surrogate to compute the bounds. The hyperparameters of the GP are found based on a type II maximum likelihood estimation. We consider: XGBoost (XGB) with 9 hyperparameters and Random Forest (RF) with three hyperparameters on 19 small tabular datasets using cross-validation. All relevant details can be found in Appendix A.1. We optimize classification error and rooted mean square error for classification and regression datasets, respectively.

For our termination criterion, we use Eq. (7) as our stopping threshold. We apply automatic termination only after the first 20 iterations to ensure a robust fit of the surrogate model both for our method and the baselines. We present the aggregated results across datasets for XGB and RF in Fig. 3.

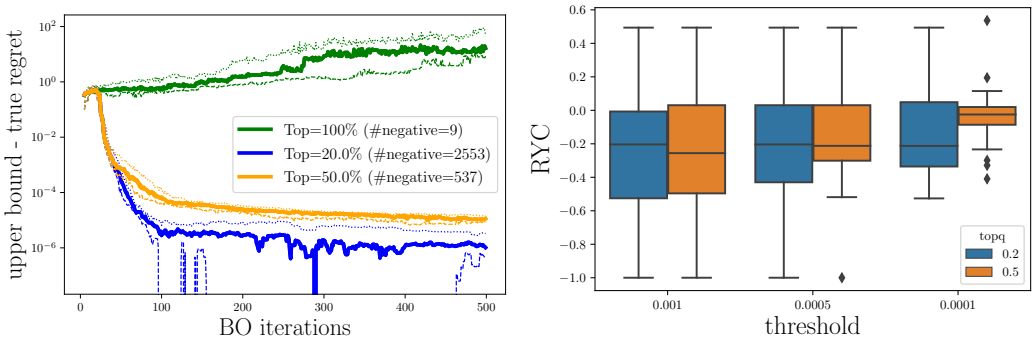


Figure 2: The upper bound estimation quality is affected by the set of hyperparameters evaluations used in the surrogate model training. **Left**: Bound quality for using all, top 50% and top 20% hyperparameter evaluations, measured by the difference between upper bound and true regret. Solid line represents the median over 50 replicates, the dashed is 20'th quantile and dotted line is 80'th quantile. The legend also shows the number of negative differences (the upper bound is smaller than the true regret). **Right**: Box plots of RYC scores when using the top 50% and top 20% hyperparameter evaluations under common thresholds.

From Fig. 3, a general trend on i in the convergence check baseline is visible: as i increases, the speed-up decreases while the solution quality increases. The EI and PI based stopping criteria behave similarly in terms of both RTC and RYC scores. The methods tend to stop BO very early, thus leading to significant speed up. However, maybe not surprisingly, such an aggressive early stopping leads to worse test performance on average. The convergence check baseline and our method could achieve various trade-offs between speed and solution quality effectively by changing patience hyperparameter or the post-correction term, and our method prioritizes quality over speed.

Furthermore, the standard deviations of the RYC and RTC scores also provide us with interesting insights. The RYC variances of our method are usually smaller than the baselines ones, indicating that we successfully maintain a high solution quality across a wide range of scenarios. On the other hand, the RTC variances of our method are usually higher than the baselines, which highlights that our method adapts to different scenarios rather than stopping BO at similar iterations.

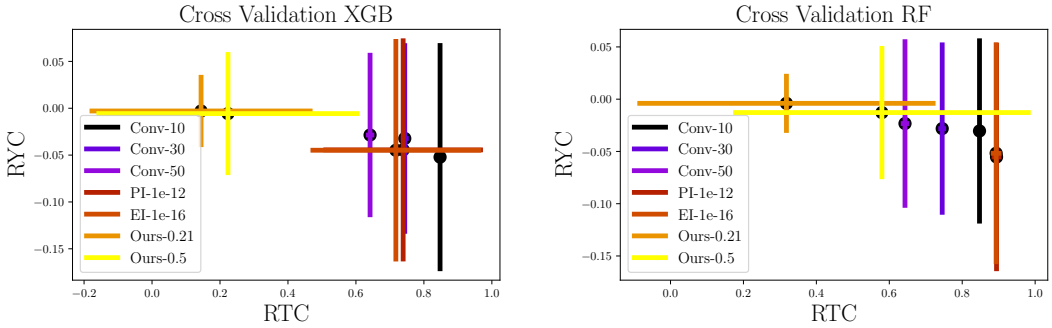


Figure 3: The mean and standard deviation of RYC and RTC scores for the compared automatic termination methods when using cross validation in the hyperparameter evaluation when tuning XGB (left) and RF (right). The mean value is shown as the large dot and the standard deviation is shown as an error bar in both dimensions.

4.2 Neural Hyperparameter and Architecture Search

A main advantage or our termination criterion is that it is applicable for any iterative HPO method. To demonstrate this, we apply it to several state-of-the-art methods: TPE (Bergstra et al., 2011), BORE (Tiao et al., 2021), GP-BO (Snoek et al., 2012) as well as random search (RS) (Bergstra and Bengio, 2012). For each method and dataset, we perform 50 independent runs with a different seed.

We consider two popular tabular benchmark suites: NAS-HPO-Bench (Klein and Hutter, 2019), which mimics the hyperparameter and neural architecture search of multi-layer perceptrons on tabular regression datasets, and NAS-Bench-201 (Dong and Yang, 2020) for neural architecture search on image classification datasets. Notice that for NAS-Bench-201, we used *validation* metrics to compute RYC instead of test metrics, thus, no positive RYC scores are observed. For a detailed description of these benchmarks we refer to the original paper. We consider the following thresholds on the final regret $\{0.0001, 0.001, 0.01\}$ here, corresponding to a loss of performance of 0.01%, 0.1% and 1%, respectively. Due to space constraints, we will only show results for BORE in Figure Fig. 4 and Fig. 5, and provide all other results in the appendix.

While *no method* Pareto dominates the others, our termination criterion shows a similar trend as in Section 4.1 by prioritising accuracy over speed. Users need to choose the threshold based on their own preference regarding the speed-accuracy tradeoff, i.e a higher threshold saves more wall-clock time but potentially leads to a higher drop in performance. We further show the distribution of true regrets at the stopping iteration triggered by our method with the considered thresholds on HPO-Bench in Fig. 5.

From Fig. 5, with a high threshold of 0.01, all the experiments (4 datasets with 50 replicates) are early stopped by our method and 41 (20%) experiments end up with true regret being higher than the threshold. With a low threshold of 0.0001, 112 experiments are stopped and 12 (10.7%) experiments end up with true regret above the threshold. In short, our method achieves 80% to 90% success rate where the true regret is within the user-defined tolerance.

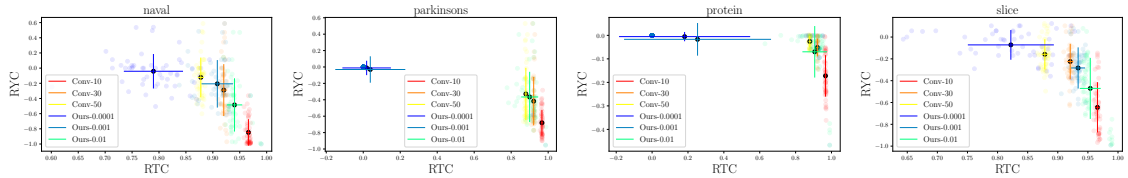


Figure 4: Mean (large dot) and standard deviation (error bar) of RYC and RTC scores for all methods for HPO-Bench datasets.

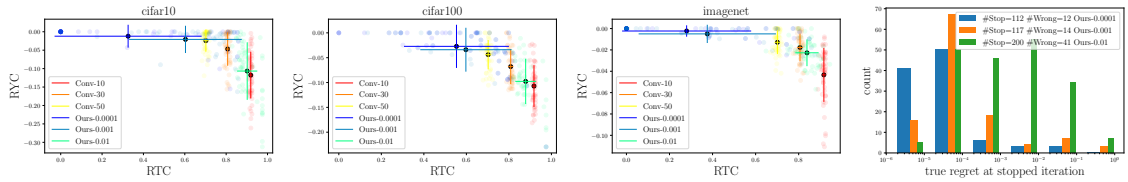


Figure 5: First three figures from the left show the mean and standard deviation of RYC and RTC scores for all methods on NAS-Bench-201. Note that *validation metrics* are used in these experiments, thus no positive RYC scores are observed. The mean value is shown in the big dot and the standard deviation is shown as error bar in both dimensions. The Figure on the right shows a distribution of true regrets at the stopping iteration triggered by our method with different thresholds for HPO-Bench.

For every method we aggregated the scores over datasets with other HPO optimizers in Fig. 6. We can see that the speed up of the convergence check baseline is affected very mildly by the

optimizers while the RYC scores largely depend on the optimiser: RYC scores with random search are worse than with BORE. In contrast, the RYC scores for our termination criterion are similar across optimisers, especially for smaller thresholds. On the other hand, the speed up for a given threshold tends to vary. This can be explained by the difference of the optimizer's performance, for examples random search is not as efficient as BORE, and hence the regret is mostly above the stopping threshold. In summary, while convergence check baselines are by design robust in terms of time saved, our method is more robust in terms of maintaining the solution quality.

4.3 Overfitting in BO for Hyperparameter Optimization

Proposition 1 emphasises an important problem of BO-based HPO: while focusing (and minimizing) the validation error, we cannot fully reduce the discrepancy between the validation and test errors. Empirically, we show this might happen when correlation between the test and validation errors is low, thus improvement in validation performance does not lead to the better test results. A particular example of such low correlation in the *small* error region is presented in Fig. 8 (Appendix A.2) when tuning XGB and Random Forest on tst-census dataset.

In our experiments, a positive RYC score is an indicator of overfitting, showing that the test error at the terminated iteration is lower than the test error in the final round. We observe positive RYC scores in both Fig. 3 and Fig. 4, one with cross-validation on small datasets and one with medium-sized datasets. Hence, we would like raise attention to the possible overfitting issue that occurs in HPO for which our method can be used as a plugin to mitigate overfitting.

5 Conclusion

Despite the usefulness of hyperparameter optimizations (HPO), setting a budget in advance remains a challenging problem. In this work, we propose an automatic termination criterion that can be plugged into many common HPO methods. The criterion uses an intuitive and interpretable upper bound of simple regret, allowing users explicitly control the accuracy loss. In addition, when cross-validation is used in the evaluations of hyperparameters, we propose to use an analytical threshold rooted from the variance of cross validation results.

The experimental results suggest that our method can be robustly used across many HPO optimizers. Depending on the user-defined thresholds, with 80% to 90% chance, our method achieves true regret within that threshold, saving unnecessary computation and reducing energy consumption. We also observe that overfitting exists in HPO even when cross-validation is used. We hope our work will draw the attention of the HPO community to the practical questions of how to set budget in advance and how to mitigate overfitting when tuning hyperparameters in machine learning.

6 Limitations and Broader Impact

In a broader context, we highlight that BO can reduce the computational cost required to tune ML models, mitigating the energy consumption and carbon footprint associated with brute force techniques such as random and grid search. The automatic termination criterion we presented in this work can have a positive societal impact by further reducing the cost of tuning ML models.

Automatic BO terminating is a rather under-explored topic, and this work attempts to bring attention to the nuances of this problem in case of HPO. We empirically demonstrate the effectiveness of the proposed, however further study of better variance estimate might be beneficial and we leave the modeling and estimation of this variance for the future work.

7 Reproducibility 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]

   (b) Did you describe the limitations of your work? [Yes]

   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss any societal impacts in Section 6.

   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? `https://automl.cc/ethics-accessibility/` [Yes]

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [Yes]

   (b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit version), an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] We provide the link to the code in Section 4.

   (b) Did you include the raw results of running the given instructions on the given code and data? [Yes]

   (c) Did you include scripts and commands that can be used to generate the figures and tables in your paper based on the raw results of the code, data, and instructions given? [Yes]

   (d) Did you ensure sufficient code quality such that your code can be safely executed and the code is properly documented? [Yes]

   (e) Did you specify all the training details (e.g., data splits, pre-processing, search spaces, fixed hyperparameter settings, and how they were chosen)? [Yes]

   (f) Did you ensure that you compared different methods (including your own) exactly on the same benchmarks, including the same datasets, search space, code for training and hyperparameters for that code? [Yes]

   (g) Did you run ablation studies to assess the impact of different components of your approach? [Yes]

   (h) Did you use the same evaluation protocol for the methods being compared? [Yes]

   (i) Did you compare performance over time? [Yes]

   (j) Did you perform multiple runs of your experiments and report random seeds? [Yes]

   (k) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

   (l) Did you use tabular or surrogate benchmarks for in-depth evaluations? [N/A]

   (m) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]

(n) Did you report how you tuned hyperparameters, and what time and resources this required (if they were not automatically tuned by your AutoML method, e.g. in a NAS approach; and also hyperparameters of your own method)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets…

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [N/A]

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

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] These are publicly available assets.

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects…

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

Bayle, P., Bayle, A., Janson, L., and Mackey, L. (2020). Cross-validation confidence intervals for test error. In *Neural Information Processing Systems (NeurIPS)*.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyperparameter optimization. In *Neural Information Processing Systems (NeurIPS)*.

Bergstra, J. and Bengio, Y. (2012). Random search for hyperparameter optimization. *Journal of Machine Learning Research*.

Chen, Y., Huang, A., Wang, Z., Antonoglou, I., Schrittwieser, J., Silver, D., and de Freitas, N. (2018). Bayesian optimization in AlphaGo.

Dai, Z., Yu, H., Low, K. H., and Jaillet, P. (2019). Bayesian optimization meets Bayesian optimal stopping. In *ICML*.

Dong, X. and Yang, Y. (2020). Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. (2019). Scalable global optimization via local Bayesian optimization. In *Neural Information Processing Systems (NeurIPS)*.

Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of The American Statistical Association*.

Ha, H., Rana, S., Gupta, S., Nguyen, T., Tran-The, H., and Venkatesh, S. (2019). Bayesian optimization with unknown search space. In *Neural Information Processing Systems (NeurIPS)*, pages 11795–11804. Curran Associates, Inc.

Hennig, P. and Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*.

Kirschner, J., Bogunovic, I., Jegelka, S., and Krause, A. (2020). Distributionally robust Bayesian optimization. In *Conference on Artificial Intelligence and Statistics (AISTATS)*.

Klein, A., Falkner, S., Springenberg, J. T., and Hutter, F. (2017). Learning curve prediction with Bayesian neural networks. In *ICLR*.

Klein, A. and Hutter, F. (2019). Tabular benchmarks for joint architecture and hyperparameter optimization. *arXiv:1905.04970*.

Kushner, H. J. (1963). A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Joint Automatic Control Conference*.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.

Lorenz, R., Monti, R. P., Violante, I. R., Faisal, A. A., Anagnostopoulos, C., Leech, R., and Montana, G. (2016). Stopping criteria for boosting automatic experimental design using real-time fMRI with Bayesian optimization. *arXiv:1511.07827*.

Makarova, A., Usmanova, I., Bogunovic, I., and Krause, A. (2021). Risk-averse Heteroscedastic Bayesian optimization. In *Conference on Neural Information Processing Systems (NeurIPS)*.

McLeod, M., Roberts, S., and Osborne, M. A. (2018). Optimization, fast and slow: optimally switching between local and Bayesian optimization. In *International Conference on Machine Learning (ICML)*.

Melis, G., Dyer, C., and Blunsom, P. (2018). On the state of the art of evaluation in neural language models. In *International Conference on Learning Representations (ICLR)*.

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization Techniques*.

Nadeau, C. and Bengio, Y. (2003). Inference for the generalization error. In *Neural Information Processing Systems (NeurIPS)*.

Nguyen, V., Gupta, S., Rana, S., Li, C., and Venkatesh, S. (2017). Regret for expected improvement over the best-observed value and stopping condition. In *Asian Conference on Machine Learning*.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Conference on Neural Information Processing Systems (NeurIPS)*.

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

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society. Series B (Methodological)*.

Tiao, L. C., Klein, A., Seeger, M. W., Bonilla, E. V., Archambeau, C., and Ramos, F. (2021). BORE: Bayesian optimization by density-ratio estimation. In *International Conference on Machine Learning (ICML)*.

A Appendix

A.1 Experiments setting

A.1.1 BO setting. We used an internal BO implementation where expected improvement (EI) together with Mat'ern-52 kernel in the GP are used. The hyperparameters of the GP includes output noise, a scalar mean value, bandwidths for every input dimension, 2 input warping parameters and a scalar covariance scale parameter. The closest open-source implementations are GPyOpt using input warped GP [2] or AutoGluon BayesOpt searcher [3]. We maximize type II likelihood to learn the GP hyperparameters in our experiments.

A.1.2 Algorithm. We present the pseudo-code for the termination criterion in Algorithm 1.

A.1.3 Search spaces for cross-validation experiments. XGBoost (XGB) and RandomForest (RF) are based on scikit-learn implementations and their search spaces are listed in Table 1.

A.1.4 Datasets in cross validation experiments. We list the datasets that are used in our experiments, as well as their characteristics and sources in Table 2. For each dataset, we first randomly draw 20% as test set and for the rest, we use 10-fold cross validations for regression datasets and 10-fold stratified cross validation for classification datasets. The actual data splits depend on the seed controlled in our experiments. For a given experiment, all the hyperparameters trainings use the same data splits for the whole tuning problem. For the experiments without cross-validation, we use 20% dataset as validation set and the rest as training set.

Algorithm 1 BO for HPO with cross-validation and automatic termination

Require: Model $\mathcal{M}_\gamma$ parametrized by $\gamma \in \Gamma$, data $\{\mathcal{D}_1, \ldots, \mathcal{D}_k\}$ for k-fold cross-validation, acquisition function $\alpha(\gamma)$

Initialize $y_t^* = +\infty$ and $G_t = \{\}$
for $t = 1, 2, \ldots$ **do**
  Sample $\gamma_t \in \arg\max_{\gamma \in \Gamma} \alpha(\gamma)$
  for $i = 1, 2, \ldots, k$ **do**
    Fit the model $\mathcal{M}_{\gamma_t}(\cdot; \mathcal{D}_{-i})$, where $\mathcal{D}_{-i} = \cup_{j \neq i} \mathcal{D}_i$
    Evaluate the fitted model $y_t^i = \frac{1}{|\mathcal{D}_i|} \sum_{\mathrm{x}_i, \mathrm{y}_i \in \mathcal{D}_i} \ell(\mathrm{y}_i, \mathcal{M}_{\gamma_t}(\mathrm{x}_i, \mathcal{D}_{-i}))$
  end for
  Calculate the sample mean $y_t = \frac{1}{k} \sum_k y_t^i$,
  if $y_t \leq y_t^*$ **then**
    Update $y_t^* = y_t$ and the current best $\gamma_t^* = \gamma_t$
    Calculate the sample variance $s_{\mathrm{cv}}^2 = \frac{1}{k} \sum_i (y_t - y_t^i)^2$
    Calculate the variance estimate $\mathrm{Var}\hat{f}(\gamma_t^*) \approx \left(\frac{1}{k} + \frac{|\mathcal{D}_i|}{|\mathcal{D}_{-i}|}\right) s_{\mathrm{cv}}^2$ from Eq. (7)
  end if
  Update $G_t = G_{t-1} \cup \gamma_t$ and $y_{1:t} = y_{1:t-1} \cup y_t$
  Update σ_t, μ_t with **????**
  Calculate upper bound $\bar{r}_t := \min_{\gamma \in G_t} \mathrm{ucb}_t(\gamma) - \min_{\gamma \in \Gamma} \mathrm{lcb}_t(\gamma)$ for simple regret from Eq. (6)
  if the condition $\bar{r}_t \leq \sqrt{\mathrm{Var}\hat{f}(\gamma_t^*)}$ holds **then**
    terminate BO loop
  end if
end for
Output: γ_t^*

A.2 Detailed results

We first show the scatter plots of RTC and RYC scores for different automatic termination methods on HPO-Bench-datasets in Fig. 6 and the results on NAS-Bench-201 in Fig. 7.

A.2.1 Detailed numbers of RYC and RTC scores. We report detailed RYC scores and RTC scores of different HPO automatic termination methods for the experiments in the main text in Table 3, Table 4 and Table 5.

A.2.2 Correlation between validation and test metrics. In Fig. 8, we show the correlation between validation and test metrics of hyperparameters when tuning XGB and RF on tst-census dataset in Fig. 8.

A.2.3 The choice of β_t. High-probability concentration inequalities (aka confidence bounds) are important to reason about the unknown objective function and are used for theoretically grounded convergence guarantees in some (GP-UCB-based) BO methods (Srinivas et al., 2010; Ha et al., 2019; Kirschner et al., 2020; Makarova et al., 2021). There, β_t stands for the parameter that balances between exploration vs. exploitation and ensures the validity of the confidence bounds. The choice of β_t is then guided by the assumptions made on the unknown objective, for example, the objective being a sample from a GP or the objective having the bounded norm in RKHS (more agnostic case used in Section 3).

[2] `https://github.com/SheffieldML/GPyOpt`
[3] `https://github.com/awslabs/autogluon`

Table 1: Search spaces description for each algorithm.

tasks	hyperparameter	search space	scale
XGBoost	n_estimators	$[2, 2^9]$	log
	learning_rate	$[10^{-6}, 1]$	log
	gamma	$[10^{-6}, 2^6]$	log
	min_child_weight	$[10^{-6}, 2^5]$	log
	max_depth	$[2, 2^5]$	log
	subsample	$[0.5, 1]$	linear
	colsample_bytree	$[0.3, 1]$	linear
	reg_lambda	$[10^{-6}, 2]$	log
	reg_alpha	$[10^{-6}, 2]$	log
RandomForest	n_estimators	$[1, 2^8]$	log
	min_samples_split	$[0.01, 0.5]$	log
	max_depth	$[1, 5]$	log

dataset	problem_type	n_rows	n_cols	n_classes	source
openml14	classification	1999	76	10	openml
openml20	classification	1999	240	10	openml
tst-hate-crimes	classification	2024	43	63	data.gov
openml-9910	classification	3751	1776	2	openml
farmads	classification	4142	4	2	uci
openml-3892	classification	4229	1617	2	openml
sylvine	classification	5124	21	2	openml
op100-9952	classification	5404	5	2	openml
openml28	classification	5619	64	10	openml
philippine	classification	5832	309	2	data.gov
fabert	classification	8237	801	2	openml
openml32	classification	10991	16	10	openml
openml34538	regression	1744	43	-	openml
tst-census	regression	2000	44	-	data.gov
openml405	regression	4449	202	-	openml
tmdb-movie-metadata	regression	4809	22	-	kaggle
openml503	regression	6573	14	-	openml
openml558	regression	8191	32	-	openml
openml308	regression	8191	32	-	openml

Table 2: Datasets used in our experiments including their characteristics and sources.

In our experiments, we follow the common practice of scaling down β_t which is usually used to improve performance over the (conservative) theoretically grounded values (see e.g., Srinivas et al. (2010); Kirschner et al. (2020); Makarova et al. (2021)). Particularly, throughout this paper, we set $\beta_t = 2\log(|\Gamma|t^2\pi^2/6\delta)$ where $\delta = 0.1$ and $|\Gamma|$ is set to be the number of hyperparameters. We then further scale it down by a factor of 5 as defined in the experiments in Srinivas et al. (2010). We provide an ablation study on the choice of β_t in Fig. 9.

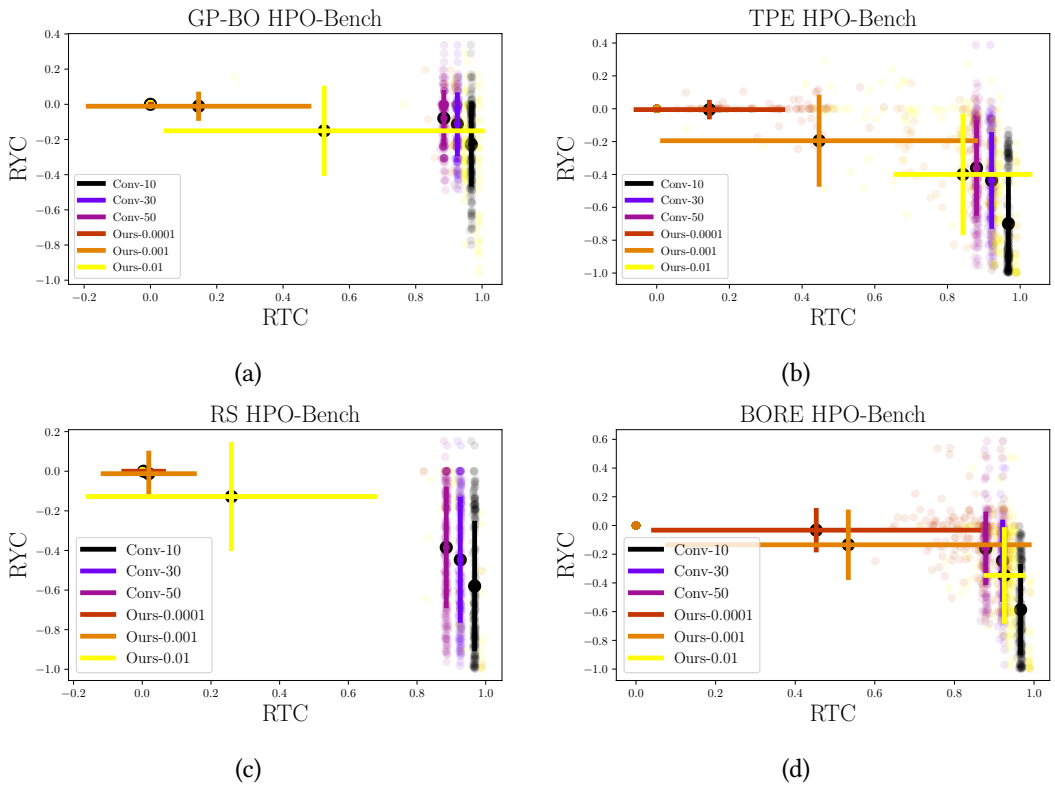


Figure 6: Fig. (a) - (d), the mean and standard deviation of RYC and RTC scores for considered automatic termination methods on HPO-Bench datasets using GP based BO (GP-BO), Random Search (RS), TPE and BORE optimizers. The mean value is shown in the big dot and the standard deviation is shown as error bar in both dimensions.

	RTC		RYC	
algo	RF	XGB	RF	XGB
Conv_10	0.840	0.841	-0.031	-0.051
Conv_30	0.686	0.666	-0.022	-0.026
Conv_50	0.498	0.504	-0.015	-0.021
EI_1e-08	0.896	0.850	-0.057	-0.052
EI_1e-12	0.895	0.779	-0.055	-0.047
EI_1e-16	0.893	0.718	-0.052	-0.045
PI_0.0001	0.898	0.875	-0.059	-0.059
PI_1e-08	0.895	0.814	-0.055	-0.052
PI_1e-12	0.894	0.739	-0.055	-0.044
Ours_0.21	0.318	0.144	-0.004	-0.003
Ours_0.5	0.580	0.224	-0.013	-0.006

Table 3: RTC and RYC scores for early stopping methods in cross validation benchmarks.

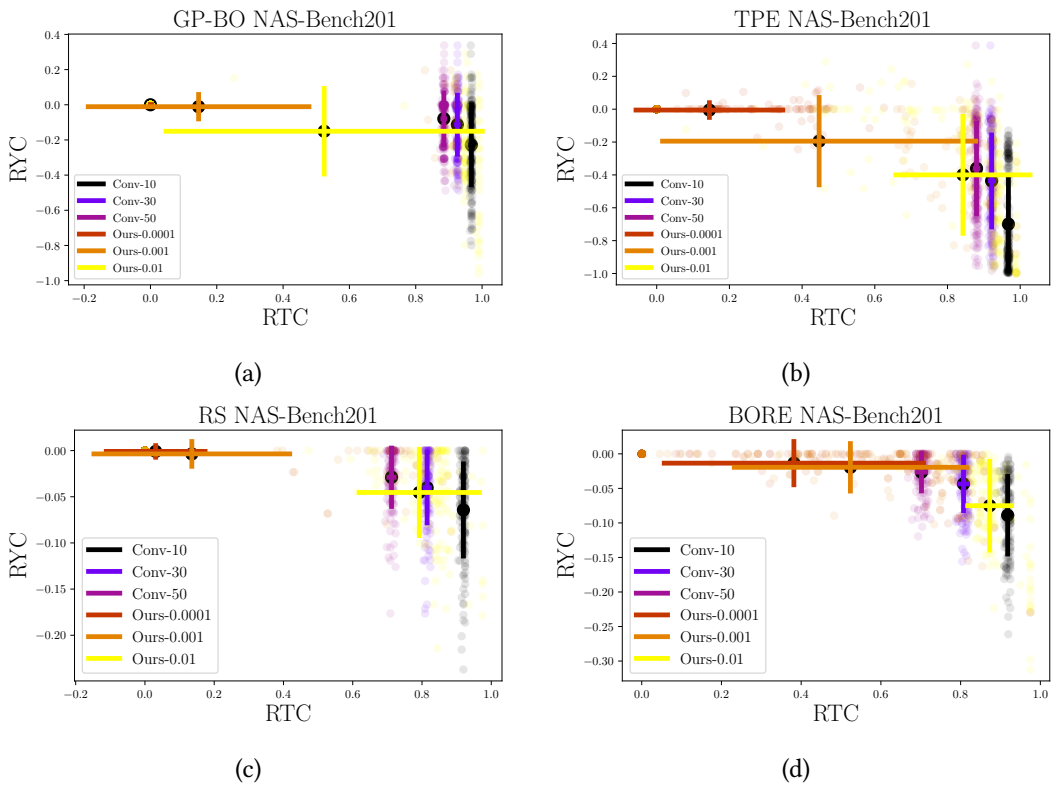


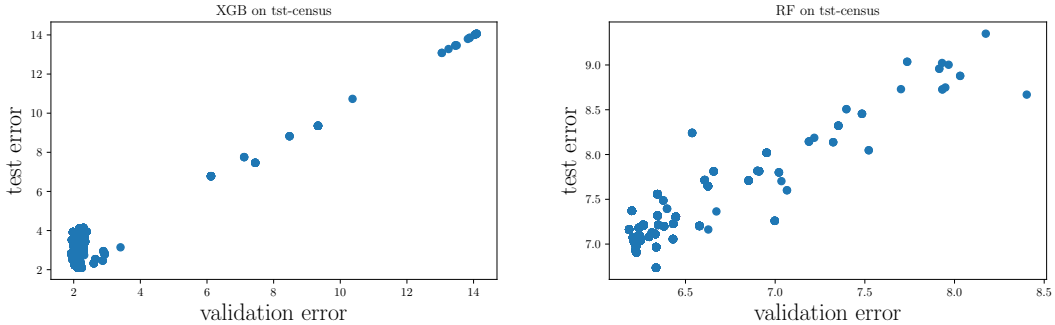
Figure 7: Fig. (a) - (d), the mean and standard deviation of RYC and RTC scores for considered automatic termination methods on NAS-Bench-201 datasets using GP based BO (GP-BO), Random Search (RS), TPE and BORE optimizers. The mean value is shown in the big dot and the standard deviation is shown as error bar in both dimensions.

	RTC				RYC			
dataset	naval	parkinsons	protein	slice	naval	parkinsons	protein	slice
Conv_10	0.943	0.947	0.946	0.942	-0.605	-0.582	-0.117	-0.432
Conv_30	0.826	0.837	0.837	0.840	-0.064	-0.235	-0.021	-0.119
Conv_50	0.748	0.729	0.734	0.747	-0.038	-0.107	-0.008	-0.058
Ours_0.0001	0.790	0.018	0.198	0.822	-0.041	-0.012	-0.005	-0.072
Ours_0.001	0.910	0.038	0.271	0.934	-0.220	-0.031	-0.018	-0.281
Ours_0.01	0.941	0.901	0.906	0.953	-0.498	-0.378	-0.071	-0.466

Table 4: RTC and RYC scores for early stopping methods in HPO-Bench.

	RTC			RYC		
dataset	ImageNet	cifar10	cifar100	ImageNet	cifar10	cifar100
Conv_10	0.880	0.889	0.888	-0.034	-0.098	-0.097
Conv_30	0.612	0.611	0.606	-0.010	-0.019	-0.036
Conv_50	0.372	0.361	0.372	-0.004	-0.006	-0.014
Ours_0.0001	0.274	0.311	0.519	-0.002	-0.008	-0.026
Ours_0.001	0.377	0.622	0.582	-0.005	-0.023	-0.033
Ours_0.01	0.837	0.902	0.879	-0.022	-0.106	-0.099

Table 5: RTC and RYC scores for early stopping methods in NAS-Bench-201.



(a) Training XGB on test-census dataset

(b) Training RF on test-census dataset.

Figure 8: We show validation error for training XGB (a) and RF (b) on `tst-census` dataset on the x-axis and test error on the y-axis. In the *low* error region, the validation metrics are not well correlated with the test metrics.

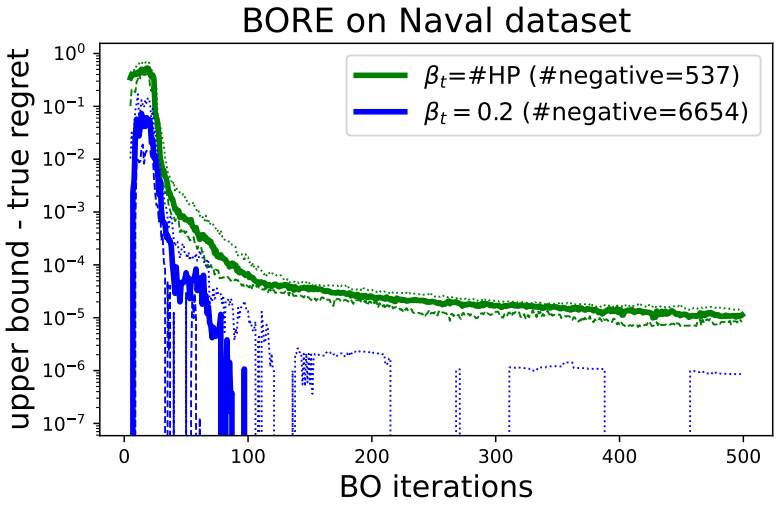


Figure 9: The differences between upper bound and true regret for every BO iterations when using BORE to tune an MLP on the Naval dataset. The number of negative differences (the upper bound is smaller than the true regret) are shown in the legend next to the two options for computing β_t.