Faster & More Reliable Tuning of Neural Networks: Bayesian Optimization with Importance Sampling

Setareh Ariafar[*] Google

Zelda Mariet Google

Dana Brooks Northeastern University

Jennifer Dy Northeastern University

Jasper Snoek Google

Abstract

Many contemporary machine learning models require extensive tuning of hyperparameters to perform well. A variety of methods, such as Bayesian optimization, have been developed to automate and expedite this process. However, tuning remains costly as it typically requires repeatedly fully training models. To address this issue, Bayesian optimization has been extended to use cheap, partially trained models to extrapolate to expensive complete models. This approach enlarges the set of explored hyperparameters, but including many low-fidelity observations adds to the intrinsic randomness of the procedure and makes extrapolation challenging. We propose to accelerate tuning of neural networks in a robust way by taking into account the relative amount of information contributed by each training example. To do so, we leverage importance sampling (IS); this significantly increases the quality of the function evaluations, but also their runtime, and so must be done carefully. Casting hyperparameter search as a multi-task Bayesian optimization problem over both hyperparameters and IS design achieves the best of both worlds. By learning a parameterization of IS that trades-off evaluation complexity and quality, our method improves upon validation error, in the average and worst-case, while using higher fidelity observations with less data. We show that this results in more reliable performance of our method in less wall-clock time across a variety of datasets and neural architectures.

[*]Work done while at Northeastern University.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

1 Introduction

The incorporation of more parameters and more data, coupled with faster computing and longer training times, has driven state-of-the-art results across a variety of benchmark tasks in machine learning. However, careful model tuning remains critical in order to find promising hyperparameters, architecture and optimization settings. This tuning requires significant experimentation, training many models, and is often guided by expert intuition, grid search, or random sampling. Such experimentation multiplies the cost of training, and incurs significant financial, computational, and even environmental costs.

Bayesian optimization (BO) offers an efficient alternative when the tuning objective can be modeled by a surrogate regression [Bergstra et al., 2011, Snoek et al., 2012], or when one can take advantage of related tasks [Swersky et al., 2013] or strong priors over problem structure [Swersky et al., 2014, Domhan et al., 2015]. BO optimizes an expensive function by iteratively building a relatively cheap probabilistic surrogate and evaluating a balanced combination of uncertain and promising regions (exploration *vs.* exploitation). In the context of neural network hyperparameter optimization, BO typically involves an inner loop of training a model given a hyperparameter configuration, and then evaluating validation error as the objective to be optimized. This inner loop is expensive and its cost grows with the size of the dataset. Training modern models even once may require training for days or weeks.

One strategy to mitigate the high cost of hyperparameter tuning is to enable the BO algorithm to trade off between the value of the information gained from evaluating a hyperparameter setting and the cost of that evaluation. [Swersky et al., 2013] and [Klein et al., 2016a] use multi-task BO, evaluating models trained on randomly chosen subsets of data to obtain more numerous, but noisier and less informative, evaluations.

The intrinsic randomness of BO is a less explored aspect which can significantly affect optimization performance [Lindauer et al., 2019, Bossek et al., 2020].

Sources of randomness include the size and selection of the initial design and modeling choices of the probabilistic surrogate. Sub-sampling data also introduces an additional source of noise, which is non-stationary relative to the subset size and data distribution.

To alleviate the high cost of tuning and control the randomness due to data sampling, we propose **I**mportance-based **B**ayesian **O**ptimization (IBO). Rather than defaulting to cheap evaluations, BO identifies situations where spending additional effort during training to obtain a higher fidelity observation is worth the incurred computational cost. To achieve this, IBO takes into account the training data and focuses the computation on more informative examples. Specifically, IBO models a distribution over the location of the optimal hyperparameter configuration, and allocates experimental budget according to cost-adjusted expected reduction in entropy [Hennig and Schuler, 2012]. Therefore, higher fidelity observations provide a greater reduction in entropy, albeit at a higher evaluation cost. At each training step, IBO allocates effort while training a network based on which training examples should be prioritized. IBO starts from treating the training examples uniformly while keeping track of the estimated impact of each example on the model, and switches to importance-weighted updates when this impact has become sufficiently large.

Although such higher-fidelity evaluations benefit the surrogate model and the extrapolation procedure, they also add significant overhead cost, which counts towards the total available computational budget. Balancing the cost of the training inner loop with BO outer loop is non-trivial; if done naively, the overall tuning procedure will be substantially slower. To address this issue, we adopt a multi-task Bayesian optimization formulation for IBO which dynamically adjusts a trade-off between the cost of training a network at higher fidelity and getting more but noisier evaluations (Fig. 1).

This approach allows us to obtain higher quality black-box function evaluations only when worthwhile, while controlling the average cost of black-box queries. Therefore, we are able to tune complex network architectures over challenging datasets in less time and with better results compared to the existing state-of-the-art BO methods . Tuning a ResNet on CIFAR-100, IBO improves the validation error by $\geq 4\%$ over the next best method; other baselines are not able to reach IBO's performance, even with additional computational budget. Furthermore, by prioritizing more informative training examples, IBO also improves the worst-case performance of hyperparameter tuning.

Contributions. We introduce a multi-task Bayesian optimization framework, IBO, which takes into account the contribution of each training point during the evaluation of a candidate hyperparameter. To do so, IBO optimizes the importance sampling (IS) tradeoff between quality and runtime while simultaneously searching hyperparameter space. We show empirically that the computational burden incurred by IS is more than compensated for by the principled search through hyperparameter space that it enables. We observe that IBO consistently improves over a variety of baseline BO methods. On more complex datasets, IBO converges significantly faster in wall-clock time than existing methods and furthermore reaches lower validation errors, even as other methods are given larger time budgets. Moreover, IBO improves upon worst-case error, proving to be reliable for tuning neural networks.

2 Related work

Several different methods have been proposed to accelerate the hyperparameter tuning process. Swersky et al. [2013] proposed Multi-Task Bayesian Optimization (MTBO), which performs surrogate cheap function evaluations on a randomly chosen small subset of training data which is then used to extrapolate the performance on the entire training set. Motivated by this work, Klein et al. [2016a] introduced Fabolas, which extends MTBO to also learn the sufficient size of training data. MTBO and Fabolas avoid costly evaluations by training on small datasets uniformly chosen at the beginning of each training round.

Another body of work involves modeling the neural network's loss as a function of both the hyperparameters and the inner training iterations. Then, the goal is to extrapolate the ultimate objective value and stop underperforming training runs early. Work such as [Swersky et al., 2014, Domhan et al., 2015, Dai et al., 2019, Golovin et al., 2017] falls under this category. These methods generally have to deal with the cubic cost of Gaussian processes — $O(n^3t^3)$, for n observed hyperparameters and t iterations. In practice, these methods typically apply some type of relaxation. For example, the freeze-thaw method [Swersky et al., 2014] assumes that training curves for different hyperparameters are independent conditioned on their prior mean, which is drawn from another global Gaussian process.

Moreover, an alternative approach to Bayesian optimization solves the hyperparameter tuning problem through enhanced random search. Hyperband [Li et al., 2016] starts from several randomly chosen hyperparameters and trains them on a small subset of data. Following a fixed schedule, the algorithm stops underperforming experiments and then retrains the remaining ones on larger training sets. Hyperband outperforms

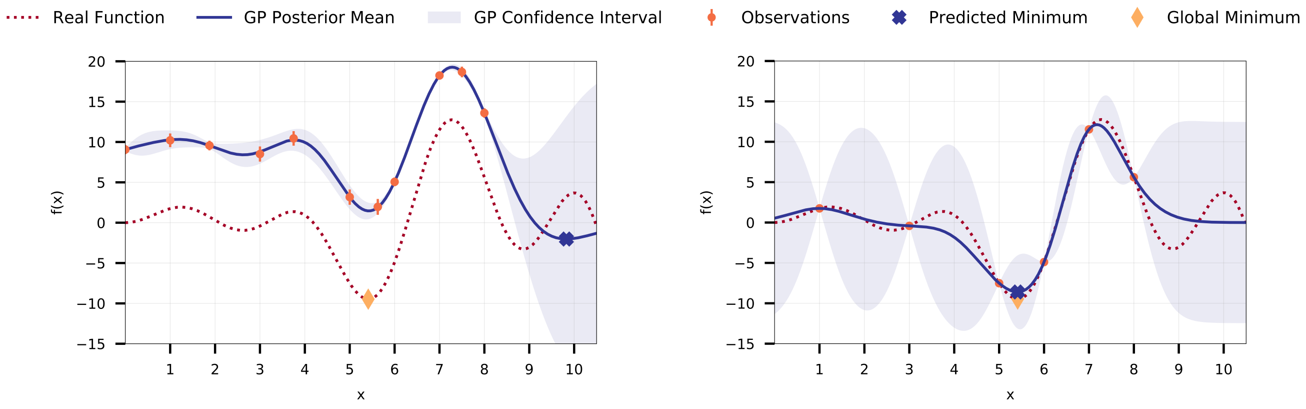

Figure 1: Motivating example; left: GP with many noisy points, right: GP with few noiseless points (motivated by minimizing the validation error, here the noisy observations have overestimated the real objective value). Compared to querying many points with lower fidelity (left), observing few points with higher fidelity (right) can significantly improve the predicted argmin but incurs a significant overhead cost; by learning to optimize the trade-off between the value and cost of high-fidelity estimates, we show in Section 4 that IBO achieves the best of both worlds.

standard BO in some settings, as it is easily parallelized and not subject to model misspecification. However, Hyperband's exploration is necessarily limited to the initial hyperparameter sampling phase: the best settings chosen by Hyperband inevitably will correspond to one of initial initializations, which were selected uniformly and in an unguided manner. To address this issue, several papers, including [Falkner et al., 2018, Wang et al., 2018, Bertrand et al., 2017], have proposed the use of Bayesian optimization to warm-start Hyperband and perform a guided search during the initial hyperparameter sampling phase.

Finally, IBO belongs to the family of multi-fidelity BO methods [Kandasamy et al., 2016, Forrester et al., 2007, Huang et al., 2006, Klein et al., 2016a], which take advantage of cheap approximations to the target black-box function. Of those methods, Fabolas [Klein et al., 2016a] focuses specifically on hyperparameter tuning, and is included as a baseline in our experiments. Fabolas uses cheap evaluations of the validation loss by training on a randomly sampled subset of the training data. Hence, both IBO and Fabolas depend directly on training examples to vary the cost of querying the black-box function; Fabolas by using fewer examples for cheap queries, whereas IBO uses the per-example contribution to training to switch to *costlier* queries.

Existing literature on tuning weighs all training examples equally and does not take advantage of their decidedly unequal influence. To the best of our knowledge, IBO is the first method to exploit the informativeness of training data to accelerate tuning, merging Bayesian optimization with importance sampling.

3 Importance sampling for BO

Bayesian optimization is a strategy for the global optimization of a potentially noisy, and generally non-convex, black-box function $f : \mathcal{X} \rightarrow \mathbb{R}$. The function f is presumed to be expensive to evaluate in terms of time, resources, or both. In the context of hyperparameter tuning, $\mathcal{X}$ is the space of hyperparameters, and $f(x)$ is the validation error or loss of a neural network trained with hyperparameter configuration x.

Given a set $\mathcal{D} = \{(x_i, y_i = f(x_i))\}_{i=1}^N$ of hyperparameters x_i and associated function evaluations y_i (which may be subject to observation noise), Bayesian optimization starts by building a surrogate model for f over $\mathcal{D}$. Gaussian processes (GPs), which provide a flexible non-parametric distribution over smooth functions, are a popular choice, because they provide tractable closed-form inference and facilitate the specification of a prior over the functional form of f [Rasmussen, 2003].

3.1 Surrogate model quality vs. computational budget

Given a zero-mean prior with covariance function k, the GP's posterior belief about the unobserved output $f(x)$ at a new point x after seeing data $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ is a Gaussian distribution with mean $\mu(x)$ and variance $\sigma^2(x)$ such that

$$\begin{aligned}(\mu(x), \sigma^2(x)) = \mathbf{k}(x)^T \left(\mathbf{K} + \sigma^2_{\text{noise}} I\right)^{-1} y,\\ k(x,x) - \mathbf{k}(x)^T \left(\mathbf{K} + \sigma^2_{\text{noise}} I\right)^{-1} \mathbf{k}(x)\end{aligned} \tag{1}$$

where $\mathbf{k}(x) = [k(x, x_i)]_{i=1}^N$, $\mathbf{K} = [k(x_i, x_j)]_{i,j=1}^N$, and σ^2_{noise} is the variance of the observation noise, that is,

$y_i \sim \mathcal{N}(f(x_i), \sigma^2_{\text{noise}})$. Given this posterior belief over the value of unobserved points, Bayesian optimization selects the next point x to query by solving

$$x = \operatorname*{argmax}_{x \in \mathcal{X}} \alpha(x \mid \mathcal{D}), \tag{2}$$

where $\alpha(\cdot)$ is the *acquisition function*, which quantifies the expected added value of querying f at point x, based on the posterior belief on $f(x)$ given by Eq. (1). Typical choices for the acquisition function α include entropy search (ES) [Hennig and Schuler, 2012] and its approximation predictive entropy search [Hernández-Lobato et al., 2014], knowledge gradient [Wu et al., 2017], expected improvement [Močkus, 1975, Jones et al., 1998] and upper/lower confidence bound [Cox and John, 1992, 1997].

Entropy search quantifies how much knowing $f(x)$ reduces the entropy of the distribution $\mathbb{P}[x^* \mid \mathcal{D}]$ over the location of the best hyperparameter x^*:

$$\alpha_{\text{ES}}(x \mid D) = \mathbb{E}_{y|x,\mathcal{D}}\Big[H\big(\mathbb{P}[x^* \mid \mathcal{D}]\big) \tag{3}$$

$$-H\big(\mathbb{P}[x^* \mid \mathcal{D} \cup \{x, y\}]\big)\Big], \tag{4}$$

where H is the entropy function and the expectation is taken with respect to the posterior distribution over the observation y at hyperparameter x.

The more accurate the observed values y, the more accurate the GP surrogate model (1). A more accurate surrogate model, in turn, defines a better acquisition function (2), and, finally, a more valuable Bayesian optimization outer loop with better predicted optima.

Crucially, hyperparameter tuning for neural networks is not an entirely black-box optimization setting, as we know the loss minimization framework in which neural networks are trained. We take advantage of this by allocating computational budget *at each SGD iteration*; based on the considered training points, IBO switches from standard SGD updates to the more computationally intensive importance sampling updates. This is the focus of the following section.

3.2 Importance sampling for loss minimization

The impact of training examples on one (batched) SGD iteration has received significant attention in machine learning [Needell et al., 2014, Schmidt et al., 2015, Zhang et al., 2017, Fu and Zhang, 2017]. For the purposes of IBO, we focus on importance sampling [Needell et al., 2014, Zhao and Zhang, 2015]. IS minimizes the variance in SGD updates;[1] however, IS is parameterized by the per-example gradient norm for the current weights of the neural network incurring a significant computational overhead.

[1] IS also benefits SGD with momentum [Johnson and Guestrin, 2018]. Although focusing on pure SGD, IBO also extends to certain SGD variants.

Specifically, let $g(w) = \frac{1}{m}\sum_{i=1}^m g_i(w)$ be the training loss, where m is the number of training points and g_i is the loss at point i. To minimize $g(w)$, SGD with IS computes estimate w_{t+1} of $w^* = \operatorname{argmin} g$ by sampling $i \in \{1, \dots, m\}$ with probability $p_i \propto \|\nabla g_i(w_t)\|$, then applying the update $w_{t+1} = w_t - \eta \frac{1}{mp_i} \nabla g_i(w_t)$ where η is the learning rate (standard SGD is recovered by setting $p_i = 1/m$). This update provably minimizes the variance of the gradient estimate, hence improving the convergence speed of SGD.

Since evaluating the per-example gradient norm $\|\nabla g_i(w_t)\|$ is prohibitively expensive, various surrogates have been suggested [Zhao and Zhang, 2015, Loshchilov and Hutter, 2015]. We leverage recent work by Katharopoulos and Fleuret [2018], which proposes a tractable upper-bound on $\|\nabla g_i\|$ as proxy for the true per-example gradient norm.

To further speed up IS, [Katharopoulos and Fleuret, 2018] introduce a *pre-sample batch size* hyperparameter B. At each SGD step, B points are first sampled uniformly at random, from which a batch of size $b \leq B$ is then sub-sampled. These b points are sampled either uniformly or, when the additional variance reduction justifies the incurred computation cost, they are sampled with importance sampling.

3.3 Multi-task BO for importance sampling

In [Katharopoulos and Fleuret, 2018], the authors state that their importance sampling algorithm is highly sensitive to the number B of the pre-sampled data points: indeed, B controls when to switch from uniform sampling to IS. We verify this empirically in section 4, showing that naively replacing standard SGD with the IS algorithm of Katharopoulos and Fleuret [2018] does not improve upon standard BO hyperparameter tuning. To maximize the utility of importance sampling, we instead opt for a multi-task BO framework, within which the search through hyperparameter space $\mathcal{X}$ is done in parallel to a second task: optimization over B.

Multi-task Bayesian optimization [Swersky et al., 2013] extends BO to share observations across multiple correlated tasks. This can be used *e.g.* to optimize an objective function over an expensive target task by extrapolating from related cheaper tasks. Here, our target task evaluates $f(x)$ using batched-SGD, where batches are sampled with IS from the entire training set; the cheaper related task $f(x)$ samples batches with IS from a uniformly sub-sampled set of B data points.

MTBO uses the entropy search acquisition function

(Eq. 3), and models an objective function over points $x \in \mathcal{X}$ and tasks $t \in \mathcal{T}$ via a multi-task GP [Journel and Huijbregts, 1978, Bonilla et al., 2008]. The covariance between two pairs of points and corresponding tasks is defined through a Kronecker product kernel:

$$k\big((x,t),(x',t')\big) = k_X(x,x') \cdot k_T(t,t'), \tag{5}$$

where k_X models the relation between the hyperparameters and k_T is the covariance between tasks.

For our case, the sub-sampling size B is the task variable while the optimal task sets B^* to the size of the entire training set. Let $f_n(x_i \mid B_i)$ denote the validation error value at hyperparameter x_i after n training iterations using IS with pre-sample size B_i. We define a multi-task GP that models $f_n(x_i \mid B_i)$ over the joint space of x and B. Additionally, following Snoek et al. [2012], we penalize the evaluation of any point (x, B) by the computational cost $c_n(x \mid B)$ of training a model for n SGD iterations at hyperparameter x with sub-sampling size B. This penalty guides the hyperparameter search towards promising yet relatively inexpensive solutions. We model the training cost $c_n(x \mid B)$ using a multi-task GP fitted to the log cost of observations $c_{i|n}$ that are collected during BO rounds while modifying the kernel on B to reflect that larger B increases training time. Our choice of kernel functions follows [Klein et al., 2016a] and a detailed description can be found in Appendix. B.

Our resulting acquisition function is thus:

$$\begin{aligned}\alpha_n(x,B) = \frac{1}{\mu(c_n(x \mid B))}\Big[&H(\mathbb{P}[x^* \mid B^*, \mathcal{D}_n]) \\ &-\mathbb{E}_y\big[H(\mathbb{P}[x^* \mid B^*, \mathcal{D}_n \cup \{x,B,y\}])\big]\Big],\end{aligned} \tag{6}$$

where $\mu(c_n(x \mid B))$ is the posterior mean of the GP modeling the training cost; as previously, $\mathbb{P}(x^* \mid B^*, \mathcal{D}_n)$ is the probability that x^* is the optimal solution at the target task B^* given data $\mathcal{D}_n = \{x_i, B_i, y_{i|n}, c_{i|n}\}_{i=1}^N$. Our algorithm is presented in Algorithm 1. The initialization phase follows the MTBO convention: we collect initial data at randomly chosen inputs x, and evaluate each hyperparameter configuration with a randomly selected B. `DoSGD` is the subroutine proposed by [Katharopoulos and Fleuret, 2018]; it determines if the variance reduction enabled by importance sampling is worth the additional cost at the current SGD iteration.

Algorithm 1 Importance-based BO

```
Obtain initial data $\mathcal{D}_n = \{x_i, B_i, y_{i|n}, c_{i|n}\}$
for $i = 1, \ldots, n_{\mathrm{BO}}$ do
    Fit multi-task GPs to $f_n$ and $c_n$ given $\mathcal{D}_n$
    $x, B \leftarrow \operatorname{argmax} \alpha_n(x, B \mid \mathcal{D}_n)$
    $\mathcal{M} \leftarrow$ model initialized with hyperparams $x$
    for $j = 1, \ldots, n$ do
        $S_B \leftarrow B$ uniformly sampled training points
        if DoSGD($\mathcal{M}, B, S_B$) then
            $\mathcal{M} \leftarrow$ IS_SGD($\mathcal{M}, S_B, x$)
        else
            $\mathcal{M} \leftarrow$ Vanilla_SGD($\mathcal{M}, S_B, x$)
        end if
    end for
    $y \leftarrow$ validation error of $\mathcal{M}$
    $c \leftarrow$ time used to train $\mathcal{M}$
    $\mathcal{D}_n \leftarrow \mathcal{D}_n \cup \{(x, B, y, c)\}$
end for
return $x^* \in \{x_i\}$ with best predicted error at $B^*$
```

Although IBO and Fabolas have a similar multi-task acquisition function, they are intrinsically different. Given a hyperparameter x, whereas Fabolas speeds up the evaluation of $f(x)$ by choosing one subset of training examples, IBO potentially uses the entire training data, reweighting points based on their relevance to the training task. Thus, each IBO iteration is slower than a Fabolas iteration. However, because IBO carries out a more directed search through hyperparameter space and queries higher fidelity evaluations, IBO requires less BO iterations — and hence potentially less time — to find a good configuration.

This difference in how data is sampled and used has downstream consequences: as Fabolas sample subsets of data of varying size, it introduces non-stationary noise into the observations modeled by the surrogate model. However, the typical GP formulation only incorporates a homogeneous noise term σ^2_{noise} in Eq. (1). Klein et al. [2016a] show that the sub-sampling noise is negligible for an SVM on MNIST, but we observe significant noise for more complex models and datasets (Fig. 4). By leveraging IS, IBO decreases its sensitivity to the randomness caused by data sub-sampling.

Our proposed algorithm can be extended in different ways. First, exploiting parallel computation is a desired criterion for recent hyperparameter optimization methods [Falkner et al., 2018], which IBO can satisfy by parallel estimation of the importance distribution at each round of training. Moreover, IBO can be extended to incorporate other optimization methods and importance criteria beyond SGD and per-example gradient norm. Particularly, the gist of IBO is to learn a cost-quality trade-off for leveraging higher-fidelity evaluations, which are enabled by weighting the more informative training examples. Although we defined the weights as per-example gradient norm to speed up SGD, these weights can be defined in accordance with other optimization algorithms. For example, Loshchilov and Hutter [2015] proposed to use per-example loss value to speed up Adam and AdaDelta. An interesting extension of IBO is to jointly optimize over the hyperparameters and the trade-off parameter as well as the type of inner optimization algorithm (SGD, Adam, AdaDelta, etc) and its corresponding weighting criterion of data points (per-example gradient norm, loss, etc), which remains an open avenue for future research.

4 Experiments

We evaluate our proposed method, IBO, on four benchmark tuning tasks: a feed-forward network on MNIST, a convolutional neural network (CNN) on CIFAR-10, a residual network on CIFAR-10, and a residual network on CIFAR-100. We include the following baselines: ES [Hennig and Schuler, 2012] and Fabolas [Klein et al., 2016a] as well as their IS-extended versions, ES-IS and Fabolas-IS, where training is performed with SGD-IS without a multitask formulation (see Appendix C).

ES-IS acts as an ablation test for IBO's multi-task framework, as it does not reason about the cost-fidelity tradeoff of IBO. Thus, we keep the training procedure for ES-IS and Fabolas-IS similar to IBO, switching to IS only if variance reduction is possible and using IS is advantageous (Alg. 1, lines $8-11$). We run all methods on a PowerEdge R730 Server with NVIDIA Tesla K80 GPUs (experiment in Appendix D) or on a DGX-2 server with NVIDIA Tesla V100 GPUs (rest).

Following Snoek et al. [2012], we marginalize over the GPs' hyperparameters using MCMC for all methods. We report performance as a function of wall-clock time, since the methods differ in per-iteration complexity (App. 5 reports results vs. iteration number). All initial design evaluations are counted towards the runtime. We measure the performance of each method by taking the predicted best hyperparameter values x^* after each BO iteration, then training a model with hyperparameters x^*, using the *entire* training set and vanilla SGD. We run each method five times unless otherwise stated, and report the median performance and 25^{th} and 75^{th} percentiles (mean and standard deviation results are included in Appendix F.1 for completeness). For brevity, additional implementation details and description of all hyperparameter ranges can be found in Appendix C and the results of the feed-forward network on MNIST in Appendix D. The code is available open-source.[2]

[2]https://github.com/SetarehAr/IBO.

4.1 CNN on CIFAR-10

We tune a convolutional neural network (CNN) using RMSProp on the CIFAR-10 dataset [Krizhevsky et al., 2009]. We fix an architecture of three convolutional layers with max-pooling, followed by a fully connected layer, in line with previous benchmarks on this problem [Falkner et al., 2018, Klein et al., 2016a, Dai et al., 2019]. Following Dai et al. [2019], we tune six hyperparameters: number of convolutional filters, number of units in the fully connected layer, batch size, initial learning rate, weight decay, and regularization weight.

All methods are run for 100 BO iterations and trained using $n = 50$ SGD epochs.

IBO, Fabolas and ES-IS exhibit the best performance (Fig. 2, left) but switch ranking over the course of time. However, after spending roughly half of the budget, IBO outperforms Fabolas and all other baselines, achieving the best final error with the lowest uncertainty. ES-IS shows that adding IS naively can improve upon ES; however, IBO outperforms both ES and ES-IS, confirming the importance of a multi-task setting that optimizes IS. Furthermore, simply adding IS during SGD is not guaranteed to improve upon any method: Fabolas-IS performs poorly w.r.t. Fabolas.

4.2 Residual Network on CIFAR-10

We next tune the a residual network trained on CIFAR-10. We follow the wide ResNet architecture in [Zagoruyko and Komodakis, 2016], and tune four hyperparameters: initial learning rate, weight decay, momentum and regularization weight. Following Klein et al. [2016a], all but the momentum are optimized over a log-scale search space.

We set $n = 50$ and multiply the learning rate by the weight decay after $n = 40$ epochs. Experimentally, we saw that $n = 50$ epochs is insufficient for the inner (SGD) optimization to converge on the ResNet architecture; this experiment evaluates BO in the setting where f is too computationally intensive to compute exactly. We ran all the methods using 80 BO iterations for Fabolas and Fabolas-IS and 50 iterations for the rest. This difference in budget iteration is to compensate for the different cost of training on a subset of data versus on the entire data.[3]

Consistently with previous results, Fabolas achieves the lowest error in the very initial stage, due to its cheap approximations (Fig. 2, middle). However, IBO

[3]For experiment D, we observed that keeping the BO iteration budget consistent is sufficient since the training costs are not very different. For experiment 4.1, we set this budget to 100, and since ResNet experiments are generally more costly, we stopped reporting the results once the first method exhausted its budget.

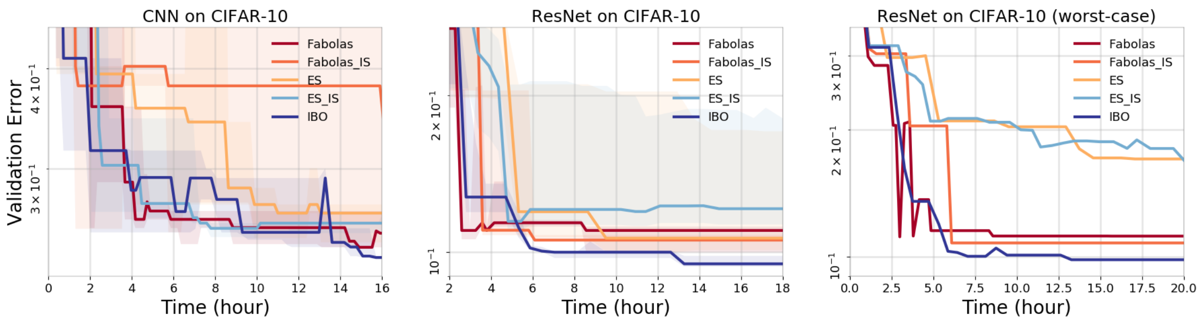

Figure 2: (Left) Best found error for CNN on CIFAR10 v.s. runtime. After around 9 hours, IBO outperforms the rest with a negligible variance while Fabolas-IS shows the weakest performance with a large uncertainty. (Middle) IBO finds the best error for ResNet on CIFAR-10 at 1/3 of the time budget and keep improving while Fabolas & Fabolas-IS achieves a minor improvement after 6 hours. (Right) IBO improves the worst-case error specially over that of ES & ES-IS, showing that simply augmenting BO with importance sampling is not robust.

quickly overtakes all other baselines, and attains a value that other methods cannot achieve with their entire budget consumption. Fabolas-IS also performs well, but suffers a large variance.

The ablation tests (ES-IS and Fabolas-IS) consistently have high variance, likely because these methods do not learn the optimal batch size for IS and opt for a random selection within the recommended range. In contrast, IBO specifically learns the batch size parameter which controls the cost-benefit trade off in IS and hence, enjoys better final results and lower variance. IBO offers the smallest uncertainty and improves the worst-case error among all the other methods demonstrating robustness, while ES and ES-IS suffer from a large variance (Fig. 2, right).

4.3 Residual Network on CIFAR-100

Finally, we tune the hyperparameters of a residual network trained on CIFAR-100. The architecture of the network, hyperparameters and ranges are the same as Section 4.2. We set $n = 200$ and multiply the learning rate by the weight decay every 40 epochs. For Fabolas and Fabolas-IS, a budget of 150 iterations is provided while the rest of the methods are given 50 iterations.

Clearly, IBO outperforms the rest of the methods after spending roughly 1/4 of the time budget (Fig. 3, left); Fabolas and ES-IS are the second best methods. Similar to the experiment 4.1, Fabolas-IS is outperformed by the other baselines, and once again incurs a large variance; This is likely because as a variant of Fabolas, Fabolas-IS depends on only one data subset for each training round. Interestingly, for Fabolas and Fabolas-IS, the additional budget does not cause an improvement in their performance. This is yet further evidence that for complex datasets, neither vanilla multi-task frameworks nor simple importance sampling is sufficient to gain the advantages of IBO. IBO improves upon the worst-case error after spending 1/4 of the time budget (Fig.3, right). From halfway of the time onwards, the largest error found by IBO is smaller than the smallest error of the other methods, highlighting the robustness of IBO for complex datasets.

Overall, both IBO and Fabolas were amongst the best performing methods in our experiments, with IBO consistently outperforming Fabolas. Table 1 shows the error of the final incumbent of Fabolas and its corresponding training data size learned by the algorithm. Although Fabolas allows for incorporating cheap approximations, it has achieved its best performance using 93% and 98% of the entire training data for ResNet on CIFAR-100 and CNN on CIFAR-10. Interestingly, for FCN on MNIST and ResNet on CIFAR-10, for which Fabolas achieved relatively lower errors, the required data is also smaller (30% and 46%). This suggests that using low-fidelity evaluations to speed up tuning, although advantageous in some scenarios, is not consistently effective, and that higher-fidelity evaluations, as used by IBO, provide a robust alternative.

5 Conclusion

Hyperparameter tuning using Bayesian optimization involves an expensive inner loop which repeatedly trains a model with new hyperparameters. Unlike prior work, which has scaled BO by using cheap evaluations to the black-box function, our method, IBO, takes the opposite approach, increasing the time spent obtaining higher-fidelity evaluations while requiring much fewer outer BO iterations.

Leveraging recent developments in importance sampling, IBO takes into account the contribution of each training point to decide whether to run vanilla SGD or

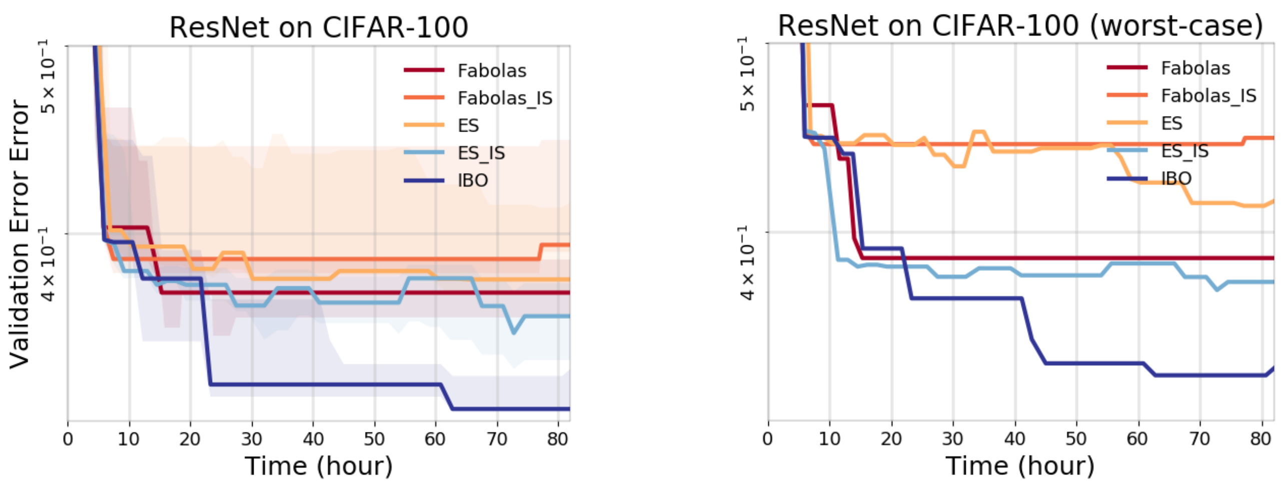

Figure 3: (Left) Best found error for ResNet on CIFAR-100 v.s. runtime. IBO outperforms all other methods roughly after spending 20 % of the time budget. Moreover, IBO is able to further improve after 60 hours while both Fabolas and Fabolas-IS are unable to progress after an early stage (around 15 hours). ES-IS exhibits the second best performance after IBO with 4-5 % margin, but with larger uncertainty. (Right) IBO achieves the best worst-case error among different runs followed by ES-IS while Fabolas-IS is outperformed by the rest.

Table 1: The best error found by Fabolas and the corresponding training data size of the incumbent. For CNN on CIFAR-10 and ResNet on CIFAR-100, Fabolas uses almost the entire training data while for the other two experiments, less than half of the data is shown to be sufficient.

Problem	Best Error	# of examples	% of examples
FCN (MNIST)	**0.06**	18034	**30%**
CNN (CIFAR10)	0.29	46533	93 %
ResNet (CIFAR10)	**0.11**	23136	**46 %**
ResNet (CIFAR100)	0.40	49121	98 %

a more involved but higher quality variant. Although this results in costlier network training, the additional precision obtained for the black-box estimates allows a more principled search through the hyperparameter space, significantly decreasing the amount of wall-clock time necessary to obtain a high-quality result. By opting for a multi-task parameterization of the problem, IBO learns to dynamically adjust the trade-off between training time and high precision, producing faster overall runtimes *as well as* better hyperparameters.

We show on four benchmark tasks of increasing complexity that IBO achieves the lowest error compared to the other methods, and scales gracefully with dataset and neural architecture complexity. When tuning a ResNet on CIFAR-100, IBO outperforms the rest by a significant margin, both as a function of runtime and number of training rounds.

References

James S. Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554. 2011.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016a.

Marius Lindauer, Matthias Feurer, Katharina Eggensperger, André Biedenkapp, and Frank Hutter. Towards assessing the impact of bayesian optimization's own hyperparameters. *arXiv preprint arXiv:1908.06674*, 2019.

Jakob Bossek, Carola Doerr, and Pascal Kerschke. Initial design strategies and their effects on sequential model-based optimization. *arXiv preprint arXiv:2003.13826*, 2020.

Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012.

Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets bayesian optimal stopping. In *International Conference on Machine Learning*, pages 1496–1506, 2019.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Elliot Karro, and D. Sculley, editors. *Google Vizier: A Service for Black-Box Optimization*, 2017.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*, 2016.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. *arXiv preprint arXiv:1807.01774*, 2018.

Jiazhuo Wang, Jason Xu, and Xuejun Wang. Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning. *arXiv preprint arXiv:1801.01596*, 2018.

Hadrien Bertrand, Roberto Ardon, Matthieu Perrot, and Isabelle Bloch. Hyperparameter optimization of deep neural networks: Combining hyperband with bayesian model selection. In *Conférence sur l'Apprentissage Automatique*, 2017.

Kirthevasan Kandasamy, Gautam Dasarathy, Junier B Oliva, Jeff Schneider, and Barnabas Poczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In *Advances in Neural Information Processing Systems*. 2016.

Alexander I.J. Forrester, András Sóbester, and Andy J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A*, 463(2088): 3251–3269, December 2007.

Deng Huang, T. T. Allen, W. I. Notz, and R. A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32:369–382, 2006.

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pages 918–926, 2014.

Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems*, pages 5267–5278, 2017.

Jonas Močkus. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

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

Dennis D Cox and Susan John. A statistical method for global optimization. In *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1241–1246. IEEE, 1992.

Dennis D. Cox and Susan John. Sdo: A statistical method for global optimization. In *in Multidisciplinary Design Optimization: State-of-the-Art*, pages 315–329, 1997.

Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems 27*, pages 1017–1025. Curran Associates, Inc., 2014.

Mark Schmidt, Reza Babanezhad, Mohamed Ahmed, Aaron Defazio, Ann Clifton, and Anoop Sarkar. Non-Uniform Stochastic Average Gradient Method for Training Conditional Random Fields. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research, pages 819–828, 2015.

Cheng Zhang, Hedvig Kjellstrom, and Stephan Mandt. Determinantal point processes for mini-batch diversification. In *UAI 2017*, 2017.

Tianfan Fu and Zhihua Zhang. CPSG-MCMC: Clustering-Based Preprocessing method for Stochastic Gradient MCMC. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 841–850. PMLR, 20–22 Apr 2017.

Peilin Zhao and Tong Zhang. Stochastic optimization with importance sampling for regularized loss minimization. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

Tyler B Johnson and Carlos Guestrin. Training deep models faster with robust, approximate importance sampling. In *Advances in Neural Information Processing Systems 31*, pages 7265–7275. Curran Associates, Inc., 2018.

Ilya Loshchilov and Frank Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. *arXiv preprint arXiv:1803.00942*, 2018.

Andre G Journel and Charles J Huijbregts. *Mining geostatistics*, volume 600. Academic press London, 1978.

Edwin V Bonilla, Kian M Chai, and Christopher Williams. Multi-task gaussian process prediction. In *Advances in neural information processing systems*, pages 153–160, 2008.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Supplementary material for fast bayesian optimization of machine learning hyperparameters on large datasets. 2016b.

Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pages 1492–1501, 2016.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.