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

Setareh Ariafar setar@google.com[*]
Google Research

Zelda Mariet zmariet@google.com
Google Research

Ehsan Elhamifar eelhami@ccs.neu.edu
Northeastern University

Dana Brooks brooks@ece.neu.edu
Northeastern University

Jennifer Dy jdy@ece.neu.edu
Northeastern University

Jasper Snoek jsnoek@google.com
Google Research

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 extremely costly as it typically requires repeatedly fully training models. To address this issue, Bayesian optimization methods have been extended to use cheap, partially trained models to extrapolate to expensive complete models. While this approach enlarges the set of explored hyperparameters, including many low-fidelity observations adds to the intrinsic randomness of the procedure and makes extrapolation challenging. We propose to accelerate tuning for neural networks in a robust way by taking into account the relative amount of information contributed by each training example. To do so, we integrate importance sampling with Bayesian optimization, which significantly increases the quality of the black-box function evaluations and their runtime. To overcome the additional overhead cost of using importance sampling, we propose a multi-task Bayesian optimization problem over both hyperparameters and importance sampling design, which achieves the best of both worlds. We show that our method improves tuning, in the average and worst-case, across a variety of neural architectures.

Keywords: Bayesian Optimization, Hyperparameter Tuning, Importance Sampling

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 good configurations of hyperparameters, architecture and optimization settings. This tuning requires training many models and is often guided by expert intuition, grid search, or

*. Work done while at Northeastern university.

random sampling. Such experimentation multiplies the cost of training, and incurs significant financial, computational, and even environmental costs (Strubell et al., 2019).

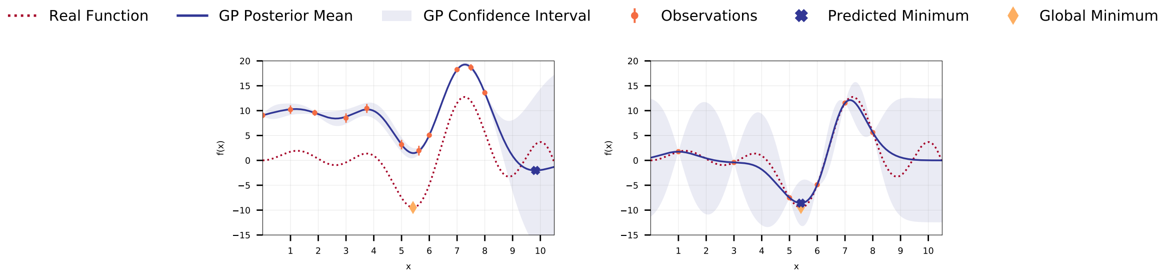
Bayesian optimization (BO) offers an efficient alternative when the tuning objective can be effectively modeled by a surrogate regression (Bergstra et al., 2011; Snoek et al., 2012), or when one can take advantage of related tasks (Swersky et al., 2013) or strong priors over problem structure (Swersky et al., 2014; Domhan et al., 2015). BO optimizes an expensive function by iteratively building a relatively cheap probabilistic surrogate and evaluating a carefully 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 cost of hyperparameter tuning is to enable the BO algorithm to trade-off between the value of the information gain of 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 data subsets to obtain more numerous, but noisier and less informative, queries.

The intrinsic randomness of BO is a less explored aspect which can significantly affect the optimization performance (Lindauer et al., 2019; Bossek et al., 2020). Sources of randomness include the size and selection strategy of the initial design and modeling choices of the surrogate. Subsampling data in multi-task BO also introduces an additional source of noise, which is non-stationary relative to the subset size and data distribution (Klein et al., 2016b).

Contributions. 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 using random sampling, 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 underlying 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 budget according to cost-adjusted expected reduction in entropy. Therefore, higher fidelity observations provide a greater reduction in entropy, albeit at a higher evaluation cost.

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 the outer BO loop is a non-trivial task; 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, achieving the best of both worlds.



(a) GP with many noisy points (b) GP with few noiseless points

Figure 1: Motivating example: compared to querying many points with lower fidelity (a), observing few points with higher fidelity (b) can significantly improve the model's predicted argmin. However, obtaining higher-quality estimates incurs a large overhead cost; IBO learns to optimize the trade-off between the value and cost of obtaining high-fidelity estimates.

2. Bayesian Optimization with Importance Sampling

Bayesian optimization is a an efficient method for the global optimization of a potentially noisy, and generally non-convex, black-box functions [1]. BO has shown to be effective in tuning neural networks by decreasing the total number of required trained models (Snoek et al., 2012). However, training each neural network model remains expensive which, in turn, can limit the final performance of BO. To address this issue, recent Bayesian optimization methods (Swersky et al., 2013; Klein et al., 2016a) decrease the cost of evaluating each hyperparameter set by training on subsets of data points. To maintain the low cost, the training data is typically subsampled once pre-training in a random fashion. As a result of lowering the per-evaluation cost, these methods enlarge the set of explored hyperparameters but at the same time, they have to rely on noisier and less informative *lower-fidelity* evaluations. Here, we propose a Bayesian optimization method to speed up tuning of neural networks following an orthogonal approach, that is using fewer number of more informative and less noisier *higher-fidelity* evaluations. We are motivated by the belief that the more accurate the evaluations, the more accurate the GP surrogate model, which in turn defines a better acquisition function and a more valuable BO outer loop.

To achieve higher-fidelity evaluations, we leverage SGD with importance sampling since it has shown to speed up training of neural networks with appropriate parameter values (Katharopoulos and Fleuret, 2018) . However, one of the main drawbacks of importance sampling is being extremely sensitive to its main parameter, the number B of the pre-sampled data points: indeed, B controls when to switch from uniform sampling to importance sampling. As we show in the experiments, naively replacing standard SGD with the IS algorithm of (Katharopoulos and Fleuret, 2018) does not improve upon standard BO hyperparameter tuning. Thus, to maximize the utility of importance sampling, we propose to use a multi-task BO framework to simultaneously search through the hyperparameter space $\mathcal{X}$ and parameterization of importance sampling, B.

To minimize the validation error over B with BO, we might need to train on large values of B for exploration purposes. This implies evaluating the importance distribution over a

1. See Appendix A.1 and A.2 for a brief review on Bayesian optimization and SGD with IS.

large number of data points *at each SGD iteration*, which is computationally inefficient. To address this issue, we extend BO to share the validation error across multiple correlated values of B by levering a multi-task BO (MTBO) framework (Swersky et al., 2013).

We model the validation error over hyperparameters $x \in \mathcal{X}$ and subsampling batch sizes (i.e., tasks) $B \in \{b, \dots, |\mathbf{D}|\}$ via a multi-task GP (Journel and Huijbregts, 1978; Bonilla et al., 2008), where b is the mini-batch size and $\mathbf{D}$ denotes the training data. Let k_X models the relation between the hyperparameters and k_B is the covariance between tasks. The covariance between two pairs of hyperparameters and corresponding subsampling batch sizes is defined through a Kronecker product kernel:

$$k\big((x, B), (x', B')\big) = k_X(x, x') \cdot k_B(B, B'). \tag{1}$$

Let $y_{i|n} = f_n(x_i \mid B_i)$ denote the validation error value at hyperparameter x_i after n training iterations using IS with subsampling size B_i. We define a multi-task GP that models $f_n(x \mid B)$ over the joint space of x and B. Following Snoek et al. (2012), we penalize the evaluation of any point (x_i, B_i) by the computational cost $c_n(x_i \mid B_i)$ of training a model for n SGD iterations at hyperparameter x_i with subsampling size B_i. This penalty guides the hyperparameter search towards promising yet relatively inexpensive solutions. We model the training cost $c_{i|n} = c_n(x \mid B)$ using a multi-task GP fitted to the log cost of observations $c_{i|n}$ that are collected during BO 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. E. Our resulting acquisition function is thus:

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

where $\mu(c_n(x \mid B))$ is the posterior mean of the GP modeling the training cost; as previously, $\mathbb{P}(x^* \mid B = |\mathbf{D}|, \mathcal{D}_n)$ is the probability that x^* is the optimal solution at the target task $B = |\mathbf{D}|$ 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.

3. Experiments

We evaluate our proposed method, IBO, on four benchmark hyperparameter 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 multi-task formulation (see Appendix F.)

ES-IS acts as an ablation test for IBO's multi-task framework, as it does not reason about the cost-fidelity trade-off 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 or on a DGX-2 server with NVIDIA Tesla V100 GPUs.

Following Snoek et al. (2012), we marginalize over the hyperparameters of GP using MCMC. We report performance as a function of runtime, 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 For brevity, additional implementation details and description of all hyperparameter ranges can be found in Appendix F and the results of the feed-forward network on MNIST and CNN on CIFAR-10 in Appendix G and H.

Algorithm 1 Importance-based BO

```
Obtain initial data $\mathcal{D}_n = \{x_i, B_i, y_{i|n}, c_{i|n}\}$
**for** $i = 1, \ldots, n_{\text{BO}}$ **do**
  Fit multi-task GPs to $f_n$ and $c_n$ given $\mathcal{D}_n$
  $x, B \leftarrow \text{argmax}\, \alpha_n(x, B \mid \mathcal{D}_n)$
  $\mathcal{M} \leftarrow$ 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 $B = |\mathbf{D}|$
```

3.1 Residual Network on CIFAR-10

We tune a ResNet on CIFAR-10 with a 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 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 SGD to converge on ResNet; 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 data subsets v.s. the entire data.

Consistently with previous results, Fabolas achieves the lowest error in the very initial stage, due to its cheap approximations (Fig. 2, right). However, IBO 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 demonstrating robustness, while ES and ES-IS suffer from a large variance (Fig. 4, right, in Appendix H).

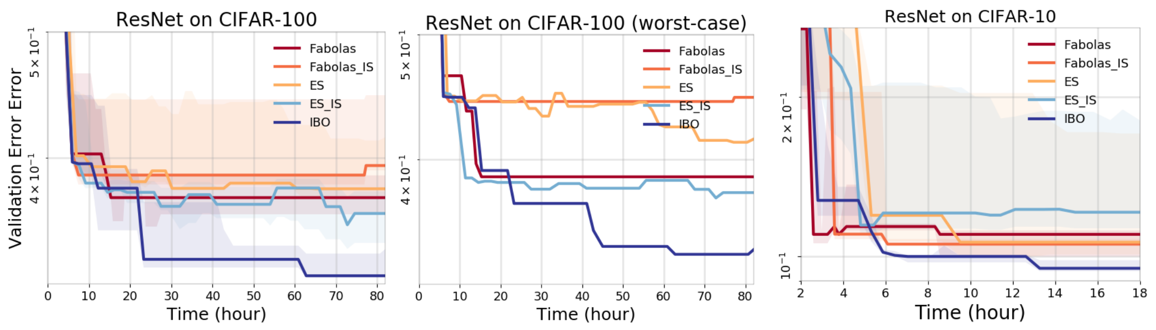


Figure 2: (Left & Middle) IBO outperforms all other methods for ResNet on CIFAR-100 roughly after spending 1/5 of the time budget and further improves while both Fabolas(-IS) are unable to progress after an early stage. IBO achieves the best worst-case error among different runs. (Right) IBO finds the best error for ResNet on CIFAR-10 at 1/3 of runtime and further improves while Fabolas(-IS) achieves a minor improvement.

3.2 Residual Network on CIFAR-100

Next, we tune a ResNet on CIFAR-100. The architecture of the network, hyperparameters and ranges are the same as §3.1. 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 BO 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. 2, left); Fabolas and ES-IS are the second best methods. Similar to the experiment H, 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 (Fig. 5, top left). 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.2, middle). From halfway of the time onwards, the largest error found by IBO is still smaller than the smallest error of the other methods, highlighting the robustness of our method for complex datasets.

4. Conclusion

Hyperparameter tuning involves repeatedly training a model with new hyperparameters. Prior BO methods scale up tuning by using cheap noisy training evaluations to enlarge the set of explored hyperparameters. Our method (IBO) takes the opposite approach by obtaining higher-fidelity evaluations while requiring to explore much fewer hyperparameters. To do so, IBO takes into account the contribution of each training point to decide whether to run vanilla SGD or a more involved but higher quality variant. Although this results in costlier training loops, IBO learns to dynamically adjust the trade-off between training time and high precision, producing faster overall runtimes *as well as* better hyperparameters.

Acknowledgments

We would like to acknowledge support for this project from NIH grant R01CA199673 from NCI.

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.

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.

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.

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

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.

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.

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.

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

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.

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

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

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.

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.

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.

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.

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

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

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.

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.

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

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

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.

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.

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

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

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.

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

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.

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.

Emma Strubell, Ananya Ganesh, and Andrew Mccallum. Energy and policy considerations for deep learning in nlp. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

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.

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.

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.

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

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

Appendix A. Problem Settings & Review

A.1 Bayesian optimization

Bayesian optimization is a strategy for the global optimization of a potentially noisy, and generally non-convex, black-box function $f : \mathcal{X} \to \mathbb{R}$. 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 of a neural network trained with hyperparameters x.

Given a set $\mathcal{D} = \{(x_i, y_i = f(x_i))\}_{i=1}^N$ of hyperparameter configurations 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).

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

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

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 (hyperparameter set) x to query by solving

$$x = \operatorname*{argmax}_{x \in \mathcal{X}} \alpha(x \mid \mathcal{D}), \tag{3}$$

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. (2). 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 hyperparameters x^*:

$$\alpha_{\text{ES}}(x \mid D) = \mathbb{E}_{y|x,\mathcal{D}}\Big[H\big(\mathbb{P}[x^* \mid \mathcal{D}]\big) - H\big(\mathbb{P}[x^* \mid \mathcal{D} \cup \{x, y\}]\big)\Big], \tag{4}$$

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

A.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). Particularly, a popular method to maximize the convergence speed of SGD is importance sampling; IS minimizes the variance in SGD updates;[2] however, at each SGD iteration, IS requires to evaluate the per-example gradient norm for the current weights of the network over the entire training data incurring a significant computational overhead.

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 examples and g_i is the loss at point i. To minimize $g(w)$, SGD with importance sampling iteratively computes estimate w_{t+1} of $w^* = \operatorname{argmin} g$ by sampling $i \in \{1, \ldots, m\}$ 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, which in turn improves the convergence speed of SGD.

Since evaluating the per-example gradient norm $\|\nabla g_i(w_t)\|$ is prohibitively expensive, different surrogates have been proposed (Zhao and Zhang, 2015; Loshchilov and Hutter, 2015). Particularly, Katharopoulos and Fleuret (2018) uses a tractable upperbound on $\|\nabla g_i\|$ as proxy for the per-example gradient norm. Estimating this proxy is at least one order of magnitude faster than the gradient norm estimation. However, it still requires a pass through the entire training data at each SGD iteration incurring high computational cost.

To address this issue, Katharopoulos and Fleuret (2018) introduce a *subsampling batch size* parameter B. At each SGD iteration, B points are first sampled randomly, from which a batch of size $b \leq B$ is then subsampled. These b points are sampled either uniformly or, when the additional variance reduction (which is a function of B) justifies the incurred computation cost, with importance sampling.

Appendix B. Related Work

Several different methods have been proposed to accelerate the hyperparameter tuning process. Swersky et al. (2013) proposed Multi-Task Bayesian Optimization, 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 randomly chosen pre-training.

Another body of related 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 and forecast 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 by Swersky et al. (2014) assumes that training curves for different hyperparameter configurations are independent conditioned on their prior mean, which is drawn from another global GP.

2. IS also benefits SGD with momentum (Johnson and Guestrin, 2018). Although focusing on pure SGD, IBO also extends to certain SGD variants.

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 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.

Appendix C. Subsampling Noise Analysis

As mentioned in Section 1, many multi-task Bayesian optimization methods, including Fabolas, treat the training data size as a an additional hyperparameter and use the performance of one subset of training data to extrapolate to the performance over the entire dataset. To prevent bias, these methods shuffle the data before each subsampling step. This shuffling, as well as the fact that these methods rely on *one* subset of data for each round of training, adds further *subsampling noise* to the observations; this noise is non-stationary relative to the subset size Klein et al. (2016a,b).

Typically, GPs in the context of BO consider a homogeneous noise which is added to the diagonal of the Gram matrix (see Eq. (2)) and is estimated either with maximum likelihood or MCMC. Klein et al. (2016b) show that the subsampling noise is negligible relative to this homogeneous noise for an SVM on MNIST. Here, we further investigate this for neural networks. We consider a ResNet on CIFAR-10 with four hyperparameter (see 3.1 for details). We drew 10 hyperparameter sets from a Latin hypercube design and divided the training size $\in [128, 50000]$ into 5 linearly spaced intervals.

Given a hyperparameter set x_i and the subset size s_j, we trained on x_i with 5 different randomly chosen training subsets with size s_j. Let $y_k(x_j, s_i)$ denote the validation error of each such run. We estimated the subsampling noise as

$$\sigma^2_{\text{subsampling}}(x_i, s_j) = \frac{1}{5}\sum_{k=1}^{5}(y_k(x_i, s_j) - \mu_{i,j})^2,$$

where $\mu_{i,j} = 1/5 \sum_k y_k(x_i, s_j)$. Moreover, using the collected data, we have estimated the noise of GP using maximum likelihood and MCMC assuming a horseshoe prior with length scale 0.1. Fig. 3 shows the mean and the std (over all hyperparameter set) of the estimated noises given each subset size (mean and std are taken over the results of all hyperparameter sets at a fixed subset size). For the smallest and largest subset sizes (s=128 and s=50000), the GP noise has mostly captured the subsampling noise, since the validation error results are more consistent (*i.e.*, given a very small data subset, the error is likely to be high). However, for intermediate subset sizes, the GP noise has significantly underestimated the subsampling noise. This shows that selected training examples can significantly affect the GP model and

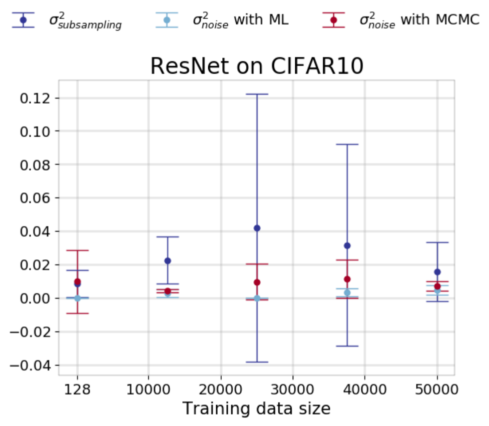


Figure 3: Estimated subsampling noise and GP noise on the collected validation errors using US. The subsampling noise is larger than the GP noise particularly for intermediate sizes.

hence negatively impact BO performance. Hence, although relying on one randomly chosen subset of data decreases the training cost, it may also significantly hinder optimization. On the other hand, spending additional effort to select more informative training examples, as done by IBO, can benefit the overall multiround optimization process.

Appendix D. Discussion

Although IBO and Fabolas have a similar multi-task acquisition function and both subsample the training data, they are intrinsically different. Given a hyperparameter x, Fabolas speeds up the evaluation of $f(x)$ by choosing one subset of training examples, while IBO potentially uses the entire training data, reweighting training data 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. (2). Klein et al. (2016a) show that the subsampling noise is negligible for an SVM on MNIST, but we observe significant noise for more complex models and datasets (Fig. 3). By leveraging IS, IBO decreases its sensitivity to the randomness caused by data subsampling.

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

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 %

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.

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.

Appendix E. Multi-task Kernel Design

We define the multi-task kernel for the GP that models $f_n(x_i \mid B_i)$ as

$$k^{(f)}\big((x, B), (x', B')\big) = k_X^{(f)}(x, x') \cdot k_B^{(f)}(B, B'),$$

with the sub-task kernels defined as

$$\begin{aligned} k_X^{(f)}(x, x') &= \text{Matérn}_{5/2}(x, x') \\ k_B^{(f)}(B, B') &= (1 - B)^{\nu}(1 - B')^{\nu} + 1. \end{aligned} \tag{5}$$

We choose the covariance function

$$k^{(c)}\big((x, B), (x', B')\big) = k_X^{(c)}(x, x') \cdot k_B^{(c)}(B, B'),$$

where this time we modify the kernel on B to reflect that larger B increases training time:

$$\begin{aligned} k_X^{(c)}(x, x') &= \text{Matérn}_{5/2}(x, x'), \\ k_B^{(c)}(B, B') &= B^{\lambda} B'^{\lambda} + 1. \end{aligned} \tag{6}$$

Appendix F. Implementation Details

For IBO, we use task kernels $k_B^{(f)}$ (Eq. 5) and $k_B^{(c)}$ (Eq. 6), with kernel hyperparameters $\lambda = 1$ and $\nu = 2$. Following (Snoek et al., 2012), we marginalize over the GPs' hyperparameters using MCMC for all methods. To set the time budget, we fix a total number of BO iterations for each method; the time at which the fastest method completes its final iteration acts as the maximum amount of time available to any other method. All initial design evaluations also count towards the runtime; this slightly advantages non-IS methods, which have cheaper initializations.

We report the performance of each method as a function of wall-clock time, since the methods differ in per-iteration complexity (App. 5 reports results vs. iteration number). 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. Recall that for Fabolas, Fabolas-IS, and IBO, the incumbent x^* is the set of hyperparameters with the best predicted objective on the target task (*e.g.*, using the full training data for Fabolas).

We run each method five times unless otherwise stated, and report the median performance and 25^{th} and 75^{th} percentiles We have included the following baselines:

– **ES**: Bayesian optimization with the entropy search acquisition function (Hennig and Schuler, 2012),

– Hyperband: the bandits-based hyperparameter tuning method introduced in (Li et al., 2016),

– **ES-IS**: BO with entropy search; inner optimization is performed using IS. For each black-box query, we draw the presample size B uniformly at random from $\{2, \ldots, 6\}\times$ batch size as prescribed in (Katharopoulos and Fleuret, 2018); B is constant during the n rounds of SGD.

– **Fabolas-IS**: Fabolas, training with SGD-IS. For this method, a fraction s of the training data is uniformly chosen as in Fabolas, but training is performed with SGD-IS. The pre-sample batch size B is the randomly uniformly sampled in $\{2, \ldots, 6\}\times$ batch size.

All methods are initialized with 5 hyperparameter configurations drawn from a Latin hypercube design. For IBO, we evaluate each configuration on the maximum value of its target task B. For Fabolas, (Klein et al., 2016a) suggest initializing by evaluating each hyperparameter on an increasing series of task values. This aims to capture the task variable's effect on the objective. However, we empirically observed that following an initial design strategy similar to IBO's, *i.e.*, evaluating each hyperparameter on the maximum target value s, worked better in practice for both Fabolas and Fabolas-IS. This is the method we use in our experiments; App. J includes results for both initialization schemes.

For IBO, Fabolas-IS and ES-IS, we reparameterize the pre-sample size B as $B = b \times s_B$. As was recommended by (Katharopoulos and Fleuret, 2018), we set $s_B \in [2, 6]$. For Fabolas-IS, if B is larger than the training subset size, we use the entire subset to compute the importance distribution.

In experiment H, following (Dai et al., 2019), we tune six hyperparameters: number of convolutional filters $n_c \in \{128, \ldots, 256\}$, number of units in the fully connected layer

$n_u \in \{256, \ldots, 512\}$, batch size $b \in \{32, \ldots, 512\}$, initial learning rate $\eta \in [10^{-7}, 0.1]$, weight decay $\beta \in [10^{-7}, 10^{-3}]$, and regularization weight $\upsilon \in [10^{-7}, 10^{-3}]$

In experiments 3.1 and 3.2, we tune four hyperparameters: initial learning rate $\eta \in [10^{-6},\ 1]$, weight decay $\beta \in [10^{-4},\ 1]$, momentum $\omega \in [0.1, 0.999]$ and L_2 regularization weight $\upsilon \in [10^{-6},\ 1]$. Following Klein et al. (2016a), all but the momentum are optimized over a log-scale search space. ES and Fabolas variations are run using RoBO. For importance sampling, we used the code provided by Katharopoulos and Fleuret (2018).

Appendix G. Feed-forward Neural Network on MNIST

Our first experiment is based on a common Bayesian optimization benchmark problem (Falkner et al., 2018; Domhan et al., 2015; Hernández-Lobato et al., 2016). We tune a fully connected neural network using RMSProp on MNIST (LeCun, 1998). The number of training epochs n and the number of BO rounds are set to 50. We tune six hyperparameters: number of hidden layers, number of units per layer, batch size, learning rate, weight decay, and dropout rate (see App. G).

Given the well-known straightforwardness of the MNIST dataset, we do not expect to see significant gains when using importance sampling during training. Indeed, we see that all methods perform similarly after exhausting their BO iteration budget, although Fabolas does reach a low validation error slightly earlier on, since training on few data points is sufficient.

Per BO iteration (Fig. 5, last row), IBO is amongst the best performing methods gaining higher utility compared to the others. However, since performing importance sampling is expensive, IBO's performance degrades over wall-clock time. After spending roughly 30% of the time budget (around two hours), Fabolas outperforms the other methods. This is expected since Fabolas utilizes cheap approximations by using training subsets. Although such approximations are noisy, we speculate that it does not significantly harm the performance, specially for simpler datasets and models such as a feed-forward network on MNIST.

We tune six hyperparameters: number of hidden layers $n_\ell \in \{1, \ldots, 5\}$, number of units per layer $n_u \in \{16, \ldots, 256\}$, batch size $b \in \{8, \ldots, 256\}$, initial learning rate $\eta \in [10^{-7}, \ldots,\ 10^{-1}]$, weight decay $\beta \in [10^{-7},\ 10^{-3}]$ and dropout rate $\rho \in [0,\ 0.5]$. Following Falkner et al. (2018), the batch size, number of units, and learning rate are optimized over a log-scale search space.

All methods are run for 50 BO iterations. The performance is averaged over five random runs and shown in the last row of Figure 5 (median with 25 and 75 percentiles over time and iteration budget).

Appendix H. CNN on CIFAR-10

We tune a convolutional neural network (CNN) (Krizhevsky et al., 2009) using RMSProp on the CIFAR-10 dataset. 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.

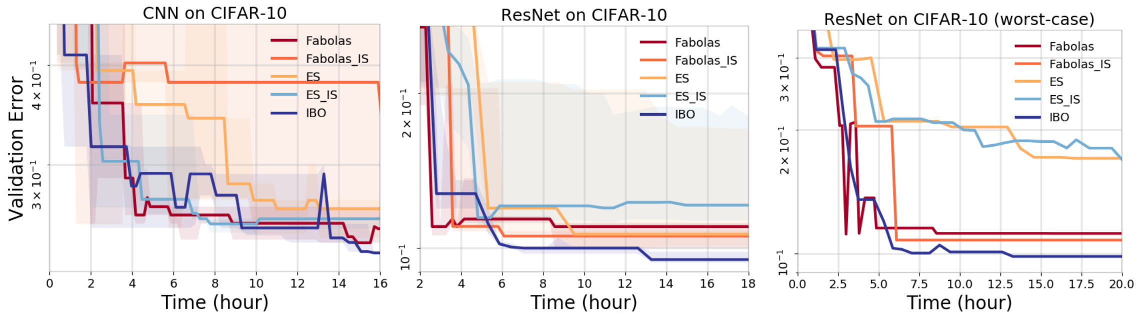


Figure 4: (Left) Best found error for CNN on CIFAR10 v.s. runtime. After around 9 hours, IBO outperforms the rest with a negligible variance while Fabolas-IS shows the weakest performance with a large uncertainty. (Middle) IBO finds the best error for ResNet on CIFAR-10 at 1/3 of the time budget and 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.

IBO, Fabolas and ES-IS exhibit the best performance (Fig. 4, 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 compared to Fabolas.

Appendix I. IBO scales with dataset and network complexity

IBO improves upon existing BO methods, moreso when tuning on large complex datasets and architectures. To illustrate, Figure 5 includes the results of all experiments over iteration budget (left column) and wall-clock time budget (right column). Moreover, the plots are sorted such that complexity of dataset and model architecture decreases along the rows; i.e., the most straight-forward problem, FCN on MNIST, lies in the bottom row and the most challenging experiment, ResNet on CIFAR100 is in the top row. In the iteration plots (left column), IBO is consistently amongst the best methods (lower curve denotes better performance), achieving high utility per BO iteration. However, since doing importance sampling is inherently expensive, the advantage of IBO over wall-clock time gradually manifests once the tuning becomes more challenging. Specifically, moving from the bottom to the top, as the complexity level of tuning increases, IBO starts to outperform the rest from and earlier stage and with an increasing margin over wall-clock time (right column).

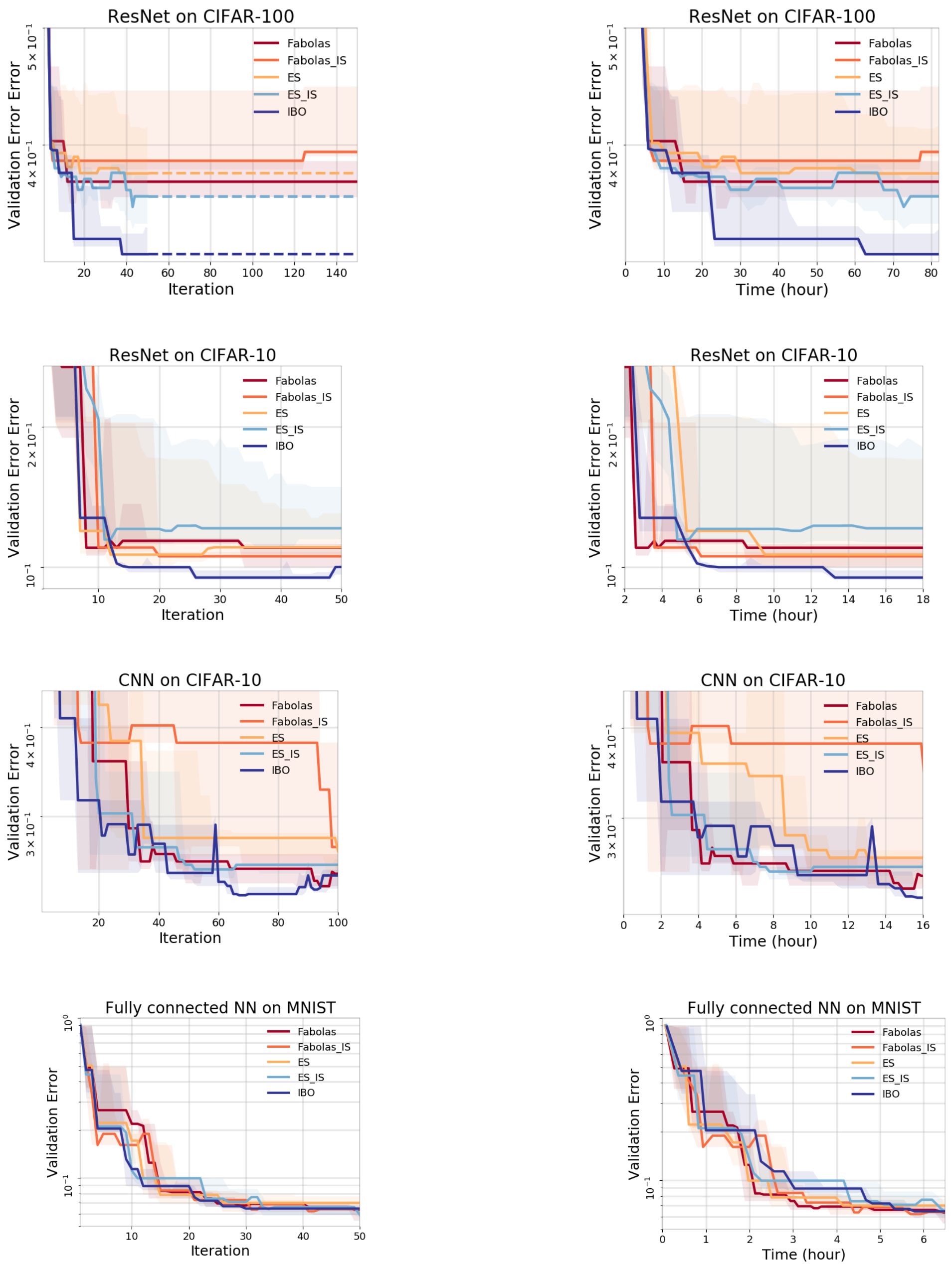


Figure 5: Average performance of all methods for all experiments as a function of both iteration budget (left column) and wall-clock time (right column). Each row represents one experiment such that the difficulty of tuning increases from the bottom row to the top i.e., the most straight-forward problem, FCN on MNIST, lies in the bottom row and the most challenging benchmark, ResNet on CIFAR100 is in the top row. In the iteration plots, IBO is consistently amongst the best methods (lower curve denotes better performance), achieving high utility per BO iteration. However, since doing importance sampling is inherently expensive, the advantage of IBO over wall-clock time gradually manifests once the tuning becomes more challenging. Specifically, IBO starts to outperform the rest earlier with an increasing margin over wall-clock time, the more difficult benchmarks become (from the bottom row to the top).

Appendix J. Initializing Fabolas

Conventionally, Bayesian optimization starts with evaluating the objective at an initial set of hyperparameters chosen at random. To leverage speedup in Fabolas, (Klein et al., 2016a) suggests to evaluate the initial hyperparameters at different, usually small, subsets of the training data. In our experiments, we randomly selected 5 hyperparameters and evaluated each on randomly selected training subsets with sizes $\{\frac{1}{128}, \frac{1}{64}, \frac{1}{32}, \frac{1}{16}\}$ of the entire training data. However, our experimental results show that Fabolas achieves better results faster if during the initial design phase, the objective evaluation use the entire training data. Figure 6 illustrates this point for CNN and ResNet on CIFAR-10. Fabolas with the original initialization scheme performs 20 evaluations (5 hyperparameters each evaluated at 4 budgets) where with the new scheme, Fabolas initializes with 5 evaluations (5 hyperparameters each evaluated at 1 budget). The plots show the mean results (with standard deviation) averaged over five and three runs for CNN and ResNet. Overall, the Fabolas with new initialization achieves better average performance.

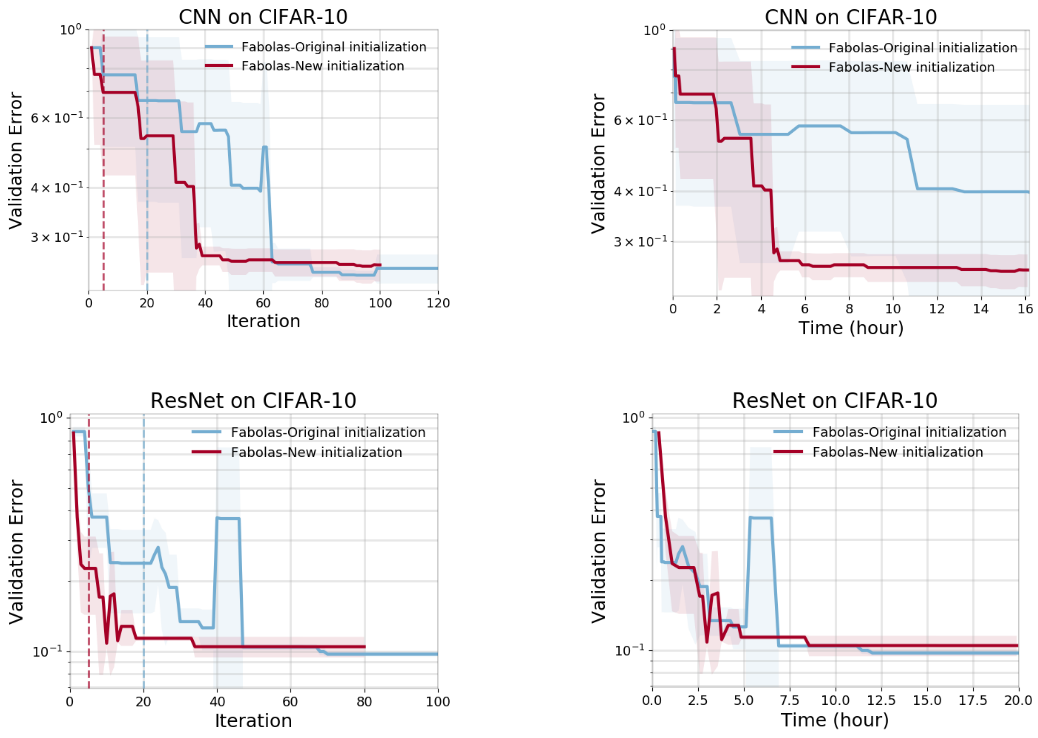


Figure 6: Comparison between two initialization schemes of Fabolas for CNN and ResNet on CIFAR-10. The dashed lines (left column) show the number of initial design evaluations for each method, immediately followed by the start of BO. We observe that with the new initial design scheme, Fabolas can potentially start progressing at a smaller iteration and a lower time, and achieve a reduced variance.