OPTIMAL DESIGNS OF GAUSSIAN PROCESSES WITH BUDGETS FOR HYPERPARAMETER OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The remarkable performance of modern deep learning methods depends critically on the optimization of their hyperparameters. One major challenge is that evaluating a single hyperparameter configuration on large datasets could nowadays easily exceed hours or days. For efficient sampling and fast evaluation, some previous works presented effective computing resource allocation schemes and built a Bayesian surrogate model to sample candidate hyperparameters. However, the model itself is not related to budgets which are set manually. To deal with this problem, a new Gaussian Process model involved in budgets is proposed. Further, for this model, an optimal design is constructed by the equivalence theorem to replace random search as an initial sampling strategy in the search space. Experiments demonstrate that the new model has the best performance among competing methods. Moreover, comparisons between different initial designs with the same model show the advantage of the proposed optimal design.

1 INTRODUCTION

In recent years, deep learning systems have reached remarkable performance on several important tasks and receive more and more attention (Lake et al., 2015; Silver et al., 2016; Wu et al., 2016). Decades of machine learning (ML) research ranging from learning strategies (Rumelhart et al., 1986; Bengio et al., 2013) to new architectures (LeCun et al., 1995; He et al., 2016) bring this huge success. Among these successful machine learning systems, almost all of them contain hyperparameters such as learning rates, batch sizes, or even model architectures that should be tuned carefully for performance. Nowadays, ML research has developed a new field, named automated machine learning (AutoML), which aims to automate the ML procedure by spending machine compute time instead of human research time. The most basic task in AutoML is to automatically set these hyperparameters to optimize performance.

The hyperparameter optimization problem can be formulated as:

$$x^* = \underset{x \in \mathcal{X}}{\operatorname{argmin}} f(x),$$

where x represents a hyperparameter configuration, $\mathcal{X}$ is a given search space of hyperparameters, and f is the target function.

Among many HPO algorithms for solving the optimization problem (Feurer & Hutter, 2019), Bayesian Optimization (BO) becomes a popular approach due to its sample efficiency (Snoek et al., 2012; Thornton et al., 2013; Snoek et al., 2015; Feurer et al., 2015). For a more detailed introduction to BO, we refer to the excellent tutorials by Brochu et al. (2010); Shahriari et al. (2015).

For improving the BO methods with a given search space, we have the following four aspects.

- Changing the initial design (Jones et al., 1998; Konen et al., 2011; Brockhoff et al., 2015; Zhang et al., 2019);
- Changing the surrogate model (Rasmussen, 2003; Hutter et al., 2011; Bergstra et al., 2011; Springenberg et al., 2016);
- Changing the acquisition function (Srinivas et al., 2010; Hennig & Schuler, 2012; Hernández-Lobato et al., 2014; Wang & Jegelka, 2017; Ru et al., 2018);

- Using the multi-fidelity methods (Thornton et al., 2013; Karnin et al., 2013; Jamieson & Talwalkar, 2016; Li et al., 2017; Falkner et al., 2018).

For multi-fidelity methods, it involves high-fidelity data obtained by more computing resources and low-fidelity data with less resources. For expensive high-fidelity models, however, even performing the number of simulations needed for fitting a surrogate may be too expensive. Inexpensive but less accurate low-fidelity models are often also available. Multi-fidelity models combine them in order to achieve accuracy at a reasonable cost.

In this paper we are concerned with multiple fidelities due to the practical applicability. In this situation, it is possible to define substantially cheaper versions of the objective function of interest, and the performance of low-fidelity roughly correlates with the performance of the full objective function. In the literature, many previous works are mainly based on the Successive Halving (SH) algorithm (Jamieson & Talwalkar, 2016). SH is proposed to identify the best configuration among K configurations. It evaluates all hyperparameter configurations, throws the worst half, doubles the budgets and repeats until one configuration left. This method uses a manually given budget for each iteration which is not desired in AutoML. Therefore, we propose a more proper surrogate model involved budgets, the corresponding acquisition function and initial design for this model. The main contribution of our work is as follows.

- A more accurate model for multi-fidelity data is presented. We add budgets as a factor into the model to measure the uncertainty caused by using low-fidelity data.
- The corresponding acquisition function is proposed. This function guides the next sampling strategy. Thus, the budget-related function will give the next dual sample, the configuration to be evaluated and the budgets it needs. This procedure helps to set budgets automatically.
- The optimal initial design for the model is constructed. This design aims to give a more accurate estimator of the model with the same number of initial samples. It helps to obtain a better performance or quick convergence.
- A theorem for judge whether a design is optimal.
- Simulation studies illustrate that the proposed new model with the corresponding optimal design outperforms other competing methods.

2 Gaussian Process with Budgets

The validation performance of machine learning algorithms can be modeled as a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X}$ is the search space of their hyperparameters. The HPO problem is then defined as finding $x^* = \text{argmin}_{x\in\mathcal{X}} f(x)$. In the literature, researchers always assume that the true performance $f(x)$ cannot be observed directly. Instead, we observe $y(x) = f(x) + \varepsilon$, where ε is a noise and follows $\mathcal{N}(0, \sigma^2_{noise})$ (Falkner et al., 2018). However, the difference between the observation and the true performance is not only caused by the noise, but also due to the used budgets. Consequently, it is natural to consider the model which involves budgets, $y(x, b) = f(x) + \varepsilon(b)$, where $\varepsilon(b) \sim \mathcal{N}(0, 1/b)$. This model makes $y(x, b)$ tend to $f(x)$ as $b \to \infty$. The most popular surrogate model of $f(x)$ is the Gaussian Process (GP) model. They are flexible, meaning that they can fit a wide variety of surfaces, from very simple to highly complex. Then, we can use a set of collected data $\{y(x_1, b_1), y(x_2, b_2), \ldots, y(x_n, b_n)\}$ to predict $f(x)$. For multi-fidelity cases, implementing high-accuracy training for the complex neural network with huge data can be costly. It is not proper to simply measure the uncertainty caused by budgets with a normal distribution since it is a major error term. For this purpose, we add another GP model to fit the error term,

$$y(x, b) = g(x) + h(x, b), \tag{1}$$

where $g(x)$ and $h(x, b)$ are realizations of two mutually independent Gaussian stochastic processes $\{G(x), x \in \mathcal{X}\}$ and $\{H(x, b), (x, b) \in \mathcal{X} \times (0, \infty)\}$. Further, we assume that $\mathbb{E}(G(x)) = f_1^\top(x)\beta_1$ and $\mathbb{E}(H(x, b)) = f_2^\top(b)\beta_2$, where β_1 and β_2 are unknown parameters, f_1 and f_2 are known regression functions and $\lim_{b\to\infty} f_2(b) = 0$. This model is also used in Tuo et al. (2014) for computer experiments. Different from theirs, for HPO problems, the covariance matrices are assumed

as follows,

$$\begin{aligned}&\mathrm{Cov}(G(x_1), G(x_2)) = \sigma^2 K_{\theta_1}(x_1, x_2)\\&\mathrm{Cov}(H(x_1, b_1), H(x_2, b_2)) = \tau^2 K_{\theta_2}(x_1, x_2)\exp\left(-\frac{1}{2}\frac{(b_1 - b_2)}{h^2}\right),\end{aligned} \tag{2}$$

where K_{θ_1} and K_{θ_2} are kernel functions and $\{\sigma, \tau, \theta_1, \theta_2, h > 0\}$ is a set of parameters for describing the correlation. Note that the second formulation of the covariance matrix in Tuo et al. (2014) is according to the simplest GP, the Brownian motion (Durrett, 2019). For computer experiments, when b increases, var$(H(x, b))$ needs to decrease to zero monotonically. However, in our cases, we just need to measure the correlation between two configurations with this RBF kernel. The limiting case $h \to 0$ is used in Ru et al. (2019) and showed its practicality. For the choice of the kernel function K, the Gaussian correlation family as the most common kernel is adopted,

$$K_{\theta_i}(x_1, x_2) = \exp\left\{-\sum_{j=1}^{p}\theta_{ij}(x_{1j} - x_{2j})^2\right\}, \tag{3}$$

where $(\cdot)_{ij}$ denotes the j-th entry of $(\cdot)_i$.

For this Gaussian Process with Budget model (GPB), we can use a standard Bayesian optimization procedure that is a sequential design strategy for finding the best hyperparameter configuration x. In hyperparameter optimization problems, the validation performance $f : \mathcal{X} \to \mathbb{R}$ of hyperparameters $x \in \mathcal{X}$ is our goal of minimization. In most cases, $f(x)$ does not admit an analytic form, which is approximated by the GPB model. The key difference between ours and other models such as Gaussian processes, random forests, or tree-structured Parzen estimator approach (Bergstra et al., 2011) is that we consider the budgets into the model and set the value of budgets automatically in iterations. Based on the data collected on the fly $D_n = \{(x_1, b_1, y_1), \ldots, (x_n, b_n, y_n)\}$, we sample next configuration (x_{n+1}, b_{n+1}) according to the acquisition function. The standard algorithmic procedure of BO is stated as follows.

1. Assume an initial surrogate model that is the GPB model in this work and take randomly initial samples (x_i, b_i, y_i) to estimate the model.
2. Compute an acquisition function $a : (\mathcal{X}, \mathcal{B}) \to \mathbb{R}$ which is the expected improvement (EI) in this work based on the current model.
3. Sample a batch of hyperparameter configurations and corresponding budgets based on the acquisition function.
4. Evaluate the configurations with corresponding budgets and refit the model.
5. Repeat Steps 2-4 until the stop condition is met.

In the literature of BO, acquisition functions (Step 2) (Wang & Jegelka, 2017) and batch sampling methods (Step 3) (González et al., 2016) are well studied, but how to choose proper initial samples for speeding up convergence (Step 1) remains largely open. Jones et al. (1998); Konen et al. (2011); Brockhoff et al. (2015); Zhang et al. (2019) used model-free initial designs Latin hypercube design or orthogonal array to improve the performance while we construct an optimal initial design for the particular GPB model in this work.

3 Optimal Initial Design

In this section we introduce some common criteria first to show the basic effect of optimal designs, minimizing the variance of estimating the key parameters β_1, β_2. This property can give a more accurate estimation for the GPB model after initialization to accelerate the convergence of the iteration. For this particular GPB model, we propose an equivalence theorem of the optimality and derive algorithms for constructing the optimal design.

3.1 Some Common Criteria

We talk about trace criteria under the normal linear model first, then we will show that these criteria can also be used in other models by using the Fisher information matrix. See Appendix B for more

criteria. Now we consider the following normal linear model,

$$y = \sum_{j=1}^{p} x_j \beta_j + \varepsilon, \quad \text{where } \varepsilon \sim N(0, \sigma^2). \tag{4}$$

Trace criterion Trace criterion is chosen when our experiment aims to minimize the total variance of the least squares estimates $\hat{\beta}$: $\text{var}\hat{\beta}_1 + \text{var}\hat{\beta}_2 + \ldots + \text{var}\hat{\beta}_p$, because $\text{Cov}(\hat{\beta}) = \sigma^2(X^\top X)^{-1}$ and $\text{var}\hat{\beta}_1 + \text{var}\hat{\beta}_2 + \ldots + \text{var}\hat{\beta}_p = \sigma^2 \text{trace}[(X^\top X)^{-1}]$.

Φ_ℓ-Optimality Now we apply the Φ_ℓ-optimality introduced by Kiefer (1974) to the Fisher information matrix M. Here $\Phi_\ell(M) = (\text{tr}M^\ell)^{1/\ell}, \ell \geq 0$. With different choices of ℓ, various criteria occur. As mentioned before, the most common examples of optimality criteria are

$$\begin{aligned}
&\Phi_0(M) = \det(M) \quad (\text{D}-\text{optimality}),\\
&\Phi_1(M) = \text{tr}(M) \quad (\text{A}-\text{optimality}),\\
&\Phi_\infty(M) = \text{the maximum eigenvalue of } M \quad (\text{E}-\text{optimality}).
\end{aligned}$$

3.2 Approximate Design

The optimization of discretized variables is more difficult than continuous variables. Hence, we derive the theory of the optimal approximate designs first. According to the approximate results, we propose algorithms for constructing the exact optimal design.

Now we review approximate designs defined as discrete probability measures with finite support points. The support points $x_1, \ldots, x_N$ of a design ξ indicate the hyperparameter configurations where observations are taken, and the corresponding weights $\omega_1, \ldots, \omega_N$ represent the probability weights at these support points. This approximate design ξ is denoted by $\{\mathcal{S}, \omega\}$ where $\mathcal{S} = \{x_1, \ldots, x_N\}$ and $\omega = \{\omega_1, \ldots, \omega_N\}$. See Kiefer (1974) for more details. By direct calculation, we have that the Fisher information $M(F_\xi) = F_\xi^\top \Phi^{-1} F_\xi$, where the i-th row of F_ξ is $(\omega_i f_1(x_i)^\top, \omega_i f_2(b_i)^\top)$ for $i = 1, 2, \ldots, N$ and the (i, j)-th entry of Φ is $\sigma^2 K_{\theta_1}(x_i, x_j) + \tau^2 K_{\theta_2}(x_i, x_j) \max(b_i, b_j)^{-h}$ for $i, j = 1, 2, \ldots, N$. We apply the Φ_ℓ-optimality to this Fisher information and obtain the optimization problem as follows,

Definition 1 *An approximate design ξ is locally Φ_ℓ-optimal if $\xi = \arg\max \Phi_\ell(M(F_\xi))$.*

3.3 Theoretical Result

Theorem 1 uses the Fréchet derivative $d(x, \xi) = \lim_{\varepsilon \to 0+} \varepsilon^{-1} \{\log \Phi_\ell[M((1-\varepsilon)\xi + \varepsilon\delta_x)] - \log \Phi_\ell(M(\xi))\}$, where δ_x is the Dirac measure on a single point x introduced by Silvey (2013) to derive an equivalence theorem for the GPB model.

Theorem 1 *An approximate design ξ is locally Φ_ℓ-optimal w.r.t. the objective function $\Phi_\ell(M(\xi))$ if and only if the Fréchet derivative holds that*

$$d(x, \xi) \triangleq \frac{\text{tr}((F_\xi^\top \Psi^{-1} F_\xi)^{\ell-1}(F_{\delta_x}^\top \Psi^{-1} F_\xi + F_\xi^\top \Psi^{-1} F_{\delta_x}))}{\text{tr}(F_\xi^\top \Psi^{-1} F_\xi)^\ell} - 2 \leq 0$$

for all $x \in \mathcal{X}$.

The brief proof is given in Appendix A. Theorem 1 helps us to judge whether a design is Φ_ℓ-optimal. Note that when the f_is are linear or quadratic functions, the optimization of $d(x, \xi)$ w.r.t. x is linear programming or quadratic programming respectively. Hence the maximum of $d(x, \xi)$ can be obtained easily.

Further, in practice, we cannot use this approximate optimal design to run real experiments. At this time, Theorem 1 is used as a criterion to find a new point x in the iterated construction algorithm. This construction will be discussed in the next section in detail.

3.4 ALGORITHMS FOR CONSTRUCTING EXACT OPTIMAL DESIGNS

This section proposes two algorithms for constructing exact optimal designs instead of turning an optimal approximate design into an exact design such as rounding procedure (Pukelsheim & Rieder, 1992).

According to Theorem 1, if a design ξ is not optimal, we can find a configuration x such that $d(x, \xi) > 0$. It reveals that we should move the current design along the direction of δ_x. This intuitiveness inspires the first algorithm described in Algorithm 1.

Algorithm 1 Iterated Construction

input Maximum iteration I; Sample size N; Number of factors p.
output The exact Φ_ℓ-optimal design X^*.

1: Initialize the design with a Latin hypercube $N \times p$ matrix $X^{(0)}$.
2: **for** $s = 0, 1, \ldots, I$ **do**
3: $\quad x = \arg\max d(x, X^{(s)})$.
4: $\quad$ **if** $d(x, X^{(s)}) \leq 0$ **then**
5: $\quad\quad$ break;
6: $\quad$ **end if**
7: $\quad X^{(s+1)} = X^{(s)} \cup x \backslash X_i^{(s)}$, where $X_i^{(s)}$ is the i-th row of $X^{(s)}$ and $i = \arg\max_j \Phi_\ell(M(X^{(s)} \cup x \backslash X_j^{(s)}))$.
8: **end for**
9: Output the optimal design $X^* = X^{(s)}$.

In each iteration, we find a best alternative configuration x according to Theorem 1 to replace the worst one among N configurations of the current design. Theorem 1 also gives a termination condition $d(x, X^{(s)}) > 0$.

Compared with general initialization, random search or grid search, this optimal design guarantees to have models with minimum variance. However, it may explore the space insufficiently.

For this purpose, we introduce Latin hypercube design (LHD) (McKay et al., 1979). It is a kind of uniform designs since it guarantees that it is uniform on each one-dimensional projection. Let $\boldsymbol{A} = (a_{ij})$ be an $N \times p$ Latin hypercube matrix in which each column is a permutation on $\{1, \ldots, N\}$ and all the columns are obtained independently. An ordinary Latin hypercube design $\boldsymbol{D}_0 = (d_{ij})$ of N runs in p factors is generated through $d_{ij} = (a_{ij} - u_{ij})/N$, for $i = 1, \ldots, N,\ j = 1, \ldots, p$, where the u_{ij} are independent random variables following $U[0, 1]$, d_{ij} is the value of factor j on the ith run, and the u_{ij} and the a_{ij} are mutually independent. When $\boldsymbol{D}_0$ is projected onto any one dimension, precisely one point falls within one of the N equally spaced intervals of $(0, 1]$ given by $(0, \frac{1}{N}], (\frac{1}{N}, \frac{2}{N}], \ldots, (\frac{n-1}{N}, 1]$. Its uniformity leads to exploring the space in a more balanced manner.

Consequently, we consider a Φ_ℓ-optimal LHD, i.e., the optimal design among all LHDs. For the construction of LHD, threshold algorithm is used because the main step of the construction is permutation, which can be viewed as integer programming.

Algorithm 2 is a kind of simulated annealing. The number of iterations works as the role of temperature. These two algorithms have the same terminate conditions, $\max_x d(x, X^{(s)}) \leq 0$, i.e., one contribution of Theorem 1. The other one is to guide the decision of next iterated design reflected in Line 7 of Algorithm 1 and Line 10 of Algorithm 2.

4 EXPERIMENTAL RESULTS

In this section, we compare different initial designs for the GBP model in synthetic experiments with several classic optimization functions. For more applications, auto data augment and neural architecture search (NAS) are applied.

Algorithm 2 Threshold Acceptance

input Maximum iteration I; Sample size N; Number of factors p.
output The exact Φ_ℓ-optimal LHD X^*.

```
Initialize the design with a Latin hypercube N × p matrix X^(0).
for s = 0, 1, ..., I do
    if max_x d(x, X^(s)) ≤ 0 then
        X* = X^(s);
        break;
    end if
    Pick one column randomly.
    Exchange its two rows randomly.
    Obtain a new LHD X^(s+1).
    if max_x d(x, X*) − max_x d(x, X^(s+1)) > −1/s then
        X* = X^(s+1);
    end if
end for
Output the optimal design X*.
```

4.1 Synthetic Experiments

Consider the objective function with effect of budgets:

$$f(x,h) = \left[\frac{\sin(20x)}{1+x} + 3x^3\cos(5x) + 10(x-0.5)^2 - 0.6\right]/2 + \frac{1}{b}\sin(15\pi(x+0.1))/5.$$

In the GPB model, the space of kernel parameters are set to $\theta_1, \theta_2 \in [40, 1000]$, $\tau^2/\sigma^2 \in [0.03, 0.07]$. Using the BO method and the proposed GPB model with different initial designs, we can see the effect of this step. For all designs, the sample size is set to be six. Let IC denote the design obtained by Algorithm 1 and TA by Algorithm 2. We compare them with random initial and LHD in two aspects. The result is given in Table 1. Standard deviation (Std.) of $\hat{\beta}$ is to measure the robustness of model estimation. IC has the best performance as its construction while TA is the second best since it is constrained in LHD. For integral loss, we can see that optimal LHD obtained by Algorithm 2 outperforms other designs since its uniformity from LHD and optimality from Φ_ℓ.

Table 1: The performance of different initial designs with $N = 6$. The integral loss is defined by the absolute difference between the true model with different b and the estimated model.

Initial design	Std. of $\hat{\beta}$	integral loss, b=1	integral loss, b=2/3	integral loss, b=0.5
Random	4.598	0.297 ± 0.121	0.285 ± 0.116	0.417 ± 0.388
IC	3.184	0.305 ± 0.051	0.306 ± 0.055	0.297 ± 0.065
TA	3.525	0.256 ± 0.064	0.238 ± 0.052	0.227 ± 0.029

For more objective functions listed in Table 5 in Appendix C, Table 2 shows that the proposed initial designs have much more robust estimation than Random Search does.

Table 2: The performance of different initial designs with different number of initial points.

	rosenbrock		sixhumpcamp	
Initial points	10	20	10	20
Random	0.4963	0.4700	0.5769	0.4326
IC	0.3918	0.3276	0.3789	0.3372
TA	0.3583	0.3570	0.4526	0.2983

4.2 Data Augmentation

Data augmentation (DA) is an effective technique to generate more samples from data by rotating, inverting or other operations for improving the accuracy of image classifiers. However, most

implementations are manually designed with a few exceptions. Cubuk et al. (2018) proposed a simple procedure called AutoAugment to automatically search for improved data augmentation policies. Unfortunately, it is very time-consuming, e.g., it takes 5000 GPU hours in searching procedure for CIFAR100 (Krizhevsky et al., 2009). More recently, Ho et al. (2019) and Lim et al. (2019) designed more efficient algorithms for this particular task.

In their search space, a policy consists of 5 sub-policies with each sub-policy consisting of two image operations to be applied in sequence. Additionally, each operation is also associated with two hyperparameters: (1) the probability of applying the operation, and (2) the magnitude of the operation. In total, there are 16 operations in the search space. Each operation also comes with a default range of magnitudes. These settings are described in Cubuk et al. (2018). For this problem, we need to tune two hyperparameters of each sub-policy and choose the best five sub-policies to form a policy. This is a natural HPO problem.

We search the data augmentation policy in the image classification tasks of CIFAR-10 and CIFAR-100 and follow the setting in AutoAugment (Cubuk et al., 2018) to search for the best policy on a smaller data set, which consists of $4,000$ randomly chosen examples, to save time for training child models during the augmentation search process. For the child model architecture, we use WideResNet-28-10 (28 layers - widening factor of 10) (Zagoruyko & Komodakis, 2016). The augmentation policy is combined with standard data pre-processing: on one image, we normalize the data in the following order, use the horizontal flips with 50% probability, zero-padding and random crops, augmentation policy, and finally Cutout with 16×16 pixels (DeVries & Taylor, 2017). Experiments use 4 parallel workers for 4 iterations with $20, 10, 10, 10$ configurations respectively. BO is run with budgets of 50 epochs for each configuration. The GPB model is run with alternative budgets of $1, 10, 20, 30, 40, 50$ epochs. For each sub-task, we use a SGD optimizer with a weight decay of 0.0005, momentum of 0.9, learning rate of 0.1. We use the found policies to train final models on CIFAR-10, CIFAR-100 with 200 epochs.

Table 3: The performance of different initial designs and the comparison with the GP model in DA.

Initial design	cifar10 acc	cifar10 budget	cifar100 acc	cifar100 budget
Random (GPB)	96.97 ± 0.127	1811.0 ± 143.380	81.07 ± 0.277	1750 ± 4.0
IC (GPB)	97.25 ± 0.076	1821.7 ± 131.556	81.23 ± 0.115	1994.5 ± 79.5
TA (GPB)	97.17 ± 0.134	1755.3 ± 183.603	81.17 ± 0.065	1769 ± 24
Random (GP)	96.84 ± 0.233	2500	80.52 ± 0.105	2500

Table 3 shows that for the proposed GPB model, we have competing results with much less budgets. The reason for uncertainty of budgets is that our model samples the value of budgets automatically while BO always use the maximum budgets for evaluation. For comparing accuracy, IC is the best on CIFAR-100 and TA is the second best, but the advantage is very weak. It is because for these public data, the training of neural network is often time-consuming and offsets most impact of the initial designs.

4.3 Neural Architecture Search

One crucial aspect of the deep learning development is novel neural architectures. Designing architectures manually is a time-consuming and error-prone process. Because of this, there is a growing interest in automated neural architecture search (NAS). Elsken et al. (2019) provided an overview of existing work in this field of research. We use the search space of DARTS (Liu et al., 2019) as an example to illustrate HPO methods on NAS. Particularly, their goal is to search for a cell as a basic unit. In each cell, there are N nodes forming a fixed directed acyclic graph (DAG). Each edge of the DAG represents an operation, such as skip-connection, convolution, max pooling, etc., weighted by the architecture parameter α. For the search procedure, the training loss and validation loss are denoted by L_{train} and L_{val} respectively. Then the architecture parameters are learned with the following bi-level optimization problem: $\min_\alpha \quad L_{val}(\omega^*(\alpha), \alpha),$ s.t. $\omega^*(\alpha) = \arg\min_\omega L_{train}(\omega, \alpha)$. Here, α are hyperparameters in the HPO framework. For evaluating α, we need to optimize its network parameters ω. It is usually time-consuming. We search the neural architectures in the image classification tasks of CIFAR10 and follow the settings of DARTS (Liu et al., 2019). The architecture parameter α determines two kinds of basic units: normal cell and reduction cell. We search the

network architecture in the image classification tasks of CIFAR-10 and CIFAR-100 on a smaller data set, which consists of $4,000$ randomly chosen examples, to save time for training child models during the network architecture search process.

Table 4: The performance of different initial designs and the comparison with the GP model in NAS.

Initial design	cifar10 acc	cifar10 budget	cifar100 acc	cifar100 budget
Random (GPB)	96.56 ± 1.685	1834 ± 119.2	79.625 ± 0.435	1840.5 ± 5.5
IC (GPB)	97.22 ± 0.258	1790.333 ± 98.324	82.655 ± 0.015	1970 ± 26
TA (GPB)	97.67 ± 0.385	1681.523 ± 134.523	82.920 ± 0.150	1793.5 ± 9.5
Random (GP)	96.99 ± 0.307	2500	81.755 ± 0.195	2500

Experiments use 4 parallel workers for 4 iterations with $20, 10, 10, 10$ configurations respectively. BO is run with budgets of 50 epochs for each configuration. The GPB model is run with alternative budgets of $1, 10, 20, 30, 40, 50$ epochs. For a sampled architecture parameter, we fix it in the training process of updating model parameters. A SGD optimizer is used with learning rate of 0.025, momentum of 0.9, weight decay of 0.0003, and a cosine learning decay with an annealing cycle. We use the found network architecture cell to build a 20-layer network and to train final models on CIFAR-10, CIFAR-100 with 200 epochs. Table 4 shows the similar results as Table 3 but more significantly, especially on CIFAR100.

The weakness of DARTS is that it has many skip-connect operations which is not preferred. Zela et al. (2020); Liang et al. (2019) reduced the number of skip-connect operations by early stopping. However, this issue disappears in the proposed method naturally which is depicted in Figure 1, because DARTS changes architecture parameters and network parameters in turn while we do not change the architecture during the training of network.

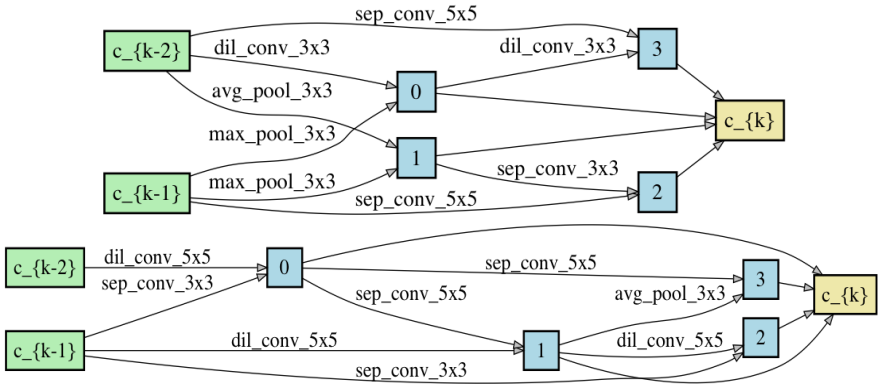

Figure 1: The architectures of normal cell (above) and reduction cell (bottom) learned by GPB on CIFAR10.

5 Conclusions

This work has proposed a new model called GPB to involve budgets in the model. This helps to automatically set the evaluation resources for each hyperparameter configuration and take the variance caused by different budgets into account. Further, for this particular model, an optimal design is constructed for estimating the model more accurately. The construction algorithm is derived by the equivalence theorem (i.e., Theorem 1) which can also be used to judge whether a design is optimal. Simulation studies support our theoretical results that optimal initial designs constructed by our algorithms can improve the model robustness and speed up the convergence. In the end, we apply the method to two popular machine learning problems, NAS and DA, and have the same conclusion as synthetic experiments. Note that the optimal designs proposed here rely on the particular model. Zhang et al. (2019) used orthogonal array and range analysis without assuming a model to make the initialization efficient. However it is quite simple and not compared to other uniform designs. How to construct an optimal model-free design is still an open problem that we leave for future work.

REFERENCES

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pp. 2546–2554, 2011.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Dimo Brockhoff, Bernd Bischl, and Tobias Wagner. The impact of initial designs on the performance of matsumoto on the noiseless bbob-2015 testbed: A preliminary study. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pp. 1159–1166, 2015.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019.

Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446, 2018.

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated Machine Learning*, pp. 3–33. Springer, 2019.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pp. 2962–2970, 2015.

Javier González, Zhenwen Dai, Philipp Hennig, and Neil Lawrence. Batch bayesian optimization via local penalization. In *Artificial intelligence and statistics*, pp. 648–657, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

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*, pp. 918–926, 2014.

Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population based augmentation: Efficient learning of augmentation policy schedules. *arXiv preprint arXiv:1905.05393*, 2019.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pp. 507–523. Springer, 2011.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pp. 240–248, 2016.

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.

Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pp. 1238–1246, 2013.

Jack Kiefer. General equivalence theory for optimum designs (approximate theory). *The annals of Statistics*, pp. 849–879, 1974.

Wolfgang Konen, Patrick Koch, Oliver Flasch, Thomas Bartz-Beielstein, Martina Friese, and Boris Naujoks. Tuned data mining: a benchmark study on different tuners. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pp. 1995–2002, 2011.

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

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019.

Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. *arXiv preprint arXiv:1905.00397*, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1eYHoC5FX`.

Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

Friedrich Pukelsheim and Sabine Rieder. Efficient rounding of approximate designs. *Biometrika*, 79 (4):763–770, 1992.

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

Binxin Ru, Michael A Osborne, Mark Mcleod, and Diego Granziol. Fast information-theoretic bayesian optimisation. In *International Conference on Machine Learning*, pp. 4384–4392, 2018.

Binxin Ru, Ahsan S Alvi, Vu Nguyen, Michael A Osborne, and Stephen J Roberts. Bayesian optimisation over multiple continuous and categorical inputs. *arXiv preprint arXiv:1906.08878*, 2019.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Samuel Silvey. *Optimal design: an introduction to the theory for parameter estimation*, volume 1. Springer Science & Business Media, 2013.

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

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pp. 2171–2180, 2015.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in neural information processing systems*, pp. 4134–4142, 2016.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *International Conference on Machine Learning*, pp. 1015–1022, 2010.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 847–855, 2013.

Rui Tuo, CF Jeff Wu, and Dan Yu. Surrogate modeling of computer experiments with different mesh densities. *Technometrics*, 56(3):372–380, 2014.

Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning*, pp. 3627–3635. JMLR. org, 2017.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

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

Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=H1gDNyrKDS`.

Xiang Zhang, Xiaocong Chen, Lina Yao, Chang Ge, and Manqing Dong. Deep neural network hyperparameter optimization with orthogonal array tuning. In *International Conference on Neural Information Processing*, pp. 287–295. Springer, 2019.

APPENDICES

A PROOF OF THEOREM 1

The approximate design ξ is a minimizer if and only if the Fréchet derivative

$$d(x,\xi) = \lim_{\varepsilon\to 0+} \varepsilon^{-1}\{\log\Phi_\ell[M((1-\varepsilon)F_\xi + \varepsilon F_{\delta_x})] - \log\Phi_\ell(M(F_\xi))\}$$

is non-positive for any $x \in \mathcal{X}$, where δ_x denotes a one-point design on x. To get the Fréchet derivative, we can calculate the Gâteaux derivative of $\log(\Phi_\ell(M(\cdot)))$ at ξ in the direction δ_x, i.e., $G(\xi,\delta_x) = \lim_{\varepsilon\to 0+}\varepsilon^{-1}[\log\Phi_\ell(M(F_\xi+\varepsilon F_{\delta_x})) - \log\Phi_\ell(M(F_\xi))]$. Note that $\log\Phi_\ell(M(\cdot)) = \log[\mathrm{tr}(M(\cdot)^\ell)]^{1/\ell} = \ell^{-1}\log\mathrm{tr}(M(\cdot)^\ell)$. The key calculation is as follows,

$$\begin{aligned}
\mathrm{tr}(M(F_\xi+\varepsilon F_{\delta_x})^\ell) &= \mathrm{tr}(((F_\xi+\varepsilon F_{\delta_x})^\top\Psi^{-1}(F_\xi+\varepsilon F_{\delta_x}))^\ell) \\
&= \mathrm{tr}((F_\xi^\top\Psi^{-1}F_\xi + \varepsilon F_{\delta_x}^\top\Psi^{-1}F_\xi + \varepsilon F_\xi^\top\Psi^{-1}F_{\delta_x})^\ell) + o(\varepsilon) \\
&= \mathrm{tr}((F_\xi^\top\Psi^{-1}F_\xi)^\ell + \ell\varepsilon(F_\xi^\top\Psi^{-1}F_\xi)^{\ell-1}(F_{\delta_x}^\top\Psi^{-1}F_\xi + F_\xi^\top\Psi^{-1}F_{\delta_x})) + o(\varepsilon) \\
&= \mathrm{tr}(M(\xi)^\ell + \ell\varepsilon(F_\xi^\top\Psi^{-1}F_\xi)^{\ell-1}(F_{\delta_x}^\top\Psi^{-1}F_\xi + F_\xi^\top\Psi^{-1}F_{\delta_x})) + o(\varepsilon).
\end{aligned}$$

Then, the molecule of the Gâteaux derivative is that

$$\begin{aligned}\log\frac{\Phi_\ell(M(\xi+\varepsilon\delta_x))}{\Phi_\ell(M(\xi))} &= \ell^{-1}\log\frac{\mathrm{tr}(M(\xi+\varepsilon\delta_x)^\ell)}{\mathrm{tr}(M(\xi)^\ell)}\\ &= \ell^{-1}\log\left(1+\ell\varepsilon\frac{\mathrm{tr}((F_\xi^\top\Psi^{-1}F_\xi)^{\ell-1}(F_{\delta_x}^\top\Psi^{-1}F_\xi+F_\xi^\top\Psi^{-1}F_{\delta_x}))}{\mathrm{tr}(F_\xi^\top\Psi^{-1}F_\xi)^\ell}\right)+o(\varepsilon)\\ &= \varepsilon\frac{\mathrm{tr}((F_\xi^\top\Psi^{-1}F_\xi)^{\ell-1}(F_{\delta_x}^\top\Psi^{-1}F_\xi+F_\xi^\top\Psi^{-1}F_{\delta_x}))}{\mathrm{tr}(F_\xi^\top\Psi^{-1}F_\xi)^\ell}+o(\varepsilon).\end{aligned}$$

By the definition of the Gâteaux derivative, it follows that

$$G(\xi,\delta_x)=\frac{\mathrm{tr}((F_\xi^\top\Psi^{-1}F_\xi)^{\ell-1}(F_{\delta_x}^\top\Psi^{-1}F_\xi+F_\xi^\top\Psi^{-1}F_{\delta_x}))}{\mathrm{tr}(F_\xi^\top\Psi^{-1}F_\xi)^\ell}. \tag{5}$$

The Fréchet derivative can be rewritten as

$$d(x,\xi)=\lim_{\varepsilon\to 0+}\varepsilon^{-1}\{\log\Phi_\ell[M(F_\xi+\varepsilon(F_{\delta_x}-F_\xi))]-\log\Phi_\ell(M(F_\xi))\}.$$

Replace F_{δ_x} by $F_{\delta_x}-F_\xi$ in Equation (5), it is obtained that

$$d(x,\xi)=\frac{\mathrm{tr}((F_\xi^\top\Psi^{-1}F_\xi)^{\ell-1}(F_{\delta_x}^\top\Psi^{-1}F_\xi+F_\xi^\top\Psi^{-1}F_{\delta_x}))}{\mathrm{tr}(F_\xi^\top\Psi^{-1}F_\xi)^\ell}-2.$$

□

B More Optimal Criteria

A-Optimality A-optimality is used when the experiment aims to estimate more than one linear function of the parameters , e.g., $K^\top\hat{\beta}$, because $\mathrm{Cov}(K^\top\hat{\beta})=\sigma^2K^\top(X^\top X)^{-1}K$. Then we minimize $\sigma^2\mathrm{trace}[K^\top(X^\top X)^{-1}K]=\sigma^2\mathrm{trace}[(X^\top X)^{-1}KK^\top]=\sigma^2[(X^\top X)^{-1}A]$ with $A=KK^\top$, i.e., A can be any $p\times p$ symmetric non-negative definite matrix.

C-Optimality C-optimality is chosen when estimating one particular linear function of the parameters is of our interest, $c^\top\beta$, this criterion is a special case of A-optimality criterion. It is also called linear optimality. So we aim to minimize $\mathrm{var}(c^\top\hat{\beta})=\sigma^2c^\top(X^\top X)^{-1}c=\sigma^2\mathrm{trace}[c^\top(X^\top X)^{-1}c]=\sigma^2\mathrm{trace}[(X^\top X)^{-1}cc^\top]$.

D-Optimality D-optimality is used to minimize the confidence ellipsoid of the estimate $\hat{\beta}$, $\det\mathrm{Cov}(\hat{\beta})=\sigma^2\det(X^\top X)^{-1}=\sigma^2|X^\top X|^{-1}=\sigma^2\Pi_j\lambda_j^{-1}$, where the λ_j are eigenvalues of $X^\top X$.

E-Optimality E-optimality is used when we are interested in estimating a normalized linear function of the parameters. It can be considered a special case of C-optimality. So we may want to minimize $\max\mathrm{var}(c^\top\hat{\beta})$ for any c, such that, $\|c\|=1$. By leaving σ^2 out we have $\max_{\|c\|=1}\ \mathrm{var}(c^\top\hat{\beta})=\max_{\|c\|=1}\ c^\top(X^\top X)^{-1}c=\max$ eigenvalue of $(X^\top X)^{-1}$.

Non-Linear Models For non-linear models, we can apply the optimality to the Fisher information matrix of $\hat{\beta}$. The Cramér-Rao bound states that the inverse of the Fisher information is a lower bound on the variance of any unbiased estimator of β. Thus, when the model is not linear, we can maximize the Fisher information in different ways to achieve different optimality.

C More Objective Functions

Table 5 presents the objective functions used in Table 2. The search space is that $x_1\in[-2,2]$, $x_2\in[-2,2]$, $b\in\{10^4,10^3,10^2,10^1,10^0,10^{-1},10^{-2},10^{-3},10^{-4},10^{-5}\}$.

Table 5: objective functions for simulation experiments.

	objective function
sixhumpcamp	$f(x) = (4 - 2.1x_1^2 + \frac{x_1^4}{3})x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2 + b \cdot \mathcal{N}(0, 1)$
rosenbrock	$f(x) = 100(x_2 - x_1^2)^2 + (x_1 - 1)^2 + b \cdot \mathcal{N}(0, 1)$