FlexiBO: Cost-Aware Multi-Objective Optimization of Deep Neural Networks

Md Shahriar Iqbal [1] **Jianhai Su** [1] **Lars Kotthoff** [2] **Pooyan Jamshidi** [1]

Abstract

One of the key challenges in designing machine learning systems is to determine the right balance amongst several objectives, which also oftentimes are incommensurable and conflicting. For example, when designing deep neural networks (DNNs), one often has to trade-off between multiple objectives, such as accuracy, energy consumption, and inference time. Typically, there is no single configuration that performs equally well for all objectives. Consequently, one is interested in identifying Pareto-optimal designs. Although different multi-objective optimization algorithms have been developed to identify Pareto-optimal configurations, state-of-the-art multi-objective optimization methods do not consider the different evaluation costs attending the objectives under consideration. This is particularly important for optimizing DNNs: the cost arising on account of assessing the accuracy of DNNs is orders of magnitude higher than that of measuring the energy consumption of pre-trained DNNs. We propose FlexiBO, a flexible Bayesian optimization method, to address this issue. We formulate a new acquisition function based on the improvement of the Pareto hyper-volume weighted by the measurement cost of each objective. Our acquisition function selects the next sample and objective that provides maximum information gain per unit of cost. We evaluated FlexiBO on 7 state-of-the-art DNNs for object detection, natural language processing, and speech recognition. Our results indicate that, when compared to other state-of-the-art methods across the 7 architectures we tested, the Pareto front obtained using FlexiBO has, on average, a 28.44% higher contribution to the true Pareto front and achieves 25.64% better diversity.

[1]University of South Carolina [2]University of Wyoming. Correspondence to: Pooyan Jamshidi <pjamshid@cse.sc.edu>.

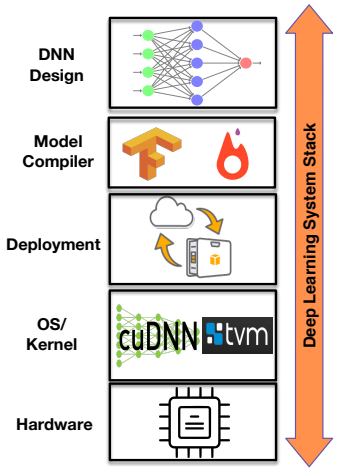


Figure 1. The deep learning system stack.

1. Introduction

DNNs are becoming ubiquitous and being deployed in many resource-constrained environments, such as mobile phones, smart home devices, and robots. Many different network design and hardware parameters impact the computational and memory requirements of DNNs and make them difficult to effectively deploy in resource-constrained environments. Designing an optimal DNN system is fundamentally a co-design problem: a designer must develop a DNN that achieves high prediction accuracy on the application task while simultaneously minimizing the resource requirements. The following factors create the challenging obstacles confronted in producing such a system design: (i) the design space is large and complex not only for DNN design options (Snoek et al., 2012) but also system stack options (see Figure 1). Further, (ii) the parameters tend to interact (e.g., a DNN design parameter like the number of filters may affect resource requirements). In addition, (iii) evaluating DNN architectures is expensive, since it involves training a DNN, which thus limits the total number of configurations that can be measured (Iqbal et al., 2019). Finally, (iv) due to multiple conflicting objectives, it is not possible to determine an optimal configuration that excels across all objectives. Hence, Pareto-optimal configurations are required in order to find the appropriate balance between multiple objectives,

which leads to the following research question: *how can we effectively and efficiently explore the configuration space of DNNs by sampling promising configurations that are likely to hit Pareto-optimal solutions*?

Multi-objective optimization techniques can effectively be utilized to identify the Pareto front of a DNN system for performance optimization. Several multi-objective optimization algorithms have been proposed to determine the Pareto front of a system (Knowles, 2006), (Ponweiser et al., 2008), (Emmerich & Klinkenberg, 2008), (Campigotto et al., 2013), (Picheny, 2015), (Zuluaga et al., 2013), (Hernández-Lobato et al., 2016a), (Zuluaga et al., 2016), (Sener & Koltun, 2018). However, these methods perform measurements by always evaluating all objectives contained in the objective space jointly at the same point in the design space. This approach is, in particular, sub-optimal for problems like DNN optimization as evaluating for accuracy of a DNN is more expensive when compared to evaluating energy consumption or inference time, because it requires the retraining of the network (Hernández-Lobato et al., 2016b). An approach based on decoupled objective evaluations has been proposed to enable independent evaluations across objectives in two-objectives optimization of neural hardware accelerators (Hernández-Lobato et al., 2016b). In this work, we explicitly incorporate the cost of evaluating each objective into a new acquisition function that chooses not only which point to evaluate next but also across which objective.

To motivate our approach, we first performed a small experiment to separately optimize accuracy and energy consumption of Inception v1 (Szegedy et al., 2015) using a cost-aware and a cost-unaware single objective Bayesian optimization approach with the expected improvement (EI) acquisition function (Emmerich & Klinkenberg, 2008). We tuned the CPU frequency of the runtime environment (NVIDIA Jetson TX2) and the total number of filters in the Inception v1 network. Here, the total number of filters affects both energy consumption and accuracy, whereas CPU frequency only affects energy consumption. In contrast to tuning CPU frequency, every time the number of filters is changed, it necessitates the retraining of the DNN before the effect of the change on accuracy and energy consumption can be measured. As a result, measuring the effect of varying the number of filters has a higher computational cost versus measuring the effect of varying CPU frequency. Given a limited measurement budget, an optimization method is more efficient if it explores the configuration space by tuning the less costly objective, i.e., CPU frequency. The cost-aware single objective Bayesian optimization approach classifies the time needed to retrain a network as the cost and incorporates this computed cost into the EI acquisition function in determining the next configuration to select. The optimization paths with and without cost awareness are shown in Figure 2. Although both paths

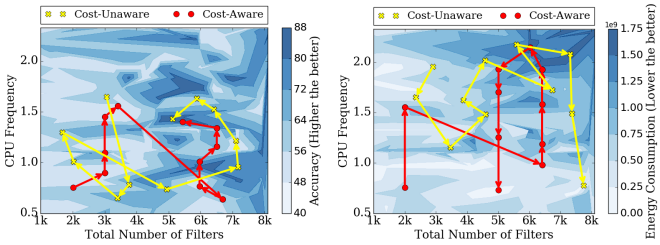


Figure 2. Given that changing the number of filters results in a costly retraining of the Inception v1 network, the cost-aware approach reaches the optimum using fewer costly evaluations when comparing to the cost-unaware approach.

ultimately found the optimum, the cost-aware approach had to retrain the network only 7 times, while the cost-unaware approach had to retrain it 11 times.

In this paper, we propose FlexiBO, a framework for multi-objective Bayesian optimization that has the following properties:

- *Multi-objective*: FlexiBO explicitly considers multiple objectives involved in DNN optimization, such as accuracy and energy consumption.
- *Flexible and Cost-aware*: FlexiBO allows users to specify the costs, which can be customized to account for the specific objectives involved in the particular optimization problem and for the target platform for which the DNN is being optimized, that are then incorporated by FlexiBO in guiding the search process.
- *Generality*: FlexiBO is a general framework that can be employed for optimizing not only different DNN architectures but also other optimization problems where the cost would be different across objectives. In this paper, we demonstrate this property by applying FlexiBO to optimize Xception (Chollet, 2017), MobileNet (Howard et al., 2017), LeNet (LeCun et al., 1998), ResNet (He et al., 2016), SqueezeNet (Iandola et al., 2016), BERT (Devlin et al., 2018), DeepSpeech (Hannun et al., 2014).
- *Cross-stack*: FlexiBO enables automatic optimization of DNNs in the joint space of architectures, hyperparameters, and the computer system stack[1] (see Figure 1).
- *Effectiveness and Efficiency*: The configurations of DNNs found by FlexiBO achieve higher accuracy and lower energy consumption. Experimental results also confirm that FlexiBO efficiently searches the configuration space as it explores the search space more effectively using the cost-aware approach.

[1] Also known as deep learning system stack (Chen et al., 2018)

Our work is motivated by a state-of-the-art multi-objective optimization algorithm called Pareto Active Learning (PAL) (Zuluaga et al., 2013) that identifies Pareto-optimal configurations; nonetheless, our approach uses a different Pareto front construction method than that of PAL and introduces a new acquisition function to sample the next configuration and objective for evaluation that incorporates cost into account. To identify Pareto-optimal configurations, a surrogate model is fit to the performance response surface in order to predict the objective values given a configuration, and an acquisition function is repeatedly evaluated on it to determine the next configuration and objective to evaluate. The acquisition function incorporates the uncertainty of the surrogate model's prediction so as to balance exploration and exploitation. We thus formulate a new cost-aware acquisition function that iteratively improves the quality of the Pareto-optimal solution space, also known as the Pareto region. This acquisition function selects the objective across which the configuration will be evaluated in addition to selecting the next configuration to evaluate. Consequently, we can make a trade-off between the additional information obtained through an evaluation with the cost of obtaining it; whereby, we ensure that we do not perform costly evaluations for little potential gain. Our intuition is that by evaluating the less costly objective without the necessity of evaluating the more costly objective, we can traverse the objective space and find the near-optimal configuration with increased efficiency.

We evaluated FlexiBO to optimize 7 different DNN architectures deployed in resource-constrained environments for object detection, natural language processing, and speech recognition. The results from our experiments indicate that FlexiBO consistently samples better configurations than state-of-the-art algorithms throughout the design space search, which results in the faster production of superior Pareto fronts. More precisely, the Pareto front determined by FlexiBO is of similar quality to those produced by the state-of-the-art algorithms, despite that FlexiBO does so with an average of 80.23% less cost. In addition, we also show that given the same cost, the Pareto front obtained by FlexiBO has a 28.44% higher contribution and achieves 25.64% better diversity. Notably, while we merely evaluated FlexiBO for optimizing DNN systems, our proposed approach is general and can be applied in other application domains as well.

Our primary contributions are as follows:

- We propose FlexiBO: a new cost-aware multi-objective optimization approach that iteratively selects the next configuration and an objective to measure in multi-objective optimization scenarios (Section 3).
- We also propose a new acquisition function that allows for trade-off between the additional information gained through an evaluation and the cost being incurred as a result of the evaluation (Section 3.3).
- We comprehensively evaluate multiple variations of FlexiBO with different acquisition functions in 7 different types of DNN architectures selected from 3 different domains and compare its performance to PAL, random search, and two forms of single-objective Bayesian optimization methods (Section 5).

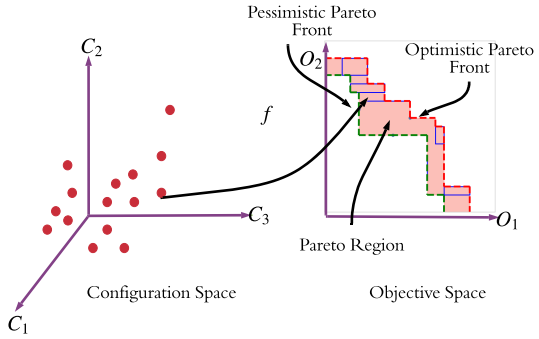


Figure 3. The configuration space is mapped to the objective space. In the objective space, rectangles show uncertainty regions for configurations that are not evaluated for any objective; while lines show uncertainty regions for configurations that are not evaluated for one objective, points are used for configurations that are evaluated for both objectives, which points accordingly include no uncertainty.

The code and experimental data that can be used to replicate our experiments are also available at `https://github.com/softsys4ai/FlexiBO`.

2. Background

2.1. Surrogate Models

Surrogate models replace expensive, time-consuming, and otherwise difficult to measure black-box functions with models that not only provide an approximation of the measurements from the underlying black-box process but also are inexpensive and fast to evaluate. In this paper, we use Gaussian Processes (GP) and random forests (RF) as surrogate models.

2.1.1. Gaussian Processes

A function f is modeled as a sample from an n-variate GP distribution. A GP distribution over $f(\boldsymbol{x})$ is characterized by its mean, $\mu(\boldsymbol{x})$, and co-variance, $k(\boldsymbol{x}, \boldsymbol{x}^*)$, (Williams & Rasmussen, 2006). We model each objective function $f_i(\boldsymbol{x})$ as a sample from an independent GP distribution. At each iteration t, the algorithm selects a design $\boldsymbol{x_t}$ for evaluation that yields a noisy sample as $y_{t,i} = f(\boldsymbol{x_t}) + v_{t,i}$. The posterior distribution of f_i, for the set of configura-

tions sampled from the design space E, is a GP with mean $\mu_{t,i}(\boldsymbol{x})$, co-variance $k_{t,i}(\boldsymbol{x}, \boldsymbol{x}^*)$, and variance $\sigma^2_{t,i}(\boldsymbol{x})$, assuming $v_{t,i} \sim N(0, \sigma^2)$ (i.i.d. Gaussian noise).

$$\mu_{t,i}(\boldsymbol{x}) = k_{t,i}(\boldsymbol{x})^T (K_{t,i} + \sigma^2 I)^{-1} y_{t,i} \tag{1}$$

$$k_{t,i}(\boldsymbol{x}, \boldsymbol{x}^*) = k_i(\boldsymbol{x}, \boldsymbol{x}^*) - k_{t,i}(\boldsymbol{x})^T (K_{t,i} + \sigma^2 I)^{-1} k_{t,i}(\boldsymbol{x}^*) \tag{2}$$

$$\sigma^2_{t,i}(\boldsymbol{x}) = k_{t,i}(\boldsymbol{x}, \boldsymbol{x}) \tag{3}$$

Here the posterior distribution captures the uncertainty of $f(\boldsymbol{x})$ for points $\boldsymbol{x} \in E$, namely, those points for which the objectives have not yet been evaluated.

2.1.2. Random Forests

RF is an ensemble learning algorithm that builds forests of multiple trees, which are trained independently and whose predictions are averaged to obtain the prediction of the entire forest, i.e., $\mu_t(\boldsymbol{x})$. We compute the uncertainty of each prediction as the standard deviation $\sigma_t(\boldsymbol{x})$ across all w trees.

$$\mu_t(\boldsymbol{x}) = \frac{\sum_{i=1}^{l} \mu_i(\boldsymbol{x})}{w} \tag{4}$$

$$\sigma_t(\boldsymbol{x}) = \sqrt{\frac{\sum_{i=1}^{l} |\mu_t(\boldsymbol{x}) - \mu_i(\boldsymbol{x})|^2}{w}} \tag{5}$$

2.2. Multi-Objective Optimization

In multi-objective optimization, we simultaneously optimize n potentially conflicting objective functions $\boldsymbol{f} = f_1, \ldots, f_n$, where $\boldsymbol{f} : E \subseteq \mathbb{R}^m \rightarrow \boldsymbol{O} \subseteq \mathbb{R}^n$ for some n-dimensional objective space $\boldsymbol{O}$ and a finite design space E with dimensionality m. A multi-objective optimization (maximization) can be expressed as:

$$argmax_{\boldsymbol{x} \in E} f(\boldsymbol{x}) \tag{6}$$

It is generally not possible to find a solution that maximizes each objective equally, but instead, there is a trade-off between them. Pareto-optimal points represent the best compromises across all objectives; in particular, a Pareto-optimal solution is a point $\boldsymbol{x} \in E$ for which it is not possible to find another point $\boldsymbol{x}^* \in E$, such that $f_i(\boldsymbol{x}^*) > f_i(\boldsymbol{x})$ for all $i \in n$.

Formally, for multi-objective maximization, $\boldsymbol{x}^* \succeq \boldsymbol{x}$ ($\boldsymbol{x}^*$ dominates $\boldsymbol{x}$) if and only if $f(\boldsymbol{x^*}) \succeq f(\boldsymbol{x})$, which means $f_i(\boldsymbol{x}^*) \geq f_i(\boldsymbol{x})$ for all $i \in n$. Pareto-optimal points are not dominated and form the Pareto front, which maps points with the optimal trade-off in the objective space.

Pareto-optimality Bayesian multi-objective optimization is an iterative algorithm where a prior model is chosen for the expensive-to-evaluate black-box functions involved in an optimization problem. In Bayesian multi-objective optimization, each point $\boldsymbol{x}$ in the configuration space is assigned an uncertainty region $R_t(\boldsymbol{x})$ at each iteration t using model f. The minimum value of the uncertainty region, $\min(R_t(\boldsymbol{x}))$, is regarded as the pessimistic value of $\boldsymbol{x}$; similarly, the maximum value of the uncertainty region, $\max(R_t(\boldsymbol{x}))$, is regarded as the optimistic value of $\boldsymbol{x}$. The pessimistic Pareto front, P_{pess}, is constructed using the undominated pessimistic values of $\boldsymbol{x}$, where $\boldsymbol{x} \in E$. The optimistic Pareto front, P_{opt}, is likewise constructed using the undominated optimistic values of $\boldsymbol{x}$ in the design space.

Pareto region is the region between the pessimistic and optimistic Pareto fronts, and as we denote it as PR throughout this paper. As shown in Figure 4, the set of solutions inside the Pareto region are classified as the Pareto-optimal solutions. Let $R(P)$ be a region in the objective space, where each point in the region is dominated by at least one point in the Pareto front, P. Then, the Pareto region, PR, can be calculated as:

$$\begin{aligned} PR &= R(P_{opt}) - R(P_{pess}), \\ R(P) &= \{\boldsymbol{o} \in O, o_i \geq 0; \exists \boldsymbol{o^*} \in P, \boldsymbol{o} \preceq \boldsymbol{o^*}\} \end{aligned} \tag{7}$$

2.3. Metrics

To determine the evaluation cost for each of the objectives that we seek to optimize and in order to assess the quality of the Pareto front approximations obtained, two types of performance metrics are used: cost and prediction quality metrics.

2.3.1. Cost Metrics

As a quality indicator, we use the cost of measuring an objective, which we term as the objective evaluation cost.

Objective evaluation cost For multi-objective optimization, we assume that the objective evaluation cost for the cheapest objective, O_{cheap}, is 1. The cheapest objective is identified by comparing the computational efforts of different objectives, and the objective with the lowest computational effort is determined as the cheapest one. For any objective O_i, the objective evaluation cost, $\boldsymbol{\Psi_i}$, where $1 \leq i \leq n$, is defined as follows:

$$X = \begin{cases} 1, & \text{if } O_i = O_{cheap} \\ \Psi_i = \frac{\theta_i}{\phi \times \theta_{cheap}}, & \text{otherwise} \end{cases} \tag{8}$$

where θ_i is the computational effort to evaluate O_i; θ_{cheap} is the effort to evaluate O_{cheap}; and ϕ is a balancing parameter to be chosen later (see Section 3.3).

2.3.2. Prediction Quality Metrics

To assess the quality of the Pareto front, we use the following prediction quality metrics: total evaluation cost, Pareto

volume reduction, contribution, and diversity.

Pareto volume reduction For a query q, the Pareto volume reduction is defined as follows:

$$\Delta V = V - V_q, \tag{9}$$

where V and V_q are the volumes of the region between the pessimistic and optimistic Pareto fronts before and after query q, respectively.

Contribution rate indicator When the true Pareto front is unknown, the contribution rate indicator can be used to determine the quality of the obtained Pareto front (Cao et al., 2015). Here, we refer to the contribution rate indicator as "contribution" throughout the paper. The contribution of a Pareto front P is the ratio of hypervolume indicators, I_H, of the corresponding modified Pareto front, P^*, and the surrogate of the true Pareto front, P_s, which is defined as follows:

$$Contribution = \frac{I_H(P^*, r)}{I_H(P_s, r)}, \tag{10}$$

where P_s is constructed by combining points from l competing Pareto fronts P_1, P_2, P_3,..., P_l. Furthermore, the modified Pareto front, P^*, contains only those points of P that are not dominated by P_s, which is defined using the following:

$$P^* = P \cap P_s \tag{11}$$

The hypervolume indicators for P^* and P_s with respect to a reference point r are $I_H(P^*, r)$ and $I_H(P_s, r)$ respectively. The hypervolume indicator I_H of a Pareto front P with reference to r is computed using the following equation:

$$I_H(P, r) = \lambda(\cup_{s \in P} space(s, r)), \tag{12}$$

where λ is the Lebesgue measure and $space(s, r)$ corresponds to the set of objective vectors that are dominated by points in P, which is enclosed by the reference set $Q_{ref} = r$. The value of the contribution metric ranges between 0 and 1. A contribution value of 1 indicates that the particular Pareto front is exactly the surrogate of the true Pareto front, while a contribution value of 0 means that all points on the particular Pareto front are dominated by the surrogate of the true Pareto front.

Diversity measure is used to determine the spread between a Pareto front with an ideal point, ip_i, and nadir point, np_i, for the objective $(O_i)_{1 \leq i \leq n}$, and the Pareto-optimal points (Audet et al., 2018). Here, we refer to the diversity measure as "diversity" throughout the the paper. To determine diversity, a grid environment is constructed using the following:

$$ub_i = np_i + \frac{np_i - ip_i}{2 \times div}, lb_i = ip_i, \tag{13}$$

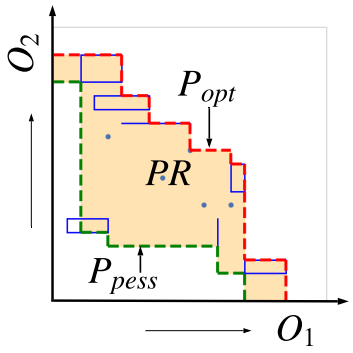


Figure 4. The shaded area between the pessimistic and optimistic Pareto fronts indicates the Pareto region.

where ub_i and lb_i indicate the upper and lower bounds of the grid for the objective $(O_i)_{1 \leq i \leq n}$. Once the grid is constructed, the number of divisions (div) are set by the user, and hyper-boxes of size d_i are afterward constructed using the following equation:

$$d_i = \frac{ub_i - lb_i}{div} \tag{14}$$

Finally, the ratio of the number of hyper-boxes containing one or more Pareto-optimal points and the total number of hyper-boxes are computed to determine diversity. The values of diversity range from 0 to 1, where 1 indicates the maximum diversity for a hyper-box containing at least one Pareto-optimal point.

3. FlexiBO: Flexible Bayesian Multi-Objective Optimization

We describe our *Flexible Bayesian Multi-Objective Optimization* (FlexiBO) algorithm in this section. Typically, in a multi-objective setting, each objective O_i, where $1 \leq i \leq n$, is modeled with a separate surrogate model. In many applications, evaluating different objectives, $O_1, \ldots, O_n$, incurs different costs, which may be expensive. Therefore, we wish to identify the Pareto region, PR, that is constructed using undominated points $U \subset E$ without evaluating all inputs $\boldsymbol{x} \in E$.

FlexiBO extends the concepts employed by the state-of-the-art PAL algorithm (Zuluaga et al., 2013); PAL identifies a set of Pareto-optimal points to build a Pareto front in a multi-objective optimization setting by iteratively selecting a sequence of points $(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_T})$ in E which are Pareto-optimal; albeit, PAL collect the points in a coupled fashion by always evaluating all objectives involved (e.g., accuracy, energy, inference time) jointly at the same design point, which is sub-optimal. In contrast, FlexiBO does so using a decoupled approach (Hernández-Lobato et al., 2016b) in which, at each iteration, it chooses which point to evaluate

Algorithm 1 FlexiBO algorithm

Input: Design space: E; Maximum number of objectives: n; Number of iterations: T; Surrogate model prior: $\mu_{0,i}$, $\sigma_{0,i}$ where $1 \leq i \leq n$; Scaling parameter: β_t for $1 \leq t \leq T$.

Output: Pareto region: PR.

/* Initialization */

$U_0 = \varnothing$; /* Undominated points set */

S_0 = Randomly selected k samples from E : $k \ll |E|$; /* Evaluated samples set */

$R_0(\boldsymbol{x}) = \mathbb{R}^n$ for all $\boldsymbol{x} \in E$; /* Uncertainty region for each points */

for $t = 1, 2, 3 \ldots, T-1$ **do**

  /* Modeling */

  Fit a surrogate model to S_t;

  Obtain $\boldsymbol{\mu_t}(\boldsymbol{x})$ and $\boldsymbol{\sigma_t}(\boldsymbol{x})$ for all $\boldsymbol{x} \in E$. Here, $\boldsymbol{\mu_t}(\boldsymbol{x}) = \hat{\boldsymbol{f}}(\boldsymbol{x})$ and $\boldsymbol{\sigma_t}(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in S_t$;

  $R_t(\boldsymbol{x}) = \boldsymbol{\mu_t}(\boldsymbol{x}) - \beta_t^{1/2}\boldsymbol{\sigma_t}(\boldsymbol{x}) \leq \boldsymbol{y} \leq \boldsymbol{\mu_t}(\boldsymbol{x}) + \beta_t^{1/2}\boldsymbol{\sigma_t}(\boldsymbol{x})$ for all $\boldsymbol{x} \in E$;

  /* Pareto region construction */

  for *all* $\boldsymbol{x} \in E$ **do**

    if *no* $\boldsymbol{x}^* \neq \boldsymbol{x}$ *exists such that* $\min(R_t(\boldsymbol{x})) \succeq \max(R_t(\boldsymbol{x}^*))$ **then**

      $U_t = U_t \cup \{\boldsymbol{x}\}$

  Get the pessimistic Pareto front P_{pess} and the optimistic Pareto fronts P_{opt} using U_t as the set of candidate points and $R_t(\boldsymbol{x})$ as uncertainty region from Algorithm 2;

  Compute the volume of the Pareto region, V, using P_{pess} and P_{opt};

  /* Sampling */

  Compute the change of volume per cost $\Delta_i(\boldsymbol{x})$ for each objective O_i, $1 \leq i \leq n$, for all $\boldsymbol{x} \in U_t$ using Algorithm 3;

  Choose the next sample $\boldsymbol{x}_{t+1}$ and objective O_i using $argmax_{\boldsymbol{x} \in \{P_{pess} \cup P_{opt}\}} \Delta_i(\boldsymbol{x})$, $1 \leq i \leq n$;

  $S_t = S_t \cup \{\boldsymbol{x}_{t+1}\}$;

Compute PR using equation 7;

return PR;

next but also across which objective. More specifically, FlexiBO is a cost-aware multi-objective optimization algorithm that iteratively and adaptively selects a sequence of points and objectives, $(\boldsymbol{x_1}, \boldsymbol{O_1}), (\boldsymbol{x_2}, \boldsymbol{O_2}), \ldots, (\boldsymbol{x_T}, \boldsymbol{O_T})$, for the purpose of determining a *Pareto region* instead of a Pareto front and then continuing by improving its quality until the maximum number of iterations, T, is reached. To identify the Pareto-optimal points in E from which to construct PR, we train surrogate models with a small subset of E. We use separate surrogate models for each objective to predict the values of the objective functions f_i, where $1 \leq i \leq n$ for

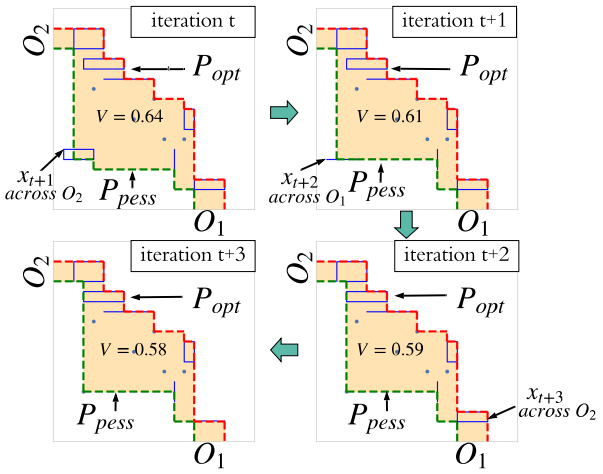


Figure 5. The volume of the Pareto region, V, is decreased over time. The next sample to be evaluated, $\boldsymbol{x}_{t+1}$, is determined by the acquisition function.

points in E, with the aim of predicting Pareto-optimality with high probability.

A point $\boldsymbol{x}$ that has not been sampled for any objective is predicted as a vector $\hat{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{\mu}(\boldsymbol{x}) = (\mu_1(\boldsymbol{x}), \ldots, \mu_n(\boldsymbol{x}))$, and the uncertainty of this prediction is interpreted as $\boldsymbol{\sigma}(\boldsymbol{x}) = (\sigma_1(\boldsymbol{x}), \ldots, \sigma_n(\boldsymbol{x}))$. We use the predicted values and the uncertainty values for such a prediction to determine the uncertainty region for each point $\boldsymbol{x} \in E$. These uncertainty regions are later used to determine the dominance of $\boldsymbol{x}$. PAL assumes that any uncertain point $\boldsymbol{x}$ which becomes dominated at iteration t would not become undominated in later iterations and considers only the undominated points $\boldsymbol{x} \in U$ for the purpose of dominance determination. However, in our approach, we do not make this assumption, but instead, we consider all the points $\boldsymbol{x} \in E$ to determine dominance. Although our consideration of all of the points $\boldsymbol{x} \in E$ in the dominance calculation leads to an increase in computational complexity of the optimization algorithm, for some applications, especially for DNN systems, this increase of cost is negligible when compared to the evaluation cost of each objective. Later, undominated points $\boldsymbol{x} \in U$ are used to classify as $\boldsymbol{x}$ belonging to the set of Pareto-optimal points or otherwise falling fully outside such areas. These identified Pareto-optimal points are then used to construct the Pareto fronts.

After initialization, we iterate until the maximum number of iterations is reached. Every iteration t consists of three stages: modeling, construction of the Pareto region, and sampling. However, unlike PAL, FlexiBO determines a Pareto region instead of an actual Pareto front and proposes a new acquisition function, i.e., change in volume of the Pareto region per unit of cost incurred for each candidate sam-

Algorithm 2 Construction procedure of the Pareto region

Input: Candidate points: C; Uncertainty region for candidate points: ρ.
Output: Pessimistic Pareto-points: P_{pess}; Optimistic Pareto-points: P_{opt}.

/* Pessimistic Pareto front */
C_{pess} = The list of pessimistic points of ρ, which are sorted in the descending order across the first objective;
$P_{pess} = C_{pess}[0]$;
$\eta_{rank} = argsort(C_{pess})$ (the ascending order across the second objective);
$iter = 1$;
while $iter \leq |\eta_{rank}|$ **do**
  $\xi_{cur} = \eta_{rank}[iter]$;
  if *there exists any* $\eta_{rank}[iter] \leq \xi_{cur}$ **then**
    Find the maximum index $iter_{max}$ where $\eta_{rank}[iter] < \xi_{cur}$;
    for $j = iter \ldots iter_{max}$ **do**
      Update $C_{pess}[j] = C_{pess}[iter]$ across the second objective;
      Update $P_{pess} = P_{pess} \cup \{C_{pess}[j]\}$;
      Update $iter = iter_{max}$;
  else if *no* $\eta_{rank}[iter] < \xi_{cur}$ *exists* **then**
    $P_{pess} = P_{pess} \cup \{C_{pess}[iter]\}$;
/* Optimistic Pareto front */
C_{opt} = The list of optimistic points of ρ, which are sorted in the descending order across the first objective;
$P_{opt} = C_{opt}[0]$;
for $iter = 1, 2, 3, \ldots, |C_{opt}| - 1$ **do**
  if $C_{opt}[iter + 1] \geq C_{opt}[iter]$ *across the second objective* **then**
    $P_{opt} = P_{opt} \cup \{C_{opt}[iter + 1]\}$;
return P_{pess}, P_{opt}

ple. The construction of each surrogate model in FlexiBO is based on the sampled points at each iteration, which are used to make predictions of the objective values and their concomitant uncertainties for non-sampled configurations. We use these predictions to construct a new Pareto front and then select the configuration to next evaluate that most improves the Pareto front (as measured as a reduction in Pareto volume) each weighted by the cost of measuring the particular objective. After the termination of the algorithm, the configurations predicted to be part of the Pareto front in the final iteration are returned as the Pareto region, namely PR. The pseudocode for Bayesian optimization procedure implemented by FlexiBO is outlined in Algorithm 1.

Algorithm 3 Calculation of the change of volume per cost

Input: Candidate points: C; Current pessimistic points: P_{pess}; Current optimistic points: P_{opt}; Current volume of Pareto front: V; Objective: O_i where $1 \leq i \leq n$; Uncertainty region for candidate points: ρ; cost for O_i: Ψ_i where $1 \leq i \leq n$.
Output: Change of Pareto front volume weighted by cost: Δ.

$\Delta = \varnothing$;
for *all* $\boldsymbol{x} \in C$ **do**
  for *each* O_i *where* $1 \leq i \leq n$ **do**
    Compute the estimated mean, $\mu_i(\boldsymbol{x})$, of pessimistic and optimistic values of $\boldsymbol{x}$ across O_i;
    Get the set of current candidate points $\acute{C}$ by replacing pessimistic and optimistic values of $\boldsymbol{x}$ across O_i with $\mu_i(\boldsymbol{x})$;
    Get updated pessimistic Pareto front P_{pess} and optimistic Pareto front P_{opt} from Algorithm 2 using $\acute{C}$ as the set of candidate points and ρ as corresponding uncertainty region;
    Compute updated volume the Pareto front V_q using P_{pess} and P_{opt};
    Compute the change of volume along O_i as $\Delta V_i = V - V_q$;
    Compute change of volume per cost along O_i as $\Delta_i(\boldsymbol{x}) = \frac{\Delta V_i}{\Psi_i}$;
    $\Delta = \Delta \cup \{\Delta_i(\boldsymbol{x})\}$
return Δ

3.1. Modeling

At iteration t, to predict the mean vector, $\boldsymbol{\mu_t(x)}$, and standard deviation, $\boldsymbol{\sigma_t(x)}$, for all $\boldsymbol{x} \in E$, we use separate surrogate models (GP or RF). Every configuration $\boldsymbol{x} \in E$ is then assigned an uncertainty region, $R_t(\boldsymbol{x})$, which is computed as follows (cf. Figure 3):

$$R_t(\boldsymbol{x}) = [\boldsymbol{\mu_t}(\boldsymbol{x}) - \beta_t^{1/2}\boldsymbol{\sigma_t}(\boldsymbol{x}), \boldsymbol{\mu_t}(\boldsymbol{x}) + \beta_t^{1/2}\boldsymbol{\sigma_t}(\boldsymbol{x})], \quad (15)$$

where β_t is a scaling parameter that defines how large the uncertain region is in proportion to $\boldsymbol{\sigma_t}$ for the purpose of analyzing the exploration-exploitation trade-off. Similar to PAL (Zuluaga et al., 2013; 2016), for n objectives at iteration t, we define β_t as $\beta_t = \frac{1}{3}\sqrt{\frac{2\log n|E|\pi^2 t^2}{6\delta}}$. The pessimistic and optimistic values within the uncertainty region $R_t(\boldsymbol{x})$ are determined by $\min(R_t(\boldsymbol{x}))$ and $\max(R_t(\boldsymbol{x}))$, respectively. The dimension of $R_t(\boldsymbol{x})$ depends on the objective space, so in two objectives space, $R_t(\boldsymbol{x})$ is thus two dimensional.

3.2. Pareto Region Construction

Similar to PAL, at each iteration t, to determine the Pareto region, we construct a pessimistic and optimistic Pareto

front using a set of undominated points, U_t. The pseudocode for construction of the pessimistic and optimistic Pareto fronts for two objectives is provided in Algorithm 2.

We first identify U_t using the following rules for each $\boldsymbol{x} \in E, \boldsymbol{x} \neq \boldsymbol{x}^*$:

$$\begin{aligned} \boldsymbol{R1} &: \boldsymbol{x} \in U_t \quad \text{if} \quad \min(R_t(\boldsymbol{x})) \preceq \max(R_t(\boldsymbol{x}^*)) \\ \boldsymbol{R2} &: \boldsymbol{x} \notin U_t \quad \text{if} \quad \min(R_t(\boldsymbol{x}^*)) \preceq \max(R_t(\boldsymbol{x})) \end{aligned} \tag{16}$$

Next, the pessimistic Pareto front, P_{pess}, is constructed from the pessimistic points $\min(R_t(\boldsymbol{x}))$ of each $\boldsymbol{x} \in U_t$. The pessimistic value $\min(R_t(\boldsymbol{x}))$ of a point $\boldsymbol{x}$ that is already in P_{pess} is updated if $\min(R_t(\boldsymbol{x}))$ is less than $\min(R_t(\boldsymbol{x}^*))$ and $\max(R_t(\boldsymbol{x}))$ is greater than $\min(R_t(\boldsymbol{x}^*))$ but less than $\max(R_t(\boldsymbol{x}^*))$ of any other point $\boldsymbol{x}^* \in P_{pess}$. This process of constructing the pessimistic Paret-front ensures that any point that has the potential to be included in the Pareto region is not discarded from our consideration as the consequence of additional emphasis we place on the pessimistic values of the points in the uncertainty region. We use the ranks of the sorted pessimistic value across one objective to construct P_{pess} as shown in Algorithm 2. In contrast, each point $\boldsymbol{x} \in U_t$ is included in the optimistic Pareto front, P_{opt}, if $\max(R_t(\boldsymbol{x}))$ is greater than $\max(R_t(\boldsymbol{x}^*))$ of any previous point $\boldsymbol{x}^*$ that is already included in P_{opt}. Finally, the volume of the Pareto region between P_{pess} and P_{opt} is then calculated.

3.3. Sampling

We compute our acquisition function that considers the change of volume of the Pareto region per evaluation cost $\frac{\Delta V_i}{\Psi_i}$ for each $\boldsymbol{x} \in \{P_{pess} \cup P_{opt}\}$ for each objective O_i, where $1 \leq i \leq n$. To compute the change of volume of the Pareto region, we shrink the uncertainty region of each $\boldsymbol{x} \in \{P_{pess} \cup P_{opt}\}$ to its estimated mean—$\mu_i(\boldsymbol{x})$—across each O_i separately and then compute the change of Pareto volume, ΔV_i (pseudocode is provided in Algorithm 3). This assists us in determining the volume change of the Pareto region per unit of cost that would be achieved if a point $\boldsymbol{x} \in \{P_{pess} \cup P_{opt}\}$ is evaluated using the estimated mean value. We choose the next sample $\boldsymbol{x}_{t+1}$ and objective O_i using the following:

$$\boldsymbol{x}_{t+1} = argmax_{\boldsymbol{x} \in \left\{P_{pess} \cup P_{opt}\right\}} \Delta(\boldsymbol{x}) \tag{17}$$

Our algorithm selects points in P_{pess} and P_{opt} that will initially result in the construction of a large Pareto region, and it decreases the volume of the Pareto region iteratively (Figure 5). We do not consider points $\boldsymbol{x} \in U_t \setminus \{P_{pess} \cup P_{opt}\}$ because the process by which we constructed the optimistic and pessimistic Pareto fronts ensures that the change of volume, ΔV, for these points will be zero. Therefore, points $\boldsymbol{x} \in U_t \setminus \{P_{pess} \cup P_{opt}\}$ will not contribute to further improvement of the Pareto region. In addition, this exclusion increases the speed of FlexiBO.

4. Implementation and Workflow

The implementation of FlexiBO is shown in Figure 6. Each pass through this 6-step search process produces one configuration to sample for one objective, and the cycle is then iteratively repeated for the purpose of exploring the design space.

Step 1—Update Model: The outputs from the performance measurements are fed back to FlexiBO. FlexiBO uses this information to update the posterior distribution of its surrogate model (i.e., the GP or RF).

Step 2—Optimize Acquisition Function: Once the surrogate model has been updated, the acquisition function can then be recomputed and the next configuration and objective will be determined by optimizing the acquisition function.

Step 3—Measure Information Gain: At each exploration iteration, the parameters are required to be assigned with appropriate values. The value of each configuration option is chosen based on its ability to maximize the expected utility, while, the selection of the objective to evaluate is based on the expected maximum change of volume along that particular objective. The process is then iteratively repeated.

Step 4—Train DNN: At each newly selected configuration, the training of the DNN is not necessarily required. Instead, for the configurations where the expensive objective was not selected, FlexiBO uses the surrogate model to approximate the accuracy of the network. In order to measure the energy consumption of such configurations, FlexiBO instantiates the DNN architecture with random parameters, and FlexiBO then automatically deploys the architecture on the target device for profiling (Step 5). In contrast, for configurations where the expensive objective is selected, FlexiBO first instantiates the target DNN architecture, then sends the model to cloud, and afterwards conducts the training of the target DNN on a cloud server.

Step 5,6—DNN performance measurements: The performance measurements are collected from the target resource-constrained hardware for which we want to optimize the DNN. We set the parameters for the target architecture and retrieve the DNN that was trained according to the parameter settings of the current configuration for deployment. Once the optimization budget is exhausted, the Pareto-optimal configuration of the DNN will be returned.

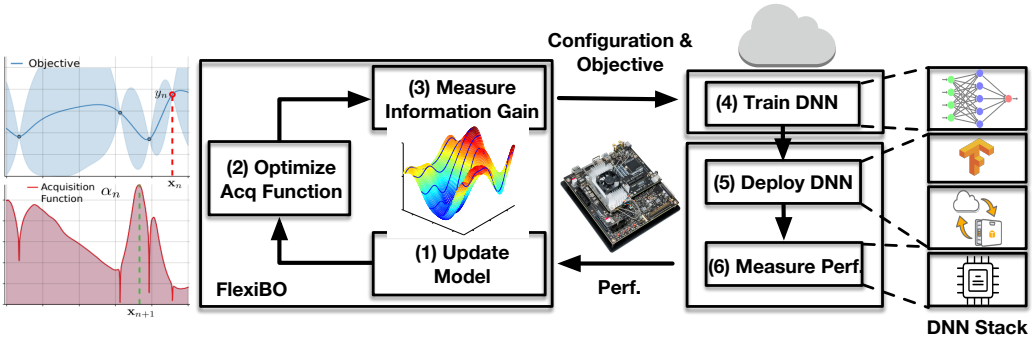


Figure 6. The implementation of FlexiBO and the experimental setup.

5. Experimental Evaluation

5.1. Experimental Setup

To evaluate FlexiBO, we use 8 different architectures in 3 different domains: object detection, natural language processing (NLP), and speech recognition. Table 1 lists the architectures, datasets, compilers, and the sizes of the training and test sets used in our experiments. We used NVIDIA Jetston TX2 as the hardware for our experiments and 4 hardware configuration options, 5 OS configuration options, and 2 different network configuration options depending on the type of architecture used to construct the DNN design space (see Table 2).

FlexiBO was used to guide the search. We used either Keras+TensorFlow or PyTorchTo to train and evaluate the DNNs (see Table 1 for details). Moreover, in contrast to previous research that used simulators, we directly measure energy usage on a real device.

To initialize FlexiBO, we measured the accuracy and energy consumption of 15 randomly sampled configurations from the configuration space of the corresponding DNN system. The cross-stack configuration space that we considered in our experiments is shown in Table 2. For example, hardware/OS configuration options affect the energy consumption of a DNN, while the network options (e.g., filter size) impact both accuracy and energy consumption.

We set the maximum number of iterations, T, to be 200. We then measured energy consumption 10 times and took the median with the aim of reducing measurement noise. Accuracy measurements do not suffer from noise and were thus not repeated. We used two versions of FlexiBO with two different surrogate models—GP (FlexiBO + GP) and RF (FlexiBO + RF). We compare our approach to random search (RS) (Bergstra & Bengio, 2012); single objective Bayesian optimization for energy consumption (SOBO (EC)) and accuracy (SOBO (Acc)); and PAL (Zuluaga et al., 2013).

RS randomly selects a configuration and an objective at each iteration to measure. For comparative analysis with FlexiBO, we set an upper bound on the number of times the expensive objective is evaluated in RS, which is equal to the number of times the expensive objective is evaluated in FlexiBO. SOBO (Acc) and SOBO (EC) optimize accuracy and energy consumption independently by using the maximum likelihood of improvement to acquire a new point and further measure both objectives at each iteration. For PAL implementation, we set $\epsilon = (\epsilon_i)_{1 \leq i \leq 2}$ value as 0.004% of each objective range $f_i : \max_{\boldsymbol{x} \in E}(f_i(\boldsymbol{x})) - \min_{\boldsymbol{x} \in E}(f_i(\boldsymbol{x}))$ for objective O_i, where $1 \leq i \leq n$. We repeated the entire optimization process 5 times for each architecture and afterward reported the median.

5.2. Results

Our results demonstrate that FlexiBO outperforms other optimization methods—RS, SOBO (EC), SOBO (Acc), and PAL—across all major metrics of interest. FlexiBO consistently finds better configurations and strictly better Pareto fronts and achieves both in fewer iterations and at less cost.

5.2.1. FlexiBO Finds Better Configurations

Figure 7 shows the evaluations of configurations in the objective space, which are selected by FlexiBO (GP, RF) versus those selected by SOBO (EC, Acc), RS, and PAL in the Xception + ImageNet architecture (moreover, similar results are observed for other architectures and are included in the supplementary materials). The objective functions—energy consumption and accuracy—are plotted on the x- and y-axis, respectively (better designs are closer to the upper-left part of the plots). We made the following observations based on the results: First, the configurations tend to cluster around the 55% accuracy mark in all methods, which is a reflection of the underlying scenarios since ImageNet is a classification problem with 200 categories for object detection and the Xception architecture correctly predicts 55% of the input images on average. Notwithstanding, DNN optimization is a difficult problem, and most configurations are sub-optimal

Table 1. The DNN architectures and datasets used in the experimental evaluation.

Architecture	Dataset	Compiler	Num. Layers	Num. Paremeters	Num. Classes	Training Size	Test Size	Domain
Xception	ImageNet	Keras	71	22M	200	150,000	20,000	Image
MobileNet	ImageNet	Keras	28	4.2M	200	100,000	30,000	
LeNet	MNIST	Keras	7	60000	10	60,000	10,000	
ResNet	CIFAR-10	Keras	50	2.5M	10	50,000	10,000	
SqueezeNet	CIFAR-10	Keras	3	1.2M	10	50,000	10,000	
BERT	SQuAD 2.0	PyTorch	12	110M	-	56,000	5,000	NLP
BERT	IMDB Sentiment	PyTorch	12	110M	2	50,000	2,000	
DeepSpeech	Common Voice	PyTorch	9	68M	29	300 (hours)	20 (hours)	Image

and cannot improve the average performance of already good architectures. Furthermore, while it is hard to discover an accurate DNN, it is easy to set a single parameter incorrectly that causes a sizeable decrease in model accuracy, such that the result is an optimization landscape reflecting sharp peaks indicating that the isolated good configurations are surrounded by many poor ones.

The results for PAL give some insight into the optimization space in a sense that it was able to find only one configuration with accuracy higher than 85%. PAL tends to sample configurations with good energy consumption but not accuracy, which suggests that of the two objectives, the DNN training space is more difficult to reason about than the hardware space. This finding is somewhat intuitive, since there are several simple relationships between hardware options with low interaction levels (Iqbal et al., 2019; Jamshidi et al., 2017a). While PAL can exploit these simple trends, it has difficulty simultaneously optimizing both objectives. In contrast, although FlexiBO and PAL have similar median energy consumption, FlexiBO consistently finds designs with substantially higher accuracy. FlexiBO's strength is derived from its ability to model intricate relationships between options and avoiding cliffs in the optimization landscape by using faster measurements across the cheaper objective. Nine of the configurations found by FlexiBO have high accuracy and among those, four have low energy consumption.

Insights. FlexiBO produces many energy-efficient configurations, which is, in part, a product of its design: FlexiBO tries to find a balance between exploitation and exploration by way of first exploring the cheaper objective in an effort to bypass the cliffs contained in the optimization landscape and further, by only measuring the DNN configuration across the expensive objective when it matters and thus, is prudent to do so.

5.2.2. FLEXIBO PRODUCES SUPERIOR PARETO FRONTS

We also evaluated the quality of the Pareto front using the contribution and diversity quality indicator metrics. Contribution is particularly useful when the shape and cardinality of the true Pareto front is unknown and can also be used to provide a fair comparison between Pareto fronts obtained from the use of different methods. A Pareto front approximation with a higher contribution value is closer to the estimated actual Pareto front. The diversity of Pareto front is another useful quality indicator that determines the extent of the spread. A Pareto front with higher diversity has more points for use in the optimization process and thereby ensures better control for performance tuning in the context of multiple objectives.

We evaluate the quality of the Pareto front approximations derived by FlexiBO in two modes of operation: Full capacity mode (FCM), where the optimization methods are allowed to run until the maximum number of iterations is reached, and Time-budget mode (TBM), where the optimization methods (other than FlexiBO) are allowed to run until the time when they have accrued the same cost in evaluating objectives as FlexiBO after the maximum number of iterations, which is the point of their respective terminations. A comparison of the Pareto fronts in FCM gives us insight into the quality of the Pareto fronts notwithstanding cost; whereas, TBM is a more realistic gauge for resource-constrained environments. Figure 8 and Figure 9 report the contribution and diversity values of the following different Pareto fronts: (i) actual Pareto fronts (FlexiBO + GP (Actual), FlexiBO + RF (Actual), PAL, RS, SOBO (Acc) and SOBO (EC); (ii) pessimistic Pareto fronts (FlexiBO + GP (Pess) and FlexiBO + RF (Pess)); and (iii) optimistic Pareto fronts (FlexiBO + GP (Opt) and FlexiBO + RF (Opt)). The actual Pareto fronts for FlexiBO are constructed using the configurations for which both objectives, such as accuracy and energy consumption, are evaluated. Pessimistic and optimistic Pareto fronts are constructed using the configurations for which at least one objective is evaluated.

Table 2. The configuration space for cross-stack optimization of DNNs.

Layer	Configuration Option	Range
Network	Image	
	Number of filters	32, 64, 128, 256, 512, 1024
	Filter size	(1×1), (3×3), (5×5), (7×7), (9×9)
	NLP	
	Max batch size	6, 12, 14, 16, 32, 64
	Max sequence length	64, 128, 256, 320, 384, 512
	Speech	
	Batch size	16, 32, 64, 128, 256, 512, 1024
	Number of epochs	13, 16, 32, 64, 128, 256, 512, 1024
OS	Scheduler policy	CFP, NOOP
	VFS cache pressure	0, 100, 500
	Swappiness	10, 60, 100
	Dirty background ratio	10,80
	Dirty ratio	5, 50
Hardware	Number of active cores	1 - 4
	Core frequency	0.3 - 2.0 (GHz)
	GPU frequency	0.1 - 1.3 (GHz)
	Memory controller frequency	0.1 - 1.8 (GHz)

Figures 8(a) and 8(b) plot the contribution values of the Pareto fronts obtained from using different optimization methods in FCM and TBM modes, respectively. We observe that in FCM, PAL has a 17.14% higher contribution value than FlexiBO + GP (next best method) in all architectures for the actual Pareto fronts (Figure 8(a)). It should be emphasized that FlexiBO + GP is able to achieve this performance with 80.23% less cost than PAL. Figure 8(b) indicates that, given the same cost, the actual Pareto front obtained by FlexiBO + GP has a 28.44% higher contribution value than PAL.

Insights. The quality of the Pareto fronts constructed by FlexiBO + GP is similar to the Pareto fronts derived from state-of-the-art algorithms but at a significantly reduced cost. Accordingly, with the same time budget, FlexiBO builds better Pareto fronts.

The diversity of the Pareto fronts as obtained by the optimization methods we evaluated with both FCM and TBM modes are shown in Figures 9(a) and 9(b), respectively. Figure 9(a) illustrates that PAL achieves a better spread for actual Pareto fronts and outperforms the next best optimization method, FlexiBO + GP, by 12.72%. However, between these two modes, where the volume of the Pareto region are almost similar, the optimistic and pessimistic Pareto fronts obtained using FlexiBO + GP are 21.64% and 14.06% more diverse than those obtained by PAL.

Figure 9(b) indicates that FlexiBO + GP (actual) outperforms other methods and obtains 25.64% better diversity values than the next best optimization method, PAL. For all architectures, the diversity value achieved by FlexiBO + RF is lower than FlexiBO + GP and PAL for actual Pareto fronts. The diversity values of the pessimistic and optimistic Pareto fronts obtained by FlexiBO + GP in TBM mode are 46.87% and 57.14% higher than those derived from PAL.

Insights. With the same time budget, FlexiBO identifies more diverse Pareto-optimal solutions.

5.2.3. FlexiBO Discovers Optimal Configurations at Lower Cost

Table 3 shows the total objective evaluation cost for 200 iterations of different optimization methods in the context of the different architectures used in our experiments. As SOBO (EC), SOBO (Acc), and PAL evaluate both accuracy and energy consumption for each of the 200 iterations, their total cost is same. The total cost for RS is equal to the minimum of the total evaluation cost of FlexiBO + GP and FlexiBO + RF, because the number of evaluations of the costly objective is the same as the minimum number of evaluations of either FlexiBO + GP or FlexiBO + RF.

It is evident from Table 3 that excluding SOBO (EC), FlexiBO + RF obtains the minimum total evaluation cost in the MobileNet (367.2), ResNet (676.4), SqueezeNet (423.9),

Table 3. The total objective evaluation cost and run time for 200 iterations of different optimization methods in different DNN architectures. The objective evaluation cost has no unit since it is the ratio of the time required to evaluate an objective and the time needed to evaluate the cheapest objective in a multi-objective setting. *Bold values indicate the minimum values for cost and time among the different optimization methods other than SOBO(EC). SOBO(EC) has the lowest cost (200) of any other methods, since the expensive objective, accuracy, is not evaluated by SOBO(EC) at any iteration.

	RS	SOBO (EC)	SOBO (Acc)	PAL		FlexiBO + RF		FlexiBO + GP	
	Cost	Cost	Cost	Cost	Time(h)	Cost	Time(h)	Cost	Time(h)
Xception	703	200	3,840	3,840	129.6	960	25.2	**703**	**22.4**
MobileNet	367.2	200	2,720	2,720	99.2	**367.2**	9.4	408	**8.1**
LeNet	272.8	200	1,360	1,360	32.1	396	5.9	**272.8**	**5.7**
ResNet	676.4	200	3,560	3,560	114.0	**676.4**	**21.3**	712	22.1
SqueezeNet	423.9	200	3,140	3,140	111.6	**423.9**	**20.4**	688.8	23.2
BERT-SQuAD	1112.8	200	4,280	4,280	151.3	1,177	37.9	**1,112.8**	**37.7**
BERT-IMDB	543	200	3,620	3,620	147.2	**543**	**23.4**	760.2	25.4
DeepSpeech	706.2	200	4,680	4,680	163.8	**706.2**	**26.2**	842.4	28.8

BERT-IMDB (543), and DeepSpeech (706.2) architectures, while in the Xception (703), LeNet (272.8), and BERT-SQuAD (1112.8) architectures, FlexiBO + GP obtains the minimum total evaluation cost. The respective total costs of FlexiBO + RF and FlexiBO + GP are 80.69% and 79.77% less than the total cost of SOBO (Acc) and PAL, on average.

From Table 3, we also observe that FlexiBO + RF performs in the minimum time (in hours) in the ResNet (21.3), SqueezeNet (20.4), BERT-IMDB (23.4) and DeepSpeech architectures (26.2), while FlexiBO + GP performs in the minimum time (in hours) in the Xception (22.4), MobileNet (8.1), LeNet (5.7), and BERT-SQuAD (37.7) architectures. On average, FlexiBO + RF and FlexiBO + GP require 84.12% and 81.68% less time to perform than PAL does in the context of all 7 architectures.

Insights. FlexiBO, on average, determines the Pareto-optimal configurations for accuracy and energy consumption at 80.23% less cost than the other optimization methods (with the exception of SOBO (EC)) and in 82.90% less time than PAL.

Multi-objective optimization of DNN systems is a costly task. In our experiments, training and performance measurements could each take hours per configuration, and it is not uncommon for modern DNNs to take days in order to train even just a single configuration. We compare the sample efficiency of FlexiBO, PAL, RS, and SOBO by analyzing the evolution of their Pareto fronts as a function of the cumulative number of configurations sampled. We use the Pareto hypervolume (Zitzler & Thiele, 1999), the volume between the bounded space made by the Pareto front and a fixed reference point in the objective space (here, we use the origin of the objective space), to evaluate the evolution of Pareto fronts derived from the samples obtained by the optimization methods.

A sound way to interpret these results is by comparing the depicted hypervolume curves against one another (Figure 10), since all three curves were computed using the same reference point. FlexiBO + GP and PAL consistently outperform the others as they decrease the hypervolume in all architectures at a more rapid pace using fewer samples. For example, in MobileNet, after only 40 samples, although FlexiBO reached a value of 0.10 (arbitrary units), it took PAL 100 samples (2.5x more) to find a Pareto front of similar quality to the one derived by FlexiBO. Only in the smallest architecture, LeNet, was PAL able to outperform FlexiBO. These results show that FlexiBO is more effective not only in larger configuration spaces but also in more difficult scenarios.

From this experiment, a comparison of the hypervolumes for RS and FlexiBO provides a marked illustration of the performance edge of FlexiBO over other approaches. A decisively illustrative observation that supports this conclusion is that between iterations 10 and 40, RS outperforms FlexiBO, meaning it found a better point earlier, which may seem to weaken the justification for using FlexiBO, but in fact, it strengthens the merits. RS samples the configuration space randomly, and thus, the large spike in hypervolume at point 10 is entirely coincidental; yet after 40 iterations, FlexiBO still outperformed RS for the remaining samples, which means that even though RS at the 10th iteration forward can find very good configurations, it was unable to leverage this capability as a means to effectively explore the design space. On the other hand, once FlexiBO had a reasonable surrogate model of the design space, it was able to exploit its samples and thereby, consistently find improvements to

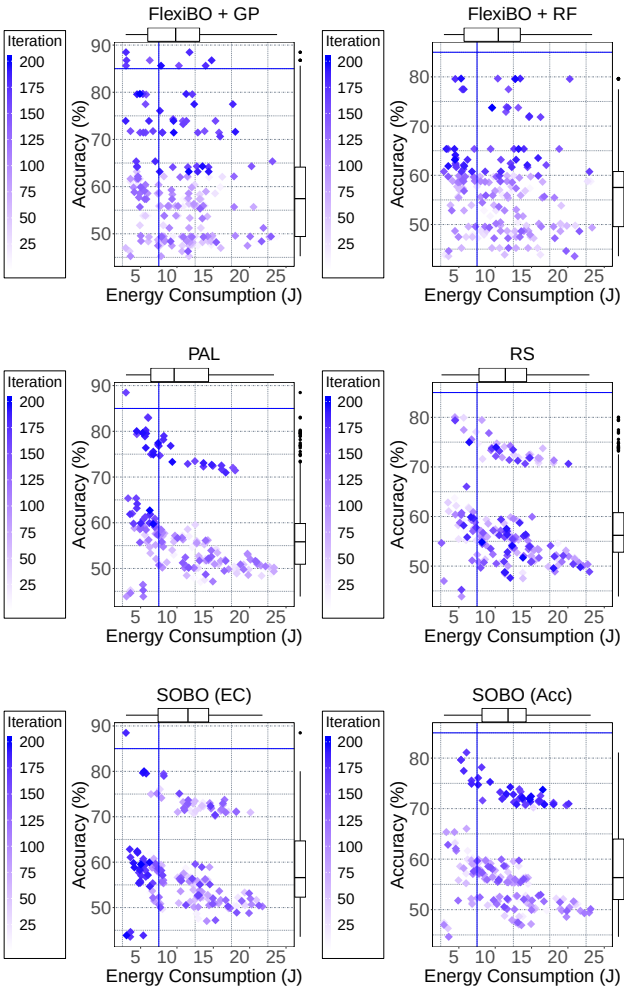


Figure 7. The exploration results in the Xception+ImageNet setting. It is desirable to have more evaluations towards the upper left part of each marginal scatter plot. FlexiBO outperforms other methods, since it finds more accurate and more efficient DNN architectures.

its approximated Pareto front.

6. Threats to Validity

In our experiments, we selected a diverse set of subject DNN systems and configuration options that influence the performance of DNNs and excluded some configuration options—such as scheduler latency, scheduler wakeup granularity, etc.—that had no impact on energy consumption or accuracy of DNNs. To select influential configuration options for this experiment, we studied the causal structure of different DNN systems' performance using partial ancestral graphs (PAG), which can be thought of as maps of dependency structures of probability distributions of system performance given specified configuration options used during DNN inference. Still, one has to be careful in generalizing FlexiBO to other systems. To reduce the effect of measurement noise in the DNN systems, we repeated each energy profiling 10 times on the NVIDIA device. Although due to the high evaluation cost we did not repeat the accuracy measurements, yet still, our initial experimental study also suggested that the accuracy measurements, i.e., network retraining, are less noisy in DNN systems.

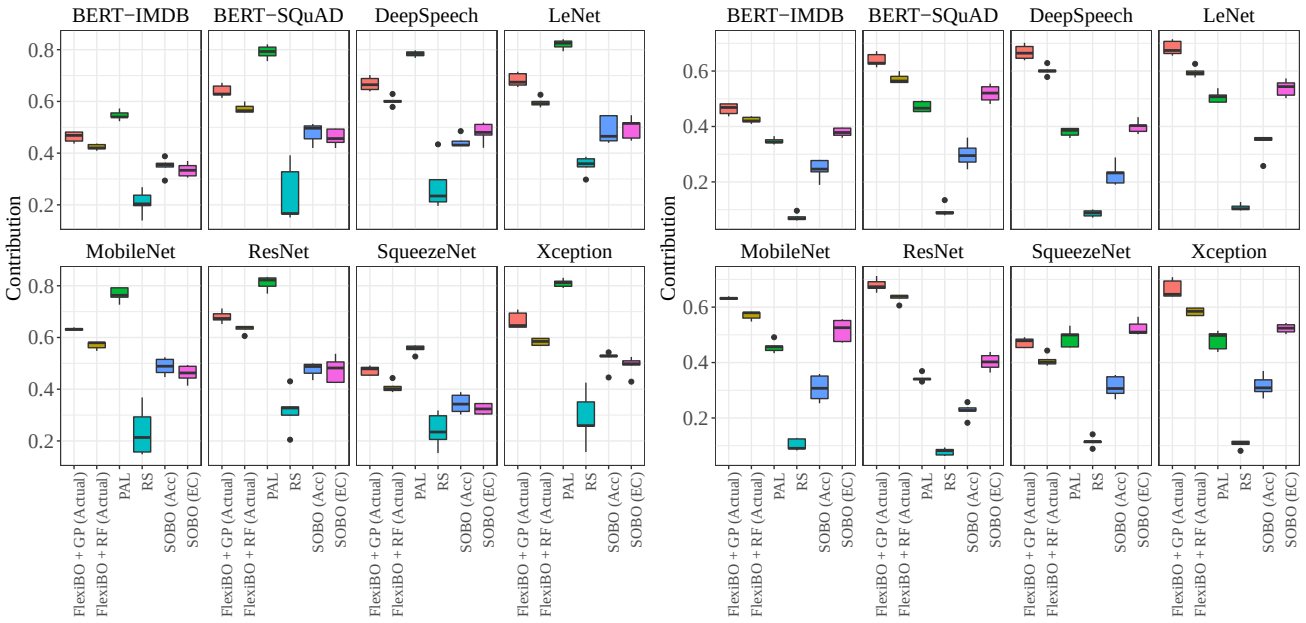
Although we measured the accuracy and energy consumption of 15 randomly selected configurations to initialize FlexiBO, we note that the results might vary if a different number of configurations are measured for initialization. To determine the quality of the Pareto metric for comparison, we used the contribution and diversity measures, which are both calculated relative to a reference point, and for our evaluations, we used the origin as the reference point. For PAL implementation, we fix the value of ϵ to 0.004% of the objective range. One might find different results when other values of ϵ are used; however, we do not vary the ϵ value due to the extremely high experimental cost associated therewith.

7. Related Work

Multi-objective optimization with preferences. Researchers have developed methods to incorporate preferences in multi-objective optimization using evolutionary methods (Deb & Sundar, 2006; Thiele et al., 2009; Kim et al., 2011); although these methods enable tradeoff for exploring the design space of systems (Kolesnikov et al., 2019), they are not sample efficient, which is an essential attribute for optimizing highly-configurable systems (Pereira et al., 2019; Jamshidi & Casale, 2016; Jamshidi et al., 2017b; 2018; Nair et al., 2018), especially for very large configuration spaces (Acher et al., 2019).

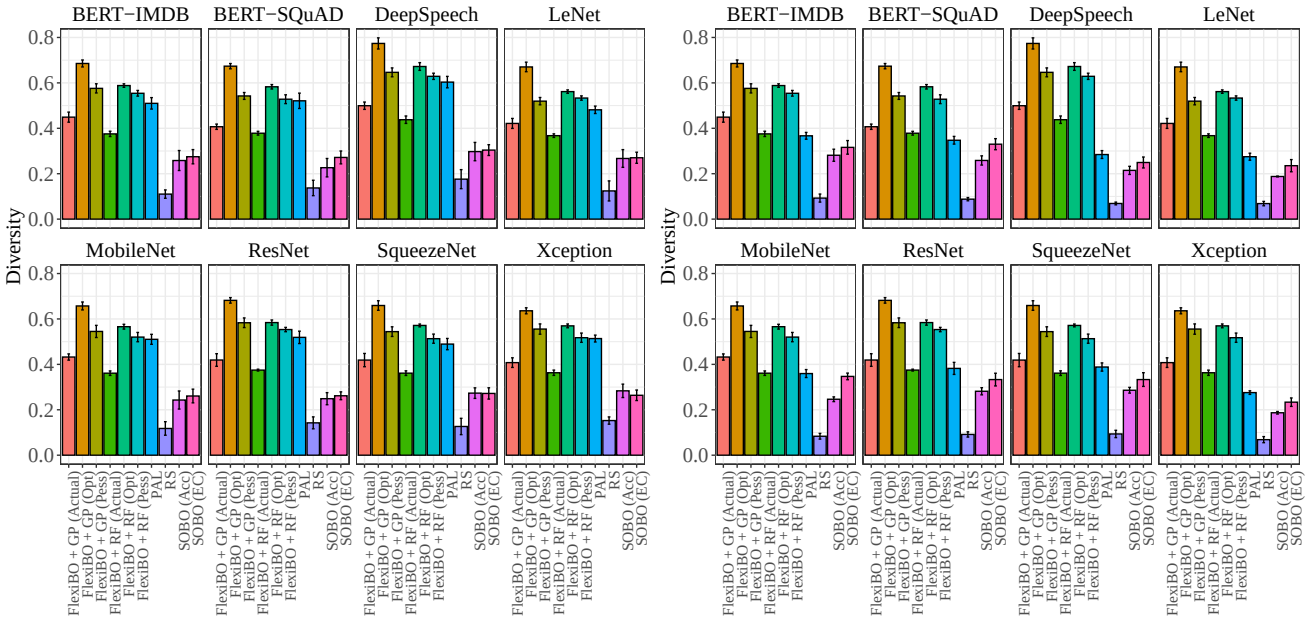
Despite that various methods have been recently proposed, which use surrogate functions to obtain objective approximations as a means to determine a Pareto front that account for preference constraints (Paria et al., 2018; Abdolshah et al., 2019), these methods need a defined set of preferences over objectives to be supplied, which is a condition for identifying a Pareto-set. In contrast, our method focuses on determining a Pareto front in the light of the cost of evaluation incorporated in an acquisition function, where the acquisition function, automatically and iteratively, determines the approximate balance of preference over objectives based on a tradeoff between the relative cost and information gain.

Multi-objective optimization with scalarizations. Different multi-objective optimization methods have been developed that use scalarizations to formulate a single-objective optimization problem such that the optimal solutions to single-objective optimization problems correspond



(a) (b)

Figure 8. The contribution indicator values of the Pareto fronts derived in (a) full-capacity mode and (b) time-budget mode, as obtained by different methods. The higher values indicate that the obtained Pareto front is closer to the surrogate of the true Pareto front.



(a) (b)

Figure 9. The diversity indicator values of the Pareto fronts derived in (a) full-capacity mode and (b) time-budget mode, as obtained by various methods applied to different DNN architectures. The higher values indicate the better quality of the respective Pareto fronts.

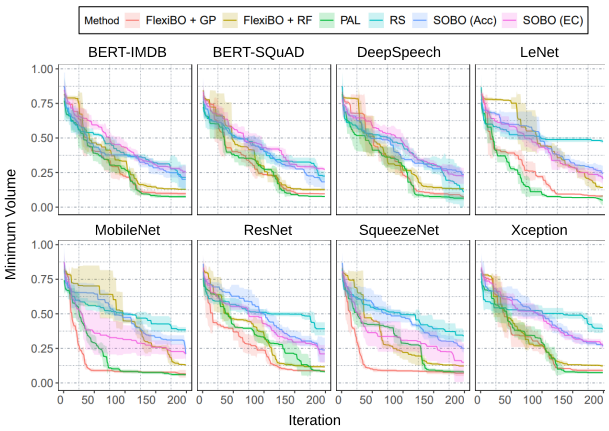


Figure 10. The Pareto volume obtained after each iteration for different optimization methods. The median (solid line) and interquartile range (shaded) of optimizing 5 runs are shown.

to Pareto-optimal solutions for multi-objective optimization problems. Random scalarizations have been used to discover the Pareto front (Knowles, 2006; Paria et al., 2018). Scalarization-based methods also include weighted product methods to obtain scalarized objectives (Deb, 2001) and utility functions (Roijers et al., 2013; Zintgraf et al., 2015; Roijers et al., 2017; 2018; Zintgraf et al., 2018). However, the scalarization approaches require the encoding of preference, as a priori, over objectives, and thus a scalarization approaches require multiple runs in order to obtain a high coverage of Pareto front solutions, which makes scalarization approaches not feasible and impractical for DNN optimization purposes.

Multi-objective optimization with Pareto front approximation. Several methods have been proposed to identify the complete Pareto front using multi-objective Bayesian optimization. For example, PESMO determines the Pareto front by reducing posterior entropy (Hernández-Lobato et al., 2016a; 2015). Another method, SMSego, uses the optimistic estimate of the objective function in order to select Pareto-optimal points and uses maximum hypervolume improvement to choose the next sample (Ponweiser et al., 2008). In addition, different gradient-based multi-objective optimization algorithms have been proposed for the purpose of finding a descent direction to optimize objectives (Schäffler et al., 2002; Désidéri, 2012). Therefore, these methods were then extended to utilize stochastic gradient descent (Poirion et al., 2017; Peitz & Dellnitz, 2018). Recently, active learning approaches have been proposed to approximate the surface of the Pareto front (Campigotto et al., 2013), through use of acquisition functions such as expected hypervolume improvement (Emmerich & Klinkenberg, 2008), and Sequential Uncertainty Reduction (SUR) (Picheny, 2015). Contemporary active learning approaches, like PAL, tend to approximate the Pareto front (Zuluaga et al., 2013) and further allow the user to set the prediction accuracy level (Zuluaga et al., 2016). Nevertheless, in contrast to our approach, these methods do not take into account the varying costs of the objective evaluations they complete. Our approach is reasonably similar to PAL but is different insofar as how the Pareto fronts are computed and meanwhile, further uses a different acquisition function as a means to incorporate the evaluation cost of optimization into its implementation.

Multi-objective, cross-layer, and hardware-aware optimization of DNNs. One of the largest difficulties in producing energy-efficient DNNs is the disconnect between the platform where the DNN is designed, developed, and tested, and the platform where it will eventually be deployed and the energy it consumes there (Guo, 2017; Chen et al.; Cai et al., 2017; Qi et al., 2016; Manotas et al., 2014; Sze et al., 2017; Chen et al., 2016). Therefore, hardware-aware multi-objective optimization approaches have been introduced (Zhu et al., 2018; Lokhmotov et al., 2018; Cai et al., 2018; Wu et al., 2019; Whatmough et al., 2019) that enable automatic optimization of DNNs in the joint space of architectures, hyperparameters, and even the computer system stack (Zela et al., 2018; Iqbal et al., 2019; Nardi et al., 2019; Hernández-Lobato et al., 2016b). Like these approaches, FlexiBO enables efficient multi-objective optimization in such joint configuration spaces. Works have also been done on multi-objective neural architecture search (NAS) (Kim et al., 2017; Dong et al., 2018; Liu et al., 2017) to optimize accuracy and limit resource consumption, whereby through limiting the search space, a fixed-length vector description is used (Kim et al., 2017). In addition, several approaches characterize runtime, power, and the energy of DNNs via analytical models, e.g., Paleo (Qi et al., 2016), NeuralPower (Cai et al., 2017), Eyeriss (Chen et al., 2016), and Delight (Rouhani et al., 2016). However, they either rely on proxies like inference time for energy estimation or otherwise extrapolate energy values from energy-per-operation tables. Due to the simplistic assumptions these methods rely upon, they cannot be used across different deployment platforms.

8. Conclusion

In this work, we developed a novel cost-aware Bayesian multi-objective optimization algorithm called FlexiBO. FlexiBO approximates the Pareto region and employs a novel cost-aware acquisition function that not only selects the next sample but also the most cost-effective objective to collect measurement. We carried out our experiments using 7 different DNN architectures for tasks of object detection, NLP and speech recognition, and optimized accuracy and energy consumption on a resource-constrained device.

The experimental results, which compared with the current state-of-the-art multi-objective optimization methods, confirmed that FlexiBO (i) consistently finds better configurations, (ii) produces superior Pareto fronts, and (iii) discovers optimal configurations at a lower cost. In parrticular, our results show that with the same time-budget, the contribution rate indicator and the diversity measures of FlexiBO + GP, on average, were 28.44% and 25.64% higher, respectively, than the state-of-the-art across all 7 architectures. Moreover, FlexiBO determines a Pareto front of similar quality were compared to the state-of-the-art with 80.23% less cost. An immediate future direction of this work is to compare FlexiBO with other Bayesian optimization frameworks such as Spearmint (Hernández-Lobato et al., 2016a), which offers multiple acquisition functions such as ParEGO (Knowles, 2006), SMSego (Ponweiser et al., 2008) and PESMO (Hernández-Lobato et al., 2016a).

Acknowledgments. This work was supported by AFRL and DARPA (FA8750-16-2-0042). This work is also partially supported by an ASPIRE grant from the Office of the Vice President for Research at the University of South Carolina. We would like to thank Marilyn Gartley for copy-editing the paper.

References

Abdolshah, M., Shilton, A., Rana, S., Gupta, S., and Venkatesh, S. Multi-objective bayesian optimisation with preferences over objectives. *arXiv preprint arXiv:1902.04228*, 2019.

Acher, M., Martin, H., Pereira, J., Blouin, A., Jézéquel, J.-M., Khelladi, D., Lesoil, L., and Barais, O. Learning very large configuration spaces: What matters for linux kernel sizes. 2019.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S., and Salomon, L. Performance indicators in multiobjective optimization. *Optimization Online*, 2018.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

Cai, E., Juan, D.-C., Stamoulis, D., and Marculescu, D. Neuralpower: Predict and deploy energy-efficient convolutional neural networks. *arXiv:1710.05420*, 2017.

Cai, H., Zhu, L., and Han, S. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Campigotto, P., Passerini, A., and Battiti, R. Active learning of pareto fronts. *IEEE transactions on neural networks and learning systems*, 25(3):506–519, 2013.

Cao, Y., Smucker, B. J., and Robinson, T. J. On using the hypervolume indicator to compare pareto fronts: Applications to multi-criteria optimal experimental design. *Journal of Statistical Planning and Inference*, 160:60–74, 2015.

Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., Cowan, M., Wang, L., Hu, Y., Ceze, L., et al. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pp. 578–594, 2018.

Chen, Y.-H., Yang, T.-J., Emer, J., and Sze, V. Understanding the limitations of existing energy-efficient design approaches for deep neural networks. *Energy*, 2(L1):L3.

Chen, Y.-H., Emer, J., and Sze, V. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. In *ACM SIGARCH Computer Architecture News*, volume 44, pp. 367–379. IEEE Press, 2016.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Deb, K. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

Deb, K. and Sundar, J. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pp. 635–642. ACM, 2006.

Désidéri, J.-A. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, J.-D., Cheng, A.-C., Juan, D.-C., Wei, W., and Sun, M. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 517–531, 2018.

Emmerich, M. and Klinkenberg, J.-w. The computation of the expected improvement in dominated hypervolume of pareto front approximations. *Rapport technique, Leiden University*, 34:7–3, 2008.

Guo, T. Towards efficient deep inference for mobile applications. *arXiv:1707.04610*, 2017.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hernández-Lobato, D., Hernandez-Lobato, J., Shah, A., and Adams, R. Predictive entropy search for multi-objective bayesian optimization. In *International Conference on Machine Learning*, pp. 1492–1501, 2016a.

Hernández-Lobato, J. M., Gelbart, M. A., Hoffman, M. W., Adams, R. P., and Ghahramani, Z. Predictive entropy search for bayesian optimization with unknown constraints. 2015.

Hernández-Lobato, J. M., Gelbart, M. A., Reagen, B., Adolf, R., Hernández-Lobato, D., Whatmough, P. N., Brooks, D., Wei, G.-Y., and Adams, R. P. Designing neural network hardware accelerators with decoupled objective evaluations. In *NIPS workshop on Bayesian Optimization*, pp. 10, 2016b.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv:1602.07360*, 2016.

Iqbal, M. S., Kotthoff, L., and Jamshidi, P. Transfer Learning for Performance Modeling of Deep Neural Network Systems. In *USENIX Conference on Operational Machine Learning*, Santa Clara, CA, 2019. USENIX Association. URL https://www.usenix.org/conference/opml19/presentation/iqbal.

Jamshidi, P. and Casale, G. An uncertainty-aware approach to optimal configuration of stream processing systems. In *Proc. Int'l Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2016. ISBN 978-1-5090-3433-8. doi: 10.1109/MASCOTS.2016.17. URL http://doi.acm.org/10.1145/2517349.2522727.

Jamshidi, P., Siegmund, N., Velez, M., Kästner, C., Patel, A., and Agarwal, Y. Transfer learning for performance modeling of configurable systems: An exploratory analysis. In *Proc. Int'l Conf. Automated Software Engineering (ASE)*. ACM, 2017a.

Jamshidi, P., Velez, M., Kästner, C., Siegmund, N., and Kawthekar, P. Transfer learning for improving model predictions in highly configurable software. In *Proc. Int'l Symp. Soft. Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2017b.

Jamshidi, P., Velez, M., Kästner, C., and Siegmund, N. Learning to sample: Exploiting similarities across environments to learn performance models for configurable systems. In *Proc. Int'l Symp. Foundations of Software Engineering (FSE)*. ACM, 2018.

Kim, J.-H., Han, J.-H., Kim, Y.-H., Choi, S.-H., and Kim, E.-S. Preference-based solution selection algorithm for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 16(1):20–34, 2011.

Kim, Y.-H., Reddy, B., Yun, S., and Seo, C. Nemo: Neuroevolution with multiobjective optimization of deep neural network for speed and accuracy. In *ICML 2017 AutoML Workshop*, 2017.

Knowles, J. Parego: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

Kolesnikov, S., Siegmund, N., Kästner, C., Grebhahn, A., and Apel, S. Tradeoffs in modeling performance of highly configurable software systems. *Software & Systems Modeling*, 18(3):2265–2283, 2019.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., and Kavukcuoglu, K. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*, 2017.

Lokhmotov, A., Chunosov, N., Vella, F., and Fursin, G. Multi-objective autotuning of mobilenets across the full software/hardware stack. In *Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning*, pp. 6. ACM, 2018.

Manotas, I., Pollock, L., and Clause, J. Seeds: a software engineer's energy-optimization decision support framework. In *Proceedings of the 36th International Conference on Software Engineering*, pp. 503–514. ACM, 2014.

Nair, V., Yu, Z., Menzies, T., Siegmund, N., and Apel, S. Finding faster configurations using flash. *IEEE Transactions on Software Engineering*, 2018.

Nardi, L., Koeplinger, D., and Olukotun, K. Practical design space exploration. In *2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 347–358. IEEE, 2019.

Paria, B., Kandasamy, K., and Póczos, B. A flexible framework for multi-objective bayesian optimization using random scalarizations. *arXiv preprint arXiv:1805.12168*, 2018.

Peitz, S. and Dellnitz, M. Gradient-based multiobjective optimization with uncertainties. In *NEO 2016*, pp. 159–182. Springer, 2018.

Pereira, J. A., Martin, H., Acher, M., Jézéquel, J.-M., Botterweck, G., and Ventresque, A. Learning software configuration spaces: A systematic literature review. *arXiv preprint arXiv:1906.03018*, 2019.

Picheny, V. Multiobjective optimization using gaussian process emulators via stepwise uncertainty reduction. *Statistics and Computing*, 25(6):1265–1280, 2015.

Poirion, F., Mercier, Q., and Désidéri, J.-A. Descent algorithm for nonsmooth stochastic multiobjective optimization. *Computational Optimization and Applications*, 68 (2):317–331, 2017.

Ponweiser, W., Wagner, T., Biermann, D., and Vincze, M. Multiobjective optimization on a limited budget of evaluations using model-assisted {S} -metric selection. In *International Conference on Parallel Problem Solving from Nature*, pp. 784–794. Springer, 2008.

Qi, H., Sparks, E. R., and Talwalkar, A. Paleo: A performance model for deep neural networks. 2016.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013.

Roijers, D. M., Zintgraf, L. M., and Nowé, A. Interactive thompson sampling for multi-objective multi-armed bandits. In *International Conference on Algorithmic DecisionTheory*, pp. 18–34. Springer, 2017.

Roijers, D. M., Zintgraf, L. M., Libin, P., and Nowé, A. Interactive multi-objective reinforcement learning in multi-armed bandits for any utility function. In *ALA workshop at FAIM*, volume 8, 2018.

Rouhani, B. D., Mirhoseini, A., and Koushanfar, F. Delight: Adding energy dimension to deep neural networks. In *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, pp. 112–117. ACM, 2016.

Schäffler, S., Schultz, R., and Weinzierl, K. Stochastic method for the solution of unconstrained vector optimization problems. *Journal of Optimization Theory and Applications*, 114(1):209–222, 2002.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pp. 527–538, 2018.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Proc. of 12th USENIX conference on Operating Systems Design and Implementation (OSDI)*, pp. 2951–2959, 2012.

Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

Thiele, L., Miettinen, K., Korhonen, P. J., and Molina, J. A preference-based evolutionary algorithm for multi-objective optimization. *Evolutionary computation*, 17(3): 411–436, 2009.

Whatmough, P. N., Zhou, C., Hansen, P., Venkataramanaiah, S. K., Seo, J.-s., and Mattina, M. Fixynn: Efficient hardware for mobile computer vision via transfer learning. *arXiv preprint arXiv:1902.11128*, 2019.

Williams, C. K. and Rasmussen, C. E. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10734–10742, 2019.

Zela, A., Klein, A., Falkner, S., and Hutter, F. Towards automated deep learning: Efficient joint neural architecture and hyperparameter search. *arXiv preprint arXiv:1807.06906*, 2018.

Zhu, Y., Mattina, M., and Whatmough, P. Mobile machine learning hardware at arm: A systems-on-chip (soc) perspective. *arXiv:1801.06274*, 2018.

Zintgraf, L. M., Kanters, T. V., Roijers, D. M., Oliehoek, F. A., and Beau, P. Quality assessment of morl algorithms: A utility-based approach. In *Benelearn 2015: Proceedings of the Twenty-Fourth Belgian-Dutch Conference on Machine Learning*, 2015.

Zintgraf, L. M., Roijers, D. M., Linders, S., Jonker, C. M., and Nowé, A. Ordered preference elicitation strategies for supporting multi-objective decision making. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1477–1485. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Zitzler, E. and Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.

Zuluaga, M., Sergent, G., Krause, A., and Püschel, M. Active learning for multi-objective optimization. In *International Conference on Machine Learning*, pp. 462–470, 2013.

Zuluaga, M., Krause, A., and Püschel, M. ε-pal: an active learning approach to the multi-objective optimization problem. *The Journal of Machine Learning Research*, 17 (1):3619–3650, 2016.