Journal of Artificial Intelligence Research 77 (2023) 645-682 Submitted 08/2022; published 05/2023

FlexiBO: A Decoupled Cost-Aware Multi-Objective Optimization Approach for Deep Neural Networks

Md Shahriar Iqbal MIQBAL@EMAIL.SC.EDU
Jianhai Su SUJ@EMAIL.SC.EDU
University of South Carolina, Columbia, SC, USA

Lars Kotthoff LARSKO@UWYO.EDU
University of Wyoming, Laramie, WY, USA

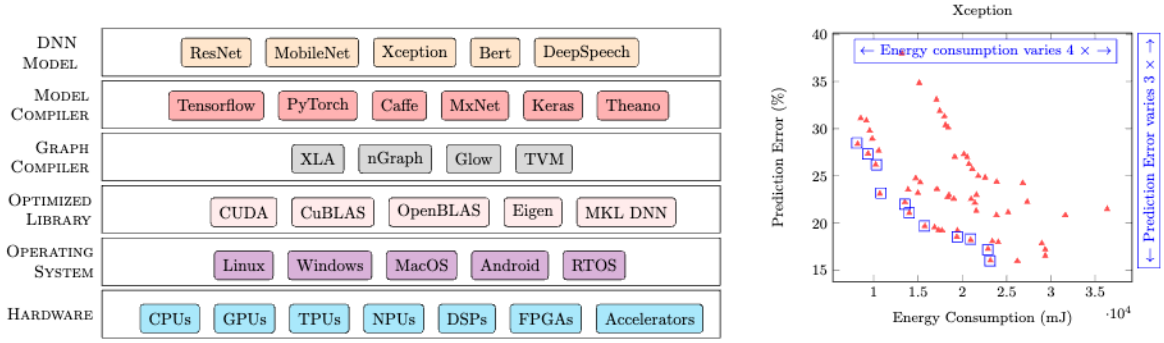
Pooyan Jamshidi PJAMSHID@CSE.SC.EDU
University of South Carolina, Columbia, SC, USA

Abstract

The design of machine learning systems often requires trading off different objectives, for example, prediction error and energy consumption for deep neural networks (DNNs). Typically, no single design performs well in all objectives; therefore, finding Pareto-optimal designs is of interest. The search for Pareto-optimal designs involves evaluating designs in an iterative process, and the measurements are used to evaluate an acquisition function that guides the search process. However, measuring different objectives incurs different costs. For example, the cost of measuring the prediction error of DNNs is orders of magnitude higher than that of measuring the energy consumption of a pre-trained DNN as it requires re-training the DNN. Current state-of-the-art methods do not consider this difference in objective evaluation cost, potentially incurring expensive evaluations of objective functions in the optimization process. In this paper, we develop a novel decoupled and cost-aware multi-objective optimization algorithm, which we call *Flexible Multi-Objective Bayesian Optimization* (FlexiBO) to address this issue. For evaluating each design, FlexiBO selects the objective with higher relative gain by weighting the improvement of the hypervolume of the Pareto region with the measurement cost of each objective. This strategy, therefore, balances the expense of collecting new information with the knowledge gained through objective evaluations, preventing FlexiBO from performing expensive measurements for little to no gain. We evaluate FlexiBO on seven state-of-the-art DNNs for image recognition, natural language processing (NLP), and speech-to-text translation. Our results indicate that, given the same total experimental budget, FlexiBO discovers designs with 4.8% to 12.4% lower hypervolume error than the best method in state-of-the-art multi-objective optimization.

1. Introduction

Recent developments of deep neural networks (DNNs) have sparked a growing demand for pushing the deployment of artificial intelligence applications from the cloud to a wide variety of edge and IoT devices. These devices are closer to data and information generation sources, so they provide a better user experience, for instance, latency and throughput sensitivity, security, etc. Compared to data centers, the edge devices are more resource-constrained and may not even be able to host these compute expensive DNN models. Therefore, designing



(a) DNN System Stack

(b) Performance distribution with different design options

Figure 1: (a) The system stack of a DNN comprises six components. (b) Performance variation of several hundred designs for an image recognition DNN system, SqueezeNet, deployed on resource-constrained Nvidia Jetson TX2 hardware while performing inference on 5,000 test images. We observe a substantial variation for each performance objective across different designs; in particular, prediction error varies 3×, and energy consumption varies 4×. Designs marked with blue rectangles optimally trade off both objectives.

energy-efficient DNNs is critical for the successful deployment of DNNs to these devices with limited resources. On the one hand, high inference error often leads to application failures (Pei et al., 2017; Sun et al., 2018), but on the other hand, it is essential to reduce the number of computation cycles and/or memory footprints of DNNs to conserve energy (Sze et al., 2017). In addition to DNN models, a number of components in the DNN system stack (see Figure 1a) must work together for the seamless deployment of energy-efficient DNNs without compromising inference accuracy (Gadepally et al., 2019; Reuther et al., 2019).

There exist 100s, if not 1000s, of design options from each component across the DNN system stack that impact the computational and memory requirements of DNNs and make them difficult to deploy effectively. One of the key challenges in designing an optimal DNN system is to efficiently explore the vast design space, with non-trivial interactions of options from different components across the system stack, for example, CPU frequency, GPU frequency, number of epochs, etc. Additionally, no single design usually performs well for all performance objectives (see Figure 1b). Therefore, we must identify designs that provide optimal trade-offs – *Pareto optimal* designs.

Previous work has focused on neural architecture search (NAS) techniques that can efficiently locate Pareto optimal designs. NAS approaches like Nemo (Kim et al., 2017), hierarchical representations (HR) (Liu et al., 2017), and DPP-NET (Dong et al., 2018) can be categorized according to three different criteria: (i) the *Search Space*, (ii) the *Optimization Method*, and (iii) the *Candidate Evaluation Method*. Unfortunately, the effectiveness of NAS approaches largely depends on selecting a quality search space to reduce the complexity of search that requires significant prior knowledge that is difficult to find in practice, which also indicates that they are not suitable for different platforms. Much recent work has focused on multi-objective Bayesian optimization (MOBO) approaches like ParEGO

Table 1: Comparison of FlexiBO to related state-of-the-art methods that can be used to identify Pareto optimal solutions.

Method		Method Type	Search Strategy	Evaluation Strategy	Cost Awareness
NEMO	(Kim et al., 2017)	NAS	Gradient Based	Coupled	✗
DPP-NET	(Dong et al., 2018)	NAS	Gradient Based	Coupled	✗
HR	(Liu et al., 2017)	NAS	Gradient Based	Coupled	✗
ParEGO	(Knowles, 2006)	MOBO	Random Scalarization	Coupled	✗
SMSego	(Ponweiser et al., 2008)	MOBO	Hypervoume Improvement	Coupled	✗
PAL	(Zuluaga et al., 2013)	MOBO	Predictive Uncertainty	Coupled	✗
MESMO	(Belakaria et al., 2019)	MOBO	Output Space Entropy	Coupled	✗
PESMO	(Hernández-Lobato et al., 2015)	MOBO	Input Space Entropy	Coupled	✗
PESMO-DEC	(Hernández-Lobato et al., 2015)	MOBO	Input Space Entropy	De-coupled	✗
CA-MOBO	(Abdolshah et al., 2019a)	MOBO	Chebyshev Scalarization	Coupled	✓
FlexiBO		MOBO	Volume of the Pareto region	De-coupled	✓

(Knowles, 2006), SMSego (Ponweiser et al., 2008), PAL (Zuluaga et al., 2013), PESMO (Hernández-Lobato et al., 2016), MESMO (Belakaria et al., 2019), MESMOC (Belakaria et al., 2020) etc. to find the Pareto optimal designs that can be used for hyperparameter tuning. MOBO approaches iteratively use the uncertainty captured by a probabilistic model (also known as the *surrogate model*) of the process to be optimized to compute the values of an *acquisition function*. The acquisition function is an approximation of the complex unknown function to be optimized, which is much faster and cheaper to evaluate. The optimum of the acquisition function provides an effective heuristic for identifying a promising design for which to evaluate the objectives. Nevertheless, there are limitations to these MOBO approaches. For instance, scalarization-based approaches (ParEGO) tend to suffer from sub-optimality, algorithms to optimize hypervolume-based acquisition function (SMSego) scale poorly when the input dimensionality increases, methods that rely on entropy-based acquisition functions (PESMO, MESMO, MESMOC) are computationally expensive, and PAL is simple to design but also requires a lot of computation at each iteration.

Furthermore, most of these MOBO methods are cost unaware and consider the objective evaluation costs uniform. As a result, existing approaches use a *coupled evaluation strategy* to evaluate the design selected by the optimizers across all objectives at each iteration. However, the cost of objective evaluations can be non-uniform in practice, for example, optimizing prediction error in DNN systems is much more expensive than making predictions with a pre-trained DNN with different deployment system design options as that involves re-training the whole DNN. For higher optimization performance, one challenge for iterative optimizers is to balance *exploiting* the knowledge gained from the evaluations with *exploring* regions in the search space where the landscape is unknown and might hold better designs. The cost-unawareness of objective functions evaluation makes this even more challenging under a limited experimentation budget, such as when using DNNs deployed on production or resource-constrained devices. To overcome this issue, CA-MOBO (Abdolshah et al., 2019a) recently proposed a cost-aware approach to identify the Pareto optimal designs by avoiding the designs with high evaluation costs in the design space. Nonetheless, this can lead to aggressive exploitation behavior and generate sub-optimal designs. PESMO-DEC introduces a *decoupled* evaluation strategy where only a subset of objectives at any given location is evaluated instead of evaluating all the objectives at each iteration. It shows that the decoupled evaluation strategy provides significant improvements over a coupled evaluation, particularly when the experimentation budget is limited. This improvement is because

in the decoupled setting, PESMO-DEC identifies the most difficult objectives and evaluates them more times. However, selecting designs for evaluation without considering their individual objective evaluation costs can result in inefficient resource utilization, especially when the difference in complexity is insignificant.

To address these limitations, we propose the cost-aware decoupled MOBO approach FlexiBO, which explicitly considers non-uniform objective evaluation costs and evaluates expensive objectives only if the gain of information is worth it. FlexiBO extends the concepts of the state-of-the-art active learning algorithm PAL (Zuluaga et al., 2013) and PESMO-DEC (Hernández-Lobato et al., 2015). To our knowledge, this is the first approach to propose a cost-aware decoupled evaluation strategy for MOBO. To formalize the notion of non-uniform evaluation costs of objectives, we define *objective evaluation cost* in terms of computation time. Our acquisition function incorporates the uncertainty of the surrogate model's predictions and objective evaluation costs to balance exploration and exploitation and iteratively improve the quality of the Pareto optimal design space, also known as the *Pareto region*. It selects the objectives across which the design will be evaluated in addition to selecting the next design to evaluate. Consequently, we explicitly trade off the additional information obtained through an evaluation with the cost of obtaining it, ensuring we do not perform costly evaluations for little potential gain. By avoiding costly evaluations, we improve the efficiency of the search for Pareto optimal designs. We demonstrate FlexiBO's promise through a comprehensive experimental evaluation on a range of different benchmarks. While we focus on DNNs, our proposed approach is general and can be extended to other applications.

1.1 Contributions

In summary, our contributions are as follows.

- We propose FlexiBO: a cost-aware approach for multi-objective Bayesian optimization that selects *a design* and *an objective* for evaluation. It makes a trade-off between the additional information gained through an evaluation and the cost incurred as a result of the evaluation (Section 5).
- We comprehensively evaluate FlexiBO on seven DNN architectures from three different domains and compare its performance to ParEGO (Knowles, 2006), SMSego (Ponweiser et al., 2008), PAL (Zuluaga et al., 2013), and PESMO,PESMO-DEC (Hernández-Lobato et al., 2016), and CA-MOBO (Abdolshah et al., 2019a). (Section 6). The dataset and scripts to reproduce our findings are available at `https://github.com/softsys4ai/FlexiBO`.

2. Motivation

In this section, we discuss our motivation to propose a *cost-aware* and *decoupled* evaluation strategy. Based on the cost distribution assumptions and evaluation strategy, existing MOBO techniques can be subdivided into the following categories: (I) cost-unaware coupled, (II) cost-aware coupled, and (III) cost-unaware decoupled approaches. Unlike cost-unaware

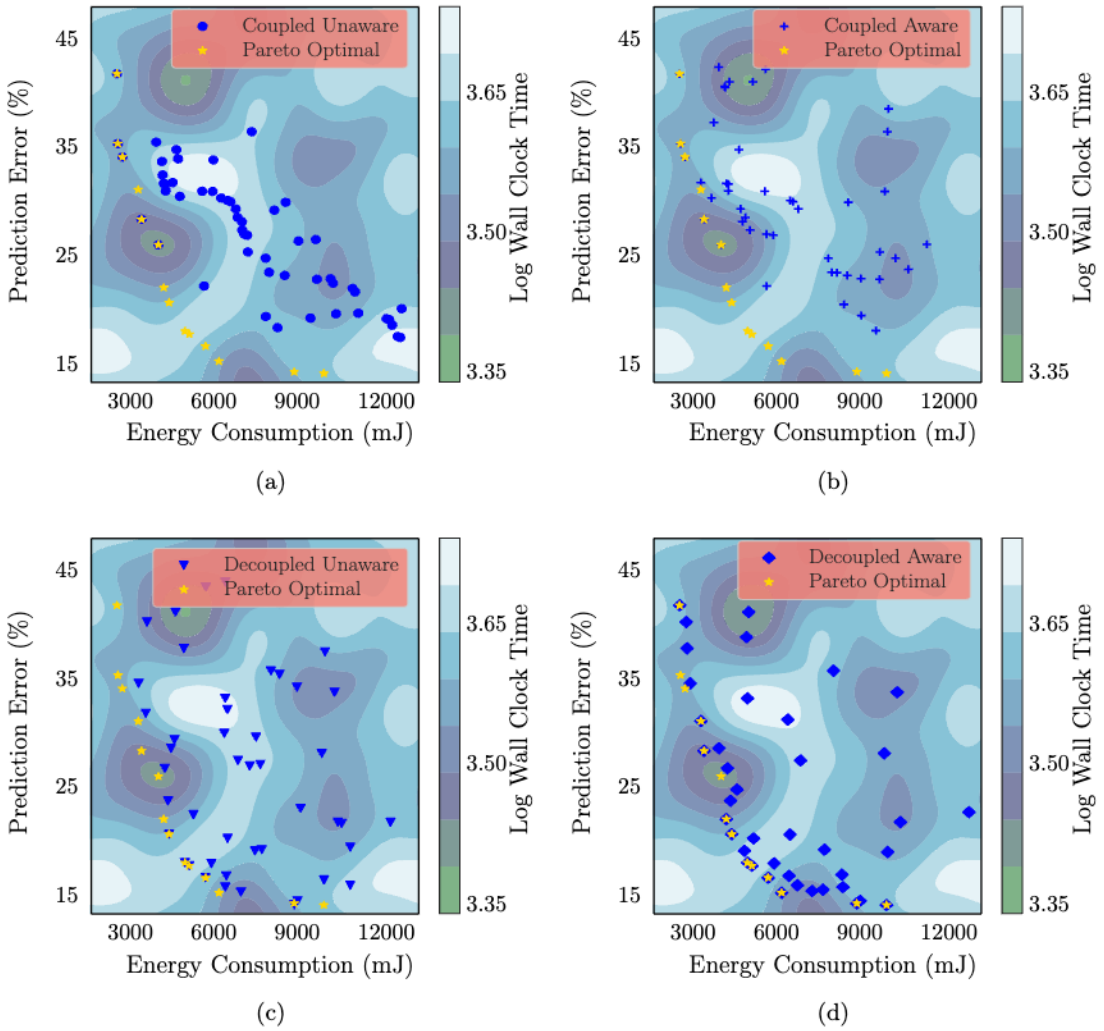


Figure 2: (a) Cost-unaware coupled approaches waste resources by evaluating designs with higher evaluation costs. (b) Cost-aware coupled approaches can suffer poor performance if the Pareto optimal designs can only be found by evaluating objectives with high evaluation costs. (c) Cost-unaware decoupled approaches can invest a lot of resources in evaluating designs with low quality (high prediction error and high energy consumption) and do not perform well across both objectives. (d) Cost-aware decoupled approaches find designs with better quality compared to other approaches. Yellow stars indicate the location of Pareto optimal designs. The color indicates the evaluation cost (log wall clock time) of a particular design.

approaches, cost-aware approaches assume the cost of evaluating different objectives is non-uniform. Decoupled approaches consider only a subset of objectives for evaluation at each iteration in the Bayesian optimization loop whereas coupled approaches evaluate all objectives. To our knowledge, the approach proposed in this paper addresses a gap in the

existing MOBO literature on cost-aware decoupled techniques. To show the clear advantage of our proposed cost-aware decoupled approach, we performed a sandbox experiment to optimize the prediction error and energy consumption of the image recognition DNN system SqueezeNet for the CIFAR-10 dataset deployed on an Nvidia JETSON TX2 device for inference on 5,000 test images. We use 8 Nvidia Tesla K80 GPUs deployed on Google Cloud for training with 45,000 training images. We tunned a small subset of design options from different layers of the system stack – CPU frequency and GPU frequency from the hardware layer, swappiness from the operating system layer, memory growth from the model compiler layer, and filter size, number of filters, and number of epochs from the DNN model layer. We use PAL as a cost-unaware coupled approach, CA-MOBO as a cost-aware coupled approach, PESMO-DEC as a cost-unaware decoupled approach, and FlexiBO as a cost-aware decoupled approach.

Cost-unaware coupled approaches are not sample (design) efficient for budget-constrained applications as they do not make the best utilization of resources by evaluating the selected designs across all objectives even for little or no gain. Figure 2a shows that most of the designs selected by the cost-unaware coupled approach are not close to the Pareto front and are concentrated in regions with high evaluation costs. It also has poor coverage of the objective space.

Cost-aware coupled approaches consider different evaluation costs for different designs. As such approaches only evaluate cheap designs, good parts of the search space may be missed. Figure 2b shows that the cost-aware coupled strategy focuses on the cheap regions of the search space here and misses Pareto's optimal designs in expensive regions.

Cost-unaware decoupled approaches evaluate the more complex objectives a higher number of times than the less complex objectives. However, Figure 3a and Figure 3b show that both the objectives (e.g., prediction error and energy consumption) have the same complexity. Cost-unaware approaches are not particularly effective in such cases and produce suboptimal results as shown in Figure 2c. This occurs as a result of them not making the best use of resources by evaluating a large number of low-quality designs (high prediction error and high energy consumption) for little information gain.

Cost-aware decoupled approaches evaluate designs in any region if the information gain is large enough given the objective evaluation cost. We observe that our cost-aware decoupled approach performs better than the other approaches and identifies more points on the Pareto front (see Figure 2d).

3. Related Work

We now discuss different directions of related work for multi-objective optimization.

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

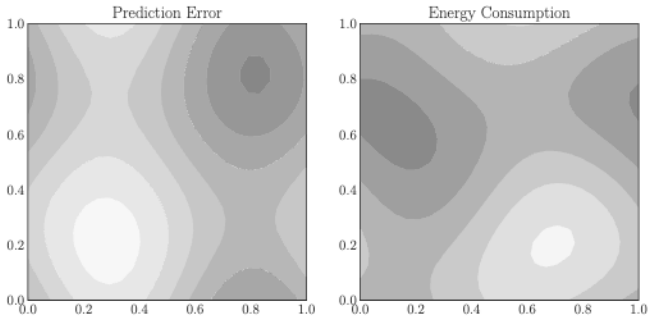


Figure 3: Contour curves for (a) prediction error and (b) energy consumption of SqueezeNet by varying CPU Frequency and Number of Filters while keeping the other design options fixed. Decoupled unaware approaches perform poorly when objectives with different costs have the same complexity (both non-linear).

stack (Zela et al., 2018; Iqbal et al., 2019; Nardi et al., 2019; Hernández-Lobato et al., 2016). Like these approaches, FlexiBO enables efficient multi-objective optimization in such joint configuration spaces. Multi-objective neural architecture search (NAS) (Kim et al., 2017; Dong et al., 2018; Liu et al., 2017, 2018) aims to optimize accuracy and limit resource consumption, for example, by limiting the search space (Kim et al., 2017). Several approaches characterize runtime, power, and the energy consumption of DNNs via analytical models, for example, 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 consumption or extrapolate energy values from energy-per-operation tables. They, therefore, cannot be used across different deployment platforms.

Multi-Objective Optimization with Different Acquisition Functions. A large body of research identifies the complete Pareto front using entropy-based acquisition functions. For example, MESMO (Wang & Jegelka, 2017; Belakaria et al., 2019), MESMOC (Belakaria et al., 2020), and PESMO (Hernández-Lobato et al., 2016) determine the Pareto front by reducing posterior entropy. SMSego uses the maximum hypervolume improvement acquisition function to choose the next sample (Ponweiser et al., 2008). Different gradient-based multi-objective optimization algorithms have been proposed to optimize objectives more efficiently (Schäffler et al., 2002; Désidéri, 2012). These methods were extended to use stochastic gradient descent (Poirion et al., 2017; Peitz & Dellnitz, 2018). Active learning approaches have been proposed to approximate the surface of the Pareto front (Campigotto et al., 2013) through the use of acquisition functions such as expected hypervolume improvement (Emmerich & Klinkenberg, 2008) and sequential uncertainty reduction (Picheny, 2015). Contemporary active learning approaches like PAL and ϵ-PAL tend to approximate the Pareto front (Zuluaga et al., 2013, 2016) using the maximum diagonal of the uncertainties in the objective space as the acquisition function. However, these methods do not take into account the varying costs of the evaluations of the objective functions are expensive.

Multi-Objective Optimization With Preferences. Some methods use *preferences* in multi-objective optimization with evolutionary methods (Deb & Sundar, 2006; Thiele et al., 2009; Kim et al., 2011); although these methods enable the user to guide the exploration of the design space of systems (Kolesnikov et al., 2019), they are not sample-efficient, which is essential for optimizing highly-configurable systems (Pereira et al., 2019; Jamshidi & Casale, 2016; Jamshidi et al., 2017, 2018; Nair et al., 2018), particularly for very large configuration spaces (Acher et al., 2019). Recently, methods that use surrogate models for optimization with preferences have been proposed (Paria et al., 2018; Abdolshah et al., 2019b). These methods require the user to specify a preference manually and are not cost-efficient.

Multi-Objective Optimization With Scalarizations. Different multi-objective optimization methods have been developed that use scalarizations to combine multiple objectives into one such that optimal solutions correspond to Pareto-optimal solutions. Examples include ParEGO, which uses random scalarizations (Knowles, 2006), weighted product methods (Deb, 2001), and utility functions (Roijers et al., 2013, 2017, 2018; Zintgraf et al., 2018). A major disadvantage of the scalarization approach is that not all Pareto optimal solutions can be recovered without further assumptions (e.g., convexity) on the objectives. Therefore, solutions obtained by scalarization approaches tend to be sub-optimal.

Cost-Aware Multi-Objective Optimization Approaches. Recently, different cost-aware methods (Abdolshah et al., 2019a; Lee et al., 2020) have been proposed that incorporate the evaluation costs of objectives into account. They assign costs to designs in the design space and attempt to identify an optimal Pareto front by avoiding costly designs, thereby selecting cheap designs for evaluation. These methods are either orthogonal or complementary to our approach. FlexiBO is a decoupled approach where we trade off the evaluation cost of an objective with the amount of information that can be gained.

4. Background and Definitions

In this section, we review MOBO and Pareto optimality and introduce the terminology and notation used in the rest of the paper. Table 9 in the appendix lists the symbols and their descriptions used throughout the paper.

Bayesian Optimization. Bayesian Optimization (BO) is an efficient framework to solve global optimization problems using black-box evaluations of expensive objective functions (Jones, Schonlau, & Welch, 1998). Let $\mathfrak{X} \subset \mathbb{R}^d$, where $d \in \mathbb{N}$, be a *finite design space*. For single-objective Bayesian optimization (SOBO), we are given a real-valued objective function $f : \mathfrak{X} \to \mathbb{R}$, which can be evaluated at each design $\boldsymbol{x} \in \mathfrak{X}$ to produce an evaluation $y = f(\boldsymbol{x})$. Each evaluation of $\boldsymbol{x}$ is expensive in terms of the consumed resources. The main goal is to find a design $\boldsymbol{x}^* \in \mathfrak{X}$ that optimizes f by performing a limited number of function evaluations. BO approaches use a cheap surrogate model learned from training data obtained using past function evaluations. They intelligently select designs for evaluation by searching over the surrogate model, trading off exploration and exploitation to quickly direct the search toward an optimal design.

Acquisition Function. This is used to score the utility of evaluating a candidate design $\boldsymbol{x} \in \mathfrak{X}$ based on the statistical model. Some popular acquisition functions in the SOBO litera-

ture include expected improvement (EI) (Emmerich & Klinkenberg, 2008), upper confidence bound (UCB) (Srinivas et al., 2012), predictive entropy search (PES) (Hernández-Lobato et al., 2014), and max-value entropy search (MES) (Wang & Jegelka, 2017).

MOBO. In MOBO, the aim is to find a set of designs that simultaneously optimizes n possibly conflicting objective functions $\boldsymbol{f} = f_1, \ldots, f_n$, where $n \geq 2$ and $f_i : \mathfrak{X} \rightarrow \mathbb{R}$ for $1 \leq i \leq n$. Each evaluation of a design $\boldsymbol{x} \in \mathfrak{X}$ produces a vector of objective values $\boldsymbol{y} = (y_1, ..., y_n)$, where $y_i = f_i(\boldsymbol{x})$ for $1 \leq i \leq n$.

Pareto-Optimality. It is generally not possible to find a design that optimizes each objective equally, but instead, there is a trade-off between them. Pareto optimal designs represent the best compromises across all objectives. In the context of maximization, a design $\boldsymbol{x}$ is said to *dominate* another design $\boldsymbol{x}'$, formally, $\boldsymbol{x} \succeq \boldsymbol{x}'$ if $f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}')$ for $1 \leq i \leq n$. A design $\boldsymbol{x} \in \mathfrak{X}$ is called *Pareto-optimal* if it is not dominated by any other designs $\boldsymbol{x}' \in \mathfrak{X}$, where $\boldsymbol{x} \neq \boldsymbol{x}'$. The set of designs $\mathfrak{X}^*$ is called the optimal Pareto set and a *hyperplane*[1] passing through the corresponding set of function values $\mathfrak{F}^*$ is called the *Pareto front*.

Surrogate Model Surrogate models $\mathfrak{M}_i$ for $1 \leq i \leq n$ are used to approximate the function to optimize, which is usually computationally expensive to evaluate and not available in closed form. Surrogate models are trained with evaluations of a small subset of the design space $\mathfrak{X}$ and are used to predict the value of the objective function using $\hat{f}(\boldsymbol{x}) = \boldsymbol{\mu}(\boldsymbol{x})$ with estimation uncertainty $\boldsymbol{\sigma}(\boldsymbol{x})$ for each design $\boldsymbol{x}$. The uncertainty region $R(\boldsymbol{x})$ of a design $\boldsymbol{x}$ is defined as a hyper-rectangle of the width of the confidence region using $\boldsymbol{\mu}(\boldsymbol{x})$ and $\boldsymbol{\sigma}(\boldsymbol{x})$ (formally defined later).

Optimistic and Pessimistic Pareto front. Each design $\boldsymbol{x}$ in the design space $\mathfrak{X}$ is assigned an uncertainty region $R(\boldsymbol{x})$ using the predictions of the objective functions $\boldsymbol{f}$ from the surrogate models $\mathfrak{M}$. Figure 4 shows an example of uncertainty region of a design and its maximum value $\max(R(\boldsymbol{x}))$ and minimum value $\min(R(\boldsymbol{x}))$ for $n = 2$ objectives. The maximum value of the uncertainty region $\max(R(\boldsymbol{x}))$ and minimum value of the uncertainty region $\min(R(\boldsymbol{x}))$ of a design $\boldsymbol{x}$ are regarded as the optimistic and pessimistic value of $\boldsymbol{x}$, respectively. A hyperplane passing through the non-dominated optimistic values of $\boldsymbol{x}$ is considered the optimistic Pareto front $\mathfrak{F}_{opt}$. Similarly, a pessimistic Pareto front $\mathfrak{F}_{pess}$ is constructed by a hyperplane passing through the pessimistic values of $\boldsymbol{x}$.

Pareto Region. The region bounded by the optimistic Pareto front $\mathfrak{F}_{opt}$ and pessimistic Pareto front $\mathfrak{F}_{\texttt{pess}}$ is defined as the Pareto region P_R (shown as the blue shaded region in Figure 5).

Objective evaluation cost. Objective evaluation cost $\theta_i(\boldsymbol{x})$ of a design $\boldsymbol{x}$ is the computational effort required to evaluate design $\boldsymbol{x}$ for an objective f_i.

5. FlexiBO: Flexible Multi-Objective Bayesian Optimization

In this section, we explain the technical details of our proposed Flexible Multi-Objective Bayesian Optimization (FlexiBO) algorithm. FlexiBO aims to identify the optimal Pareto front $\mathfrak{F}^*$ by evaluating a small subset of designs in the design space $\mathfrak{X}$ that uses a cost-aware

1. A subspace of the design space whose dimension is one less than the design space.

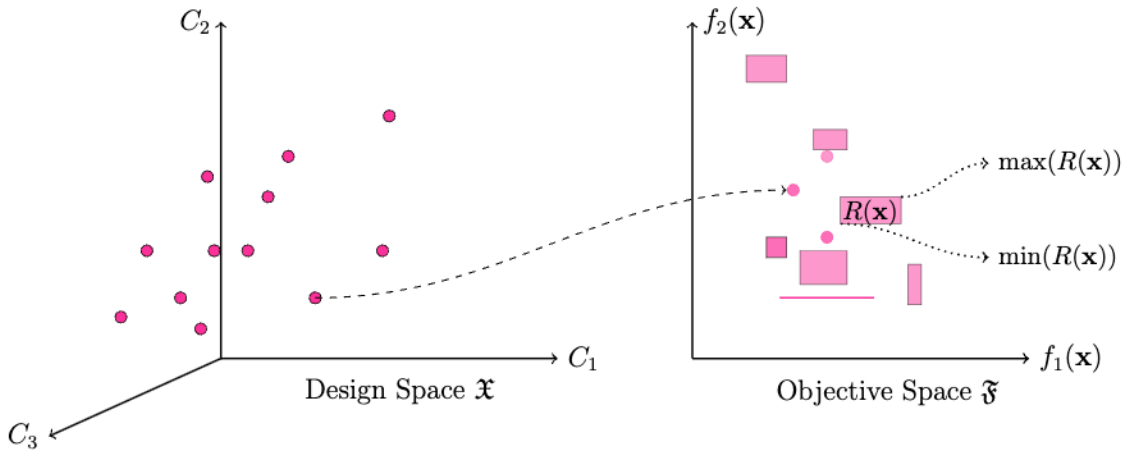


Figure 4: The design space $\mathfrak{X}$ is mapped to the objective space $\mathfrak{F}$ for $n = 2$. The objective space shows examples of uncertainty regions $R(\boldsymbol{x})$ for different designs $\boldsymbol{x}$.

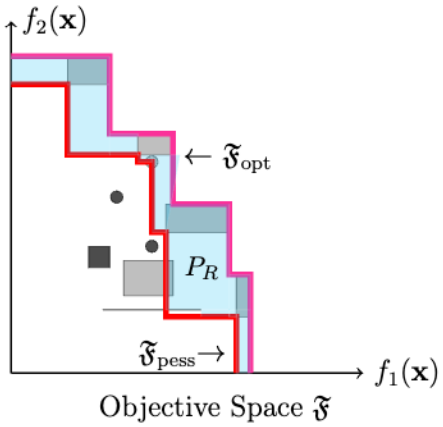


Figure 5: The objective space is showing examples of Pareto fronts $\mathfrak{F}_{pess}$ and $\mathfrak{F}_{opt}$ passing through a subset of non-dominated designs. Here, non-dominated designs are shown in gray and dominated designs are shown in black.

acquisition function to incorporate the evaluation costs of each objective in the standard Bayesian optimization framework. Given the same budget θ_T, the cost-awareness of the acquisition function enables FlexiBO to sample the search space more efficiently compared to other state-of-the-art approaches.

5.1 Algorithm Design

FlexiBO is an active learning algorithm that selects a sequence of designs $(\boldsymbol{x}_1, ..., \boldsymbol{x}_T)$ in the design space $\mathfrak{X}$ for evaluation to determine the Pareto-optimal designs; the designs classified as Pareto-optimal are then returned as the prediction $\hat{\mathfrak{F}}^*$ for $\mathfrak{F}^*$. Rather than evaluating

each design $\boldsymbol{x}$ against all objectives f_i for $1 \leq i \leq n$, our cost-aware approach iteratively evaluates a selected design $\boldsymbol{x}$ only across the most informative objective. FlexiBO evaluates a design $\boldsymbol{x}$ across an objective f_i if the change in hypervolume of the Pareto region is large enough compared to the objective evaluation cost θ_i. This allows FlexiBO to avoid expensive measurements for little or no change in hypervolume and to only evaluate across an objective when the change in hypervolume is worthy compared to the evaluation cost. Formally, FlexiBO is a cost-aware multi-objective optimization approach that iteratively and adaptively selects a sequence of designs and objectives $((\boldsymbol{x}_1, f_{1,i}), ..., (\boldsymbol{x}_T, f_{T,i}))$ for $1 \leq i \leq n$ across which the selected designs are evaluated to predict the Pareto front $\hat{\mathfrak{F}}^*$.

We then fit a separate surrogate model $\mathfrak{M}_\mathfrak{i}$ for each objective function f_i for $1 \leq i \leq n$. We select m designs set $\mathfrak{X}_\mathfrak{m}$ from the design space $\mathfrak{X}$ using Monte-Carlo sampling (Shapiro, 2003). The objective values of a design $\boldsymbol{x}$ that has not been evaluated across any objective are estimated by $\hat{\boldsymbol{f}}(\boldsymbol{x}) = \boldsymbol{\mu}(\boldsymbol{x}) = (\mu_1(\boldsymbol{x}), \ldots, \mu_n(\boldsymbol{x}))$, and the associated uncertainty is estimated by $\boldsymbol{\sigma}(\boldsymbol{x}) = (\sigma_1(\boldsymbol{x}), \ldots, \sigma_n(\boldsymbol{x}))$. If a design $\boldsymbol{x}$ is evaluated across an objective f_i, the associated uncertainty is zero against f_i. At each iteration t, we use the $\boldsymbol{\mu}_t(\boldsymbol{x})$ and $\boldsymbol{\sigma}_t(\boldsymbol{x})$ values to determine the uncertainty region $R_t(\boldsymbol{x})$ for each design $\boldsymbol{x} \in \mathfrak{X}_\mathfrak{m}$. We define the uncertainty region associated with a prediction of the surrogate model as follows:

$$R_t(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{\mu}_t(\boldsymbol{x}) - \sqrt{\beta_t}\boldsymbol{\sigma}_t(\boldsymbol{x}) \leq \boldsymbol{y} \leq \boldsymbol{\mu}_t(\boldsymbol{x}) + \sqrt{\beta_t}\boldsymbol{\sigma}_t(\boldsymbol{x})\}, \tag{1}$$

where β_t is a scaling parameter that controls the exploration-exploitation trade-off. Similar to PAL (Zuluaga et al., 2013, 2016), we use $\beta_t = 2/9 \log(n|\mathfrak{X}_\mathfrak{m}|\pi^2 t^2/6\delta)$ for $\delta \in (0, 1)$.The dimension of $R_t(\boldsymbol{x})$ depends on the number of objectives n. Later, we exploit the information about the uncertainty regions to determine the non-dominated designs set $\mathfrak{U}$. We then use the optimistic and pessimistic values of the non-dominated designs in $\mathfrak{U}$ to build the optimistic Pareto front $\mathfrak{F}_{opt}$ and pessimistic Pareto front

We now employ our cost-aware acquisition function, which makes use of an information gain based on objective space entropy. Being cost-aware, our proposed acquisition function $\alpha_{t,i}(\boldsymbol{x})$ considers the evaluation cost $\theta_{t,i}$ across each objective f_i:

$$\alpha_{t,i}(\boldsymbol{x}) = \frac{\Delta V(\{\boldsymbol{x}, f_{t,i}(\boldsymbol{x})\}, \hat{\mathfrak{F}}^* | \mathfrak{X}_\mathfrak{m}{}^*)}{\theta_{t,i}} \tag{2}$$

$$= \frac{V(P_R|\hat{\mathfrak{F}}^*) - V\left(P_R|\hat{\mathfrak{F}}^*_{R_{t,i}(\boldsymbol{x})=\boldsymbol{\mu}_{t,i}(\boldsymbol{x})}\right)}{\theta_{t,i}} \tag{3}$$

$$= \frac{\Delta V_{t,i}}{\theta_{t,i}} \tag{4}$$

Here, $\alpha_{t,i}(\boldsymbol{x})$ computes the amount of information that can be gained per cost for a design $\boldsymbol{x}$ to be evaluated for an objective f_i. In Equation 3, we compute the gain of information as the change of volume of the Pareto region if the Pareto front $\hat{\mathfrak{F}}^* = \mathfrak{F}_{opt} \cup \mathfrak{F}_{pess}$ is updated by setting the uncertainty values $R_{t,i}(\boldsymbol{x})$ of $\boldsymbol{x}$ to its mean $\mu_{t,i}(\boldsymbol{x})$ for the corresponding designs in $\mathcal{X}^*_m$. Our acquisition function computes the change of volume $\Delta V_{t,i}$ of the Pareto region P_R across each objective f_i to judiciously determine the gain of information that would be achieved if design $\boldsymbol{x}$ is evaluated for f_i. We select a design $\boldsymbol{x}_t$ and an objective $f_{t,i}$ using the

following:

$$\boldsymbol{x}_t, f_{t,i} = \text{argmax}_{\boldsymbol{x} \in \mathcal{X}_m^* \text{ for each } f_i} \ \alpha_{t,i}(\boldsymbol{x}) \tag{5}$$

Here, we identify the most promising design for an objective function that gains the most information given the cost of evaluating it. Finally, we update the surrogate model $\mathcal{M}_i$ corresponding to the chosen objective function f_i by incorporating the newly-evaluated design and objective value. We stop when the maximum budget θ_T is exhausted or the maximum change of volume of the Pareto region becomes zero (indicating that all designs in the Pareto region have been evaluated for each objective), whichever occurs earlier. Finally, we return the Pareto front obtained. Every iteration t consists of three stages: (1) modeling, (2) construction of the Pareto region, and (3) sampling. To initialize FlexiBO, we evaluate N_0 samples for each objective f_i to and populate corresponding evaluated designs set S_i for $1 \leq i \leq n$. We also determine the average computational effort θ_i for each objective f_i before proceeding to the iterative procedure. We outline the pseudocode for the FlexiBO implementation in Algorithm 1.

5.1.1 Modeling

At each iteration t, we train a surrogate model $\mathfrak{M}_i$ using the samples in the evaluated designs set S_i for objective f_i. As FlexiBO selects one objective f_i for evaluation per iteration, only the surrogate model $\mathfrak{M}_i$ corresponding to the selected objective f_i needs to be updated. Then, we determine $\mathfrak{X}_m$ from $\mathfrak{X}$ using Monte-Carlo sampling (Shapiro, 2003). At this point, we determine the uncertainty region $R_t(\boldsymbol{x})$ of each design $\boldsymbol{x} \in \mathfrak{X}_\mathfrak{m}$ using Equation 1. The 2-dimensional objective space $\mathfrak{F}$ in Figure 4 shows examples of uncertainty regions for different designs $\boldsymbol{x}$. As shown in Figure 4, the uncertainty region $R_t(\boldsymbol{x})$ of a design $\boldsymbol{x}$ that is not evaluated across any of the two objectives, f_1 and f_2, is a rectangle. If $\boldsymbol{x}$ is evaluated across one objective, say f_2, the uncertainty across f_2 will be eliminated (assuming measurements contain no noise) and $R_t(\boldsymbol{x})$ will become a line across f_1. Once, $\boldsymbol{x}$ is evaluated across both objectives, $R_t(\boldsymbol{x})$ is expressed by a point (indicating no uncertainty across f_1 and f_2).

5.1.2 Pareto Region Construction

After the uncertainty region $R_t(\boldsymbol{x})$ for each design $\boldsymbol{x} \in \mathfrak{X}_\mathfrak{m}$, we identify the set of non-dominated designs $\mathfrak{U}$ using the following rule:

$$\boldsymbol{x} \in \mathfrak{U} \text{ if } \min(R_t(\boldsymbol{x})) \succeq \max(R_t(\boldsymbol{x}')) \text{ for } \boldsymbol{x} \neq \boldsymbol{x}' \text{ and } \boldsymbol{x}, \boldsymbol{x}' \in \mathfrak{X}_\mathfrak{m} \tag{6}$$

Figure 5 shows examples of non-dominated designs (gray color) and dominated designs (black color) in the objective space for $n = 2$. Next, we identify the set of Pareto-optimal solutions ${\mathfrak{X}_\mathfrak{m}}^*$ and Pareto front $\hat{\mathfrak{F}}^* = \{\mathfrak{F}_{opt} \cup \mathfrak{F}_{pess}\}$ for the purpose of constructing the Pareto region P_R by pruning designs in $\mathfrak{U}$. A design $\boldsymbol{x} \in \mathfrak{U}$ is only included in $\mathfrak{F}_{opt}$ if the optimistic value $\max(R_t(\boldsymbol{x}))$ of $\boldsymbol{x}$ is not dominated by the optimistic value $\max(R_t(\boldsymbol{x}'))$ of another design $\boldsymbol{x}'$ across all objectives as follows.

$$\boldsymbol{x} \in \mathfrak{F}_\mathfrak{opt} \text{ if } \max(R_t(\boldsymbol{x})) \succeq \max(R_t(\boldsymbol{x}')) \text{ for } \boldsymbol{x} \neq \boldsymbol{x} \text{ and } \boldsymbol{x}, \boldsymbol{x}' \in \mathfrak{U} \tag{7}$$

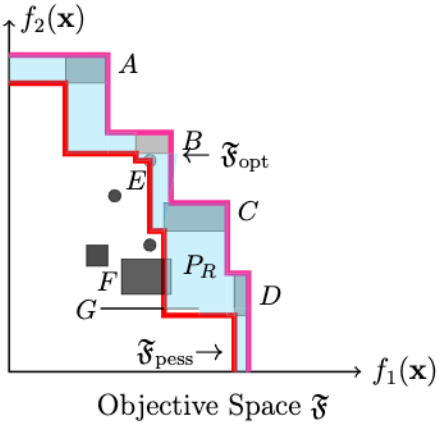


Figure 6: Example showing pruning of non-dominated points to construct $\mathfrak{F}_{opt}$ and $\mathfrak{F}_{pess}$.

Figure 6 shows an example where non-dominated designs F or G are not included in $\mathfrak{F}_{opt}$ as the optimistic values of F or G are dominated by the optimistic values of non-dominated designs B or C.

We directly add the pessimistic value $\min(R_t(\boldsymbol{x}))$ of a design $\boldsymbol{x}$ to $\mathfrak{F}_{pess}$ if it remains non-dominated by the pessimistic value $\min(R_t(\boldsymbol{x}'))$ of any other point $\boldsymbol{x}'$ as follows.

$$\boldsymbol{x} \in \mathfrak{F}_{\text{pess}} \text{ if } \min(R_t(\boldsymbol{x})) \succeq \min(R_t(\boldsymbol{x}')) \text{ for } \boldsymbol{x} \neq \boldsymbol{x}' \in \mathfrak{U} \tag{8}$$

As shown in Figure 6, pessimistic values of B, D, E etc. are added to $\mathfrak{F}_{pess}$ using the above rule in Equation 8. However, the uncertainty regions $R_t(\boldsymbol{x}')$ of some designs $\boldsymbol{x}' \in \mathfrak{U}$ ruled out of $\mathfrak{F}_{pess}$ using Equation 8 can have some degree of overlap with the uncertainty region $R_t(\boldsymbol{x})$ of a design $\boldsymbol{x} \in \mathfrak{F}_{pess}$. Consider the uncertainty regions of F and G in Figure 6. Though their pessimistic values are dominated by the pessimistic values of B and C, there is some overlap of the uncertainty regions of F and G with the uncertainty regions of B and C across an objective, in this case, f_1. Overlapping uncertainty regions of F and G with C are gray as they remain non-dominated by the pessimistic value of C in Figure 6. In such cases, the pessimistic values of F and G are updated with the minimum values of the overlapping non-dominated uncertainty region using the following rule:

$$\begin{aligned} \min(R_{t,i}(\boldsymbol{x}')) = \min(R_{t,i}(\boldsymbol{x})) \text{ if } \min(R_t(\boldsymbol{x})) \succeq \min(R_t(\boldsymbol{x}')) \text{ and} \\ \min(R_{t,i}(\boldsymbol{x})) \not\succeq \max(R_{t,i}(\boldsymbol{x}')) \text{ for each } f_i \text{ where, } \boldsymbol{x} \neq \boldsymbol{x}' \text{ and } \boldsymbol{x}, \boldsymbol{x}' \in \mathfrak{U} \end{aligned} \tag{9}$$

Later, updated pessimistic values of F or G are added to the pessimistic Pareto front $\mathfrak{F}_{pess}$ if it remains non-dominated. Note that there can be more than one design in $\mathfrak{F}_{pess}$ whose pessimistic value can dominate the pessimistic value of another design not yet included in $\mathfrak{F}_{pess}$ and has an overlap. Therefore, we need to repeat the above process for each of those designs and finally update to a value that remains non-dominated. This process of identifying $\mathfrak{F}_{pess}$ ensures that any design that has the potential to be included in the pessimistic Pareto front is not discarded from our consideration. Finally, the Pareto region P_R bounded by $\mathfrak{F}_{opt}$ and $\mathfrak{F}_{pess}$ is constructed.

Algorithm 1: The FlexiBO algorithm.

Input: design space $\mathfrak{X}$; maximum budget θ_T; number of initial designs N_0;

Initialization

S_i = Evaluate N_0 designs for each objective f_i

Determine average computational effort $\theta_i = (\sum_{t=1}^{N_0} \theta_{t,i})/N_0$ for each f_i

$t = N_0$ and $\theta = 0$

while $\theta \leq \theta_T$ **do**

  Modeling

  Train surrogate models $\mathfrak{M}_{t,i}$ using corresponding evaluated designs set S_i for f_i

  Obtain $\boldsymbol{\mu}_t(\boldsymbol{x}) = (\mu_{t,i}(\boldsymbol{x}))_{1 \leq i \leq n}$ and $\boldsymbol{\sigma}_t(\boldsymbol{x}) = (\sigma_{t,i}(\boldsymbol{x}))_{1 \leq i \leq n}$ using $(\mathfrak{M}_{t,i})_{1 \leq i \leq n}$ for all $\boldsymbol{x} \in \mathfrak{X}_{\mathfrak{m}}$.

  Compute uncertainty region $R_t(\boldsymbol{x})$ of each design $\boldsymbol{x} \in \mathfrak{X}_{\mathfrak{m}}$ using Equation 1

  Pareto region construction

  $\mathfrak{U} = \varnothing$

  for *all* $\boldsymbol{x} \in \mathfrak{X}_{\mathfrak{m}}$ **do**

    if *no* $\boldsymbol{x} \neq \boldsymbol{x}'$ *for* $\boldsymbol{x}' \in \mathfrak{X}_{\mathfrak{m}}$ *exists such that* $\min(R_t(\boldsymbol{x})) \succeq \max(R_t(\boldsymbol{x}'))$ **then**

      $\mathfrak{U} = \mathfrak{U} \cup \{\boldsymbol{x}\}$

  Identify optimistic Pareto front $\mathfrak{F}_{opt}$ and pessimistic Pareto front $\mathfrak{F}_{pess}$ using $\mathfrak{U}$ and $R_t(\boldsymbol{x})$ with Equation 7, Equation 8, and Equation 9

  Sampling

  Compute acquisition function $\alpha_{t,i}(\boldsymbol{x})$ across each objective f_i using $\boldsymbol{x} \in \mathfrak{X}_{\mathfrak{m}}{}^*$ with Equation 3

  Update $t = t + 1$

  Choose the next sample $\boldsymbol{x}_t$ and objective $f_{t,i}$ using Equation 10

  Evaluate $\boldsymbol{x}_t : y_{t,i} = f_{t,i}(\boldsymbol{x}_t)$

  Aggregate data $S_i = S_i \cup \{(\boldsymbol{x}_t, y_{t,i})\}$

  Update $\theta_{t,i} = ((t-1) * \theta_i + \theta_{t,i})/t$

  Update $\theta = \theta + \theta_{t,i}$

return Return the non-dominated designs from the evaluated designs set $(S_i)_{1 \leq i \leq n}$ as the Pareto front using Equation 6

5.1.3 Sampling

At this stage, we select the next design $\boldsymbol{x}_t$ and objective $f_{t,i}$ for evaluation using our proposed acquisition function $\alpha_{t,i}(\boldsymbol{x})$ by the following:

$$\boldsymbol{x}_t, f_{t,i} = \text{argmax}_{\boldsymbol{x} \in \mathfrak{X}_{\mathfrak{m}}{}^* \text{ for each } f_i} \ \alpha_{t,i}(\boldsymbol{x}) \tag{10}$$

Here, we only use the designs in the Pareto optimal set $\mathfrak{X}_{\mathfrak{m}}{}^*$ whose function values constitute the Pareto fronts $\mathfrak{F}_{opt}$ and $\mathfrak{F}_{pess}$ in our acquisition function calculation. We exclude designs not located on the hyperplanes passing through $\mathfrak{F}_{opt}$ and $\mathfrak{F}_{pess}$ as they do not contribute to the change of the volume of Pareto region P_R when their uncertainty across any objective is reduced to zero. Intuitively, this helps us to speed up computation.

At every iteration, the evaluated design leads to a decrease in the volume of the Pareto region, as illustrated in Figure 7. We update the set of evaluated designs S_i for $f_{t,i}$ by adding

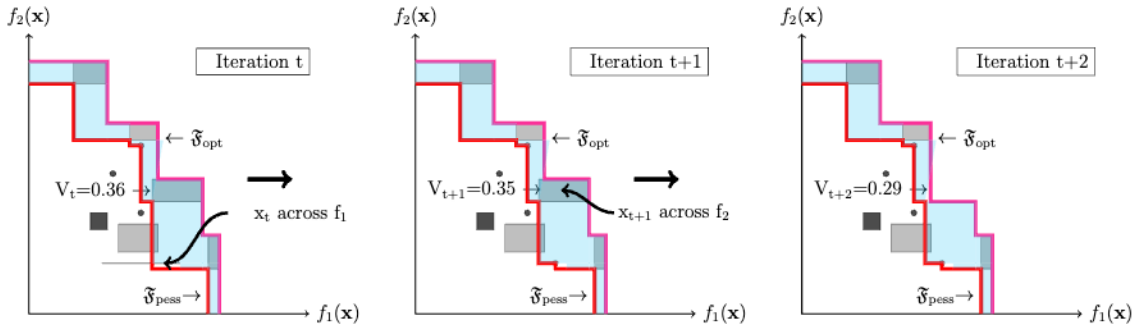


Figure 7: Decrease the volume of the Pareto region with each iteration.

$\{\boldsymbol{x}_t, f_{t,i}(\boldsymbol{x}_t)\}$. Additionally, we update the computational effort θ_i for objective $f_{t,i}$ using $\theta_i = ((t-1) * \theta_i + \theta_{t,i})/t$, where $\theta_{t,i}$ is the computational effort to evaluate $\boldsymbol{x}_t$ across $\boldsymbol{f}_{t,i}$. Once the maximum budget θ_T is exhausted, FLEXIBO returns the non-dominated designs as approximate Pareto front using the evaluated designs $\boldsymbol{x} \in S_i$ for each objective f_i. Note that the estimated mean $\mu_i(\boldsymbol{x})$ is used as the objective value for f_i if a design $\boldsymbol{x} \in S_i$ is not evaluated across f_i while determining the Pareto front.

6. Evaluation

In this section, we evaluate the following research questions (RQs):

- **RQ1**: How to select the objective evaluation function for FLEXIBO to optimize multiple objectives for DNNs?
- **RQ2**: How effective is FLEXIBO in comparison to state-of-the-art multi-objective optimization approaches for
  - ✓ different DNNs of different applications?
  - ✓ different DNNs of varying sizes (e.g., number of hyperparameters)?
- **RQ3**: How sensitive is FLEXIBO when different surrogate models are used?

6.1 Experimental Setup

We discuss the baselines, datasets, and experimental setup to evaluate FLEXIBO in this section.

6.1.1 BASELINES

We compare FLEXIBO to the following baselines:

PESMO, PESMO-DEC (Hernández-Lobato et al., 2015). These methods employ an acquisition function based on input space entropy and iteratively select the design that maximizes the information gained about the optimal Pareto set. Both of these methods are cost-aware, with PESMOemploying a coupled evaluation strategy and PESMO-DECemploying a decoupled evaluation strategy.

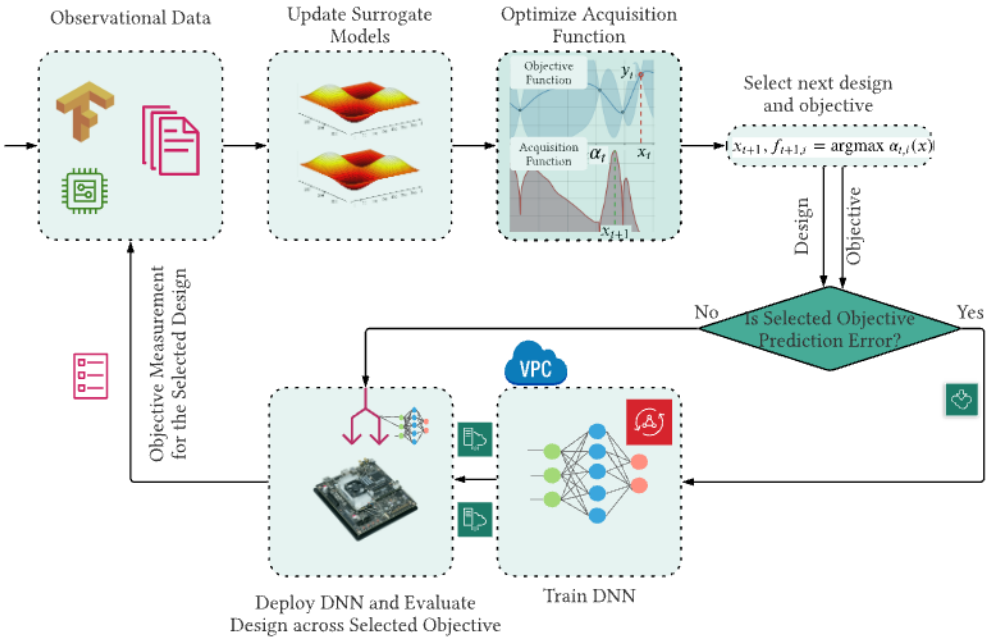


Figure 8: Experimental Setup used for FLEXIBO.

Table 2: The DNN architectures and datasets used in the experimental evaluation.

DOMAIN	ARCHITECTURE	DATASET	COMPILER	NUM. LAYERS	NUM. PARAMS	TRAIN SIZE	TEST SIZE
IMAGE	XCEPTION	IMAGENET	Keras	71	22M	100K	10K
	MOBILENET	IMAGENET	Keras	28	4.2M	100K	10K
	LENET	MNIST	Keras	7	60K	50K	10K
	RESNET	CIFAR-10	Keras	50	25M	45K	5K
	SQUEEZENET	CIFAR-10	Keras	3	1.2M	45K	5K
NLP	BERT	SQUAD 2.0	PyTorch	12	110M	56K	5K
	BERT	IMDB SENTIMENT	PyTorch	12	110M	25K	2K
SPEECH	DEEPSPEECH	COMMON VOICE	PyTorch	9	68M	300 (hrs)	2 (hrs)

PAL (Zuluaga et al., 2013). An active learning algorithm that samples the design space by classifying designs as Pareto optimal or not to identify the Pareto front. This method uses a cost unaware coupled evaluation strategy.

PAREGO (Knowles, 2006). Transforms the multi-objective problem into a single-objective problem using a scalarization technique.

SMSEGO (Ponweiser et al., 2008). This method is given by the gain in hyper-volume obtained by the corresponding optimistic estimate after an ϵ correction has been made. The hypervolume is simply the volume of points in functional space above the Pareto front (this is simply the function space values associated with the Pareto set), with respect to a given reference.

CA-MOBO (Abdolshah et al., 2019a). The acquisition function in CA-MOBO uses Chebyshev scalarization for objective functions to ensure the solutions satisfy Pareto optimality and a cost function as a component of the acquisition function that incorporates the user's prior knowledge of the search space. This multi-objective optimization method uses a cost-aware coupled evaluation scheme.

For ParEGO, SMSego, PESMO, and PESMO-DEC implementation, we employ the code from the MOBO library Spearmint[2]. For CA-MOBO, we use the code from the repository[3] provided in the paper. For PAL implementation, we develop our own version in Python by carefully following the algorithm presented in the paper (Zuluaga et al., 2013). We run each optimization pipeline 5 times using different initial evaluations, where the initial evaluations in one run are the same for all methods. The initial evaluations are sampled at random from a Sobol grid which is the same as the one in the Spearmint library.

6.1.2 Datasets

We use seven DNN architectures from three different problem domains; Image, NLP, and Speech. For each architecture, we select the most common dataset and compiler typically used in practice. Table 2 lists the architectures, datasets, compilers, and sizes of the training and test sets used in our experiments.

Image. To evaluate the performance of FlexiBO for image recognition applications, we use the Xception (Chollet, 2017), MobileNet (Sandler et al., 2018), LeNet (LeCun et al., 2015), ResNet (He et al., 2016), and SqueezeNet (Iandola et al., 2016) architectures. For both Xception and MobileNet, we use the ImageNet ILSVRC2017 challenge dataset (Russakovsky et al., 2015) and randomly select 100,000 train and 10,000 test images for our experiments. We use the MNIST dataset (LeCun & Cortes, 2010) of handwritten images for LeNet. Our training and test datasets consist of 45,000 and 5,000 images, respectively. For our evaluation of FlexiBO on ResNet and SqueezeNet, we use the CIFAR-10 dataset (Krizhevsky et al., 2009), which consists of 60,000 images of size 32×32 with 10 classes (6,000 images per class). We use 50,000 images for training and the remaining 10,000 images for testing.

NLP. We use the popular BERT (Devlin et al., 2018) architecture for our evaluation of FlexiBO for NLP applications. We combine BERT on 2 benchmark datasets: a question-answering dataset, SQuAD 2.0 (Rajpurkar et al., 2016), and the IMDB Movie Review Sentiment Analysis dataset. Out of 130,319 training and 8,863 testing examples of the original SQuAD 2.0 dataset, we randomly select 56,000 training and 5,000 testing examples for our experiments with BERT (termed BERT-SQuAD). For the IMDB movie review dataset (termed BERT-IMDB), we use all 25,000 binary sentiment analysis training examples for training and randomly select 2,000 examples for testing out of the 25,000 testing examples provided in the IMDB dataset.

Speech. To evaluate the performance of FlexiBO for speech recognition, we use DeepSpeech (Hannun et al., 2014) with the Common Voice dataset (Mozilla, 2019). We randomly extract 300 hours of voice data for 5 different languages (English, Arabic, Chinese, German,

2. https://github.com/HIPS/Spearmint/tree/PESM
3. https://github.com/MajidAbdolshah/CA-MOBO

and Spanish) from nearly 3,700 hours of voice data of the Common Voice dataset for training. To evaluate the prediction error we test on 2 hours of voice data.

Table 3: DNN-specific design options and their values.

Architecture	Design Option	Value/Range
Xception	Number of Filters Entry Flow	16, 32, 64, 128, 256
	Number of Filters Middle Flow	16, 32, 64, 128, 256
	Filter Size Entry Flow	(1×1), (3×3), (5×5), (7×7), (9×9)
	Filter Size Middle Flow	(1×1), (3×3), (5×5), (7×7), (9×9)
	Filter Size Exit Flow	(1×1), (3×3)
MobileNet	Number of Filters Stem	16, 32, 64, 128, 256
	Filter Size Stem	(1×1), (3×3), (5×5), (7×7), (9×9)
	Number of Filters Depthwise Block One	16, 32, 64, 128, 256, 512, 1024
	Number of Filters Depthwise Block Two	16, 32, 64, 128, 256, 512, 1024
	Number of Filters Depthwise Block Three	16, 32, 64, 128, 256, 512, 1024
	Number of Filters Depthwise Block Four	16, 32, 64, 128, 256, 512, 1024
LeNet	Number of Filters Layer 1	16, 32, 64, 128, 256, 512, 1024
	Filter Size Layer 1	(1×1), (3×3), (5×5), (7×7), (9×9)
	Number of Filters Layer 2	16, 32, 64, 128, 256, 512, 1024
	Filter Size Layer 2	(1×1), (3×3), (5×5), (7×7), (9×9)
	Number of Filters Layer 3	16, 32, 64, 128, 256, 512, 1024
	Filter Size Layer 3	(1×1), (3×3), (5×5), (7×7), (9×9)
	Number of Filters Layer 4	16, 32, 64, 128, 256, 512, 1024
	Filter Size Layer 4	(1×1), (3×3), (5×5), (7×7), (9×9)
ResNet	Number of Filters Stem	16, 32, 64, 128, 256, 512, 1024
	Number of Filters Projection Block	16, 32, 64, 128, 256, 512, 1024
	Filter Size Projection Block	(1×1), (3×3), (5×5), (7×7), (9×9)
	Number of Filters Bottleneck Block	16, 32, 64, 128, 256, 512, 1024
	Filter Size Bottleneck Block	(1×1), (3×3), (5×5), (7×7), (9×9)
SqueezeNet	Number of Filters Stem	16, 32, 64, 128, 256, 512, 1024
	Number of Filters Stem	16, 32, 64, 128, 256, 512, 1024
	Filter Size Fire Group One	(1×1), (3×3), (5×5), (7×7), (9×9)
	Number of Filters Fire Group Two	16, 32, 64, 128, 256, 512, 1024
	Number of Filters Fire Block	16, 32, 64, 128, 256, 512, 1024
Bert	Dropout	0.1, 0.3, 0.5, 0.7, 0.9
	Maximum Batch Size	6, 12, 16, 32, 64
	Maximum Sequence Length	13, 16, 32, 64, 128, 256
	Learning Rate	$1e^{-5}$, $2e^{-5}$, $3e^{-5}$, $4e^{-5}$, $5e^{-5}$
	Weight Decay	0, 0.1, 0.2, 0.3
Deepspeech	Num of epochs	2, 4, 8, 16, 32
	Dropout	0.1, 0.3, 0.5, 0.7, 0.9
	Maximum Batch Size	16, 32, 64, 128, 256
	Maximum Sequence Length	16, 32, 64, 128, 256, 512, 1024
	Learning Rate	$1e^{-5}$, $2e^{-5}$, $3e^{-5}$, $4e^{-5}$, $5e^{-5}$

Table 4: OS-specific design options and their values.

Design Option	Value/Range
Scheduler Policy	`CFP`, `NOOP`
Swappiness	10, 30, 60, 100
Dirty Background Ratio	10, 50, 80
Dirty Ratio	5, 50
Cache Pressure	100, 500

Table 5: Hardware-specific design options and their values.

Design Option	Value/Range
	Jetson Xavier
Num Active CPU	1 - 6
CPU Frequency (GHz)	0.3 - 2.3
GPU Frequency (GHz)	0.3 - 1.8
EMC Frequency (GHz)	0.3 - 2.0

6.1.3 Objectives and Design Options

We select two objectives: energy consumption and prediction error for optimization for each architecture in our experiments. While we restrict ourselves to two objectives, our methodology can be applied to an arbitrary number of objectives. Depending on the particular hardware platform and DNN architecture, we select 14-17 design options. Each platform and DNN has its own specific hardware and DNN design options; OS-specific options are the same. We consider 4 hardware-specific design options, 5 OS-specific options, and 5-8 DNN-specific options. Our chosen DNN-specific, OS-specific, and hardware-specific design options are listed in Tables 3, 4, and 5, respectively. We choose these options based on similar hardware configuration guides/tutorials and (Halawa et al., 2017). The choice of these design options presents an interesting scenario for optimization based on how they influence performance objectives because of the complex interactions of the options. Hardware- and OS-specific options like the number of active CPUs or the scheduler policy affect only energy consumption, whereas DNN options like filter size or the number of filters affect both energy consumption and prediction error. Depending on the DNN architecture, we use either Keras (Tensorflow as the backend) or PyTorch as the compiler for training and prediction (see Table 2 for details).

6.1.4 Setting

To initialize FlexiBO, we measure the prediction error and energy consumption of 20 randomly selected designs from the design space of a particular DNN system. As energy consumption measurements tend to be noisy, we take 10 repeated measurements for a particular design $\boldsymbol{x}$ and consider the median. We do not repeat prediction error measurements as they are not noisy. We use two different surrogate models, Gaussian process (GP) and

Table 6: The list of hyperparameters used for the surrogate models.

SURROGATE	HYPERPARAMETERS	VALUE
GP	Kernel	Squared Exponential
	Num Restarts	20
	Optimizer	L-BFGS-B
	α	$1e^{-10}$
RF	Num Trees	128
	Min Split Variable	2
	Min Impurity Split	$1e^{-7}$

Random Forest (RF), termed FLEXIBO-GP and FLEXIBO-RF, respectively. Details of the hyperparameters used for both GP and RF are provided in Table 6. We use the Wall-Clock Time $t_{wc,i}$ required to evaluate an objective f_i as the computational effort and run experiments with three different objective evaluation cost functions: (i) Logarithmic cost (LC): $\theta_i = \log(t_{wc,i})$, (ii) Ratio cost (RC): $\theta_i = \frac{t_{wc,i}}{\min(t_{wc,i})_{1 \leq i \leq n}}$ and, (iii) Constant cost (CC): $\theta_i = 1$ to simulate a method that is not cost-aware. At each iteration, FLEXIBO recommends a design and an objective for evaluation. Depending on the objective selected for evaluation, we take the following actions.

- The recommended objective is *prediction error*.
  - ✓ We retrain a DNN with the DNN-specific options of the selected design if no pre-trained model for the DNN-specific options exists. We reuse the pre-trained model otherwise. We measure prediction error by running inference using the test dataset for each DNN architecture.
- The recommended objective is *energy consumption*.
  - ✓ If no pre-trained model with the DNN-specific options of the selected design exist, we use a model whose size is the same as that of the model obtained after random initialization of the weights using the DNN-specific design options, else we reuse the pre-trained model. We measure energy consumption on Jetson Xavier for the selected design during the inference of the test dataset for each DNN architecture. The built-in sensor on the Jetson Xavier records voltage and current readings in the SYSFS nodes. We read the power data every 50 milliseconds and combine them to determine energy consumption.

Figure 8 gives a high-level overview of our experimental setup. We implement FLEXIBO in a distributed manner where the training of a DNN is done remotely on virtual machine instances with 8 NVIDIA Tesla K80 GPU deployed on the Google cloud, and the measurements and optimization algorithms run locally on resource-constrained Jetson devices, such as, Xavier. Our experiments took a total of 5552.4 hours of wall-clock time to complete. We ensure the DVFS setting is off and that no other jobs are running in the background while the experiments are running for accurate measurements.

Hypervolume Error. The Pareto hypervolume ($\mathfrak{hv}$) is commonly used to measure the quality of an estimated Pareto front $\hat{\mathfrak{F}}^*$(Cao et al., 2015; Zitzler & Thiele, 1999). As seen in Equation 11, it is defined as the volume enclosed by the estimated Pareto front $\hat{\mathfrak{F}}^*$ and a user-defined reference point r in the objective space, in our case the origin of the coordinate system.

$$\mathfrak{hv}(\hat{\mathfrak{F}}^*, r) = V(\cup_{s \in \hat{\mathfrak{F}}^*}\{q | r \preceq q \preceq s\}) \tag{11}$$

The hypervolume error η is defined as the difference between the hypervolumes of the *true Pareto front* $\mathfrak{F}^*$ and the estimated Pareto front $\hat{\mathfrak{F}}^*$.

$$\eta = \mathfrak{hv}(\mathfrak{F}^*, r) - \mathfrak{hv}(\hat{\mathfrak{F}}^*, r) \tag{12}$$

We evaluate the quality of the obtained Pareto fronts using the hypervolume error and the cumulative log wall-clock time as the objective evaluation cost required to obtain it. In a real-world setting, the true Pareto fronts $\mathfrak{F}^*$ are unknown. Therefore, we use an approximation of the true Pareto front by combining the Pareto fronts obtained by the different optimization methods considered in our experiments for evaluation. We use the algorithm presented in (Fonseca et al., 2006) for hypervolume computation.

6.2 Experimental Results

Given the same wall clock time, we observe the hypervolume error obtained by Pareto fronts identified by the different optimization methods. Furthermore, to assess the quality of the Pareto fronts, we compare the number of designs in the target region of the objective space. Our target region is where the prediction error is less than 25%, and the energy consumption is less than the first quartile. Note that energy consumption is specific to the hardware platform.

6.2.1 RQ1: Determination of Objective Evaluation Cost Function

Figures 9 and 10 show the results for optimizing prediction error and energy consumption with different cost functions. FlexiBO-GPLC has lower hypervolume error (shown in Figure 9) and a higher number of designs in the target region (shown in Figure 10) than FlexiBO-GPRC and FlexiBO-GPCC. To better understand the effect of different cost functions, we also look at the behavior of FlexiBO in Figure 11(a) and 11(b). Cost-unaware FlexiBO-GPCC greedily selects the design and objective across which the volume change is maximal for evaluation. As a result, FlexiBO-GPCC wastes resources by selecting expensive evaluations for little gain. This is evident from 11(b), which indicates that the reduction of volume by the designs selected by FlexiBO-GPCC is lower considering the evaluation cost. A larger change in the volume of the Pareto region is desired as it indicates higher information gain. The objective evaluation cost function in FlexiBO-GPRC is skewed towards selecting the objective with lower evaluation cost and evaluates a higher number of designs for the less expensive objective, such as, energy consumption (shown in Figure 11(a)). However, it achieves a lower change in the volume of the Pareto region than FlexiBO-GPLC. FlexiBO-GPLC on the other hand, selects designs that achieved a larger volume change across the Pareto region (Figure 11(b)) and, therefore, a better choice than others.

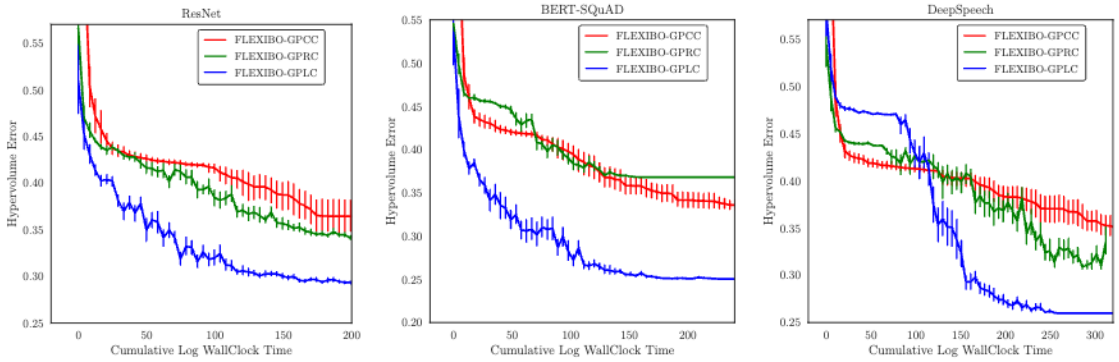


Figure 9: Comparison of hypervolume error obtained by FLEXIBO using different cost functions.

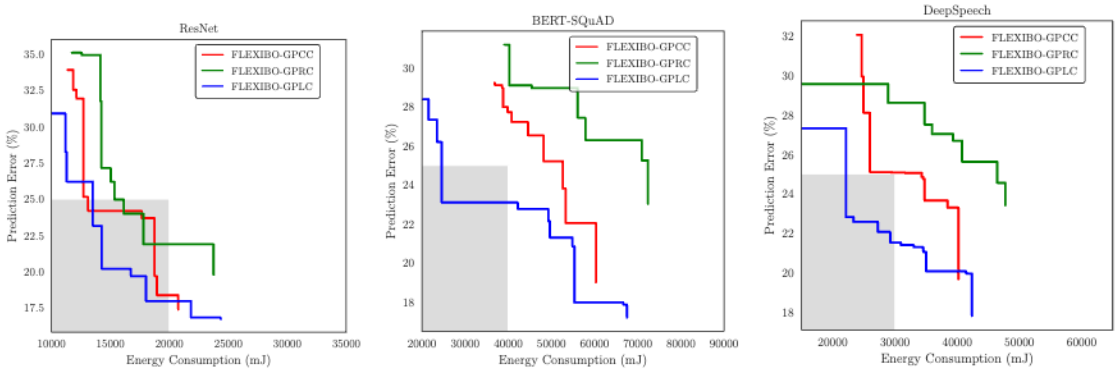


Figure 10: Comparison of Pareto optimal designs obtained by FLEXIBO using different cost functions.

6.2.2 RQ2: EFFECTIVENESS OF FLEXIBO

Effectiveness across DNNs of different applications. Figures 12 and 13 show the effectiveness analysis of FLEXIBO across different applications. In Figure 12 we observe that FLEXIBO-GPLC outperforms other methods in finding Pareto fronts with lower hypervolume error for each of the applications. For example, FLEXIBO achieves 22.4% lower hypervolume error than CA-MOBO in DEEPSPEECH. In Figure 13, we observe that FLEXIBO-GPLC is able to find a higher number of designs in the target region than other methods for RESNET, BERT-SQUAD, and DEEPSPEECH.

Effectiveness across DNNs of different sizes. Figures 14 and 15 show the effectiveness analysis of FLEXIBO across different-size DNNs. We make the following observations: (a) as shown in Figure 14, we find that FLEXIBO-GPLC outperforms other methods in finding Pareto fronts with lower hypervolume error across all applications (e.g., 22.3% lower for IMDB than CA-MOBO), (b) in Figure 15, we observe that FLEXIBO-GPLC is able to find a higher number of designs in the target region than other methods for Xception, MobileNet, and BERT-IMDB. For LeNet, PAL achieves a 3.6% lower hypervolume error

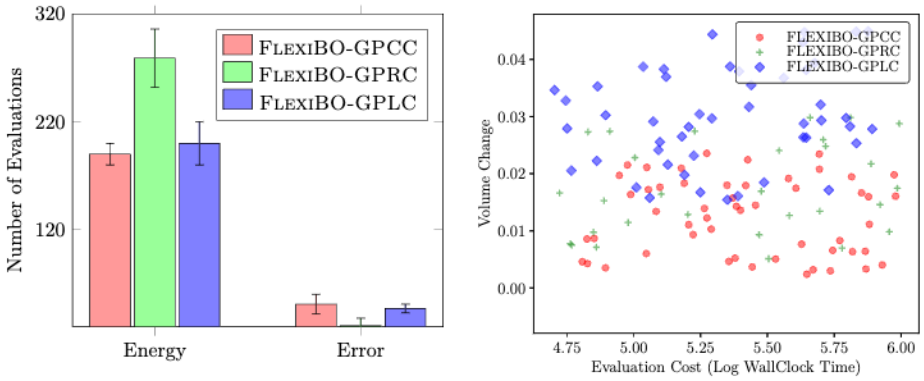


Figure 11: (a) Number of evaluations across each objective by FLEXIBO using different cost functions (b) FLEXIBO-GPLC achieved a higher change of volume of the Pareto region with the recommended design and objective when compared to others.

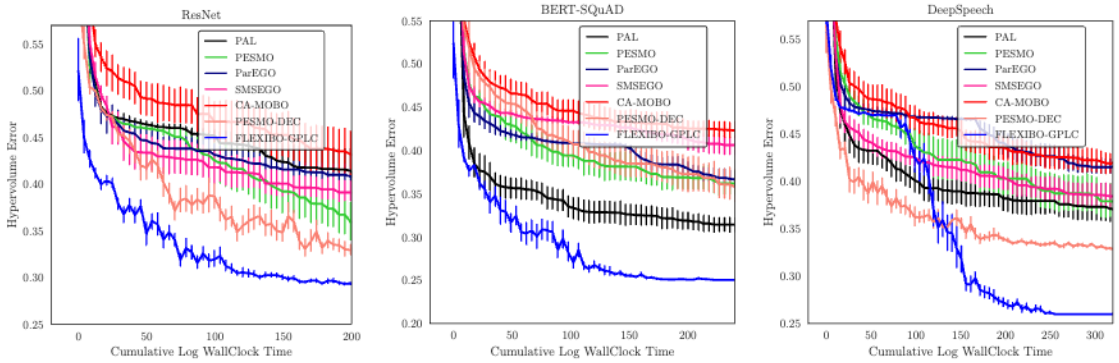


Figure 12: Comparison of hypervolume error obtained by FLEXIBO and other approaches for DNNs for object detection, NLP, and speech recognition applications.

than FLEXIBO-GPLC. LeNet is a small architecture, and FLEXIBO performs poorly for such small architectures as the effect of selecting designs based on the change of volume of the Pareto region per cost is less pronounced than for larger architectures (Xception or BERT-IMDB, etc.).

Table 7: Time (in seconds) required for one iteration with and without objective evaluation across all architectures.

	FLEXIBO	PESMO	PESMO-DEC	PAL	CA-MOBO	PAREGO	SMSEGO
NO EVALUATION	79.9±7.4	64.8±5.1	66.9± 5.4	178.2± 10.6	71.2± 8.4	56.2	48.4±4.3
WITH EVALUATION	1417.2± 97.6	9133.7±448.8	4400.4±225.9	8764.3±469.5	8411.5±365.4	8656.9±220.3	9020.9±306.3

We observe that approaches other than FLEXIBO cannot make the best use of the allocated budget as they evaluate the more expensive objectives more than the cheap objectives. As the expensive objectives can be selected any time (even for little gain), this strategy is wasteful when limited resources are available. FLEXIBO makes better use of the resources

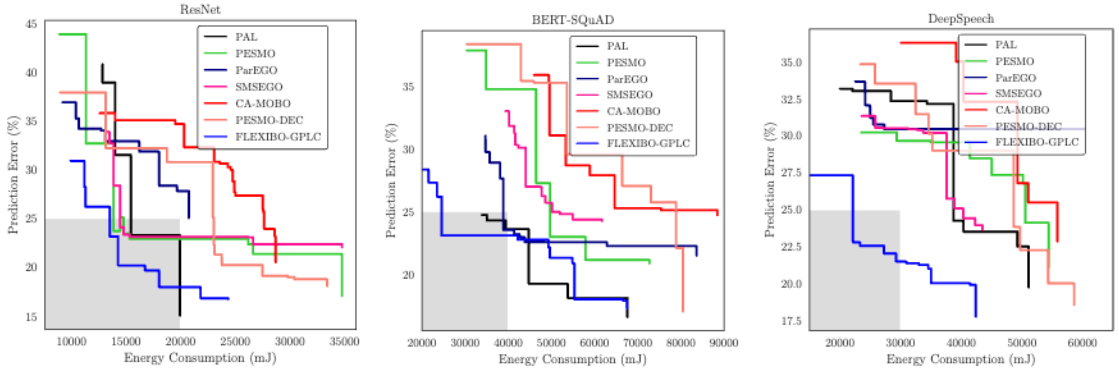


Figure 13: Comparison of Pareto fronts obtained by FLEXIBO and other approaches for DNNs for object detection, NLP, and speech recognition applications.

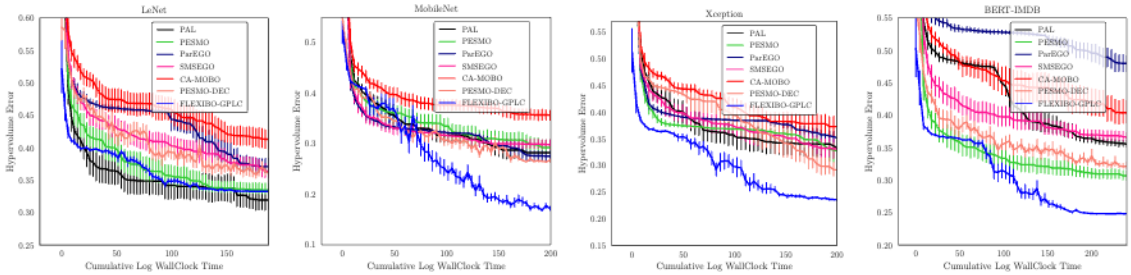


Figure 14: Comparison of hypervolume error obtained by FLEXIBO and other approaches for DNNs of different sizes.

by evaluating the cheaper objectives more in the earlier iterations and thus gaining a better understanding of the design space and only later evaluating the costly objective (Figure 16). We also find that FLEXIBO is able to evaluate more designs by prudently selecting the objectives across which to evaluate it (Figures 14 and 15). A cost-aware decoupled approach is clearly useful for scenarios where the evaluation budget is limited.

Comparison of average time required for modeling. Table 7 shows the average time required for one iteration for different multi-objective optimization methods, averaged across all architectures. Though FLEXIBO requires more time to compute the acquisition function (no evaluation) than others, the time required for one iteration including the objective evaluation time in FLEXIBO is 5.6× lower than the next best method PAREGO.

6.2.3 RQ3: SENSITIVITY ANALYSIS

Different surrogate models. We compare the performance of the different variants of FLEXIBO: FLEXIBO-GPLC and FLEXIBO-RFLC that use log objective evaluation cost. We used this cost function as it is a better choice than ratio and constant cost functions. Figures 17 and 18 show the hypervolume error and quality of the obtained Pareto fronts. We find that across all architectures, FLEXIBO-GPLC outperforms FLEXIBO-RFLC (FLEXIBO-GPLC has lower hypervolume error and a higher number of designs in the target region).

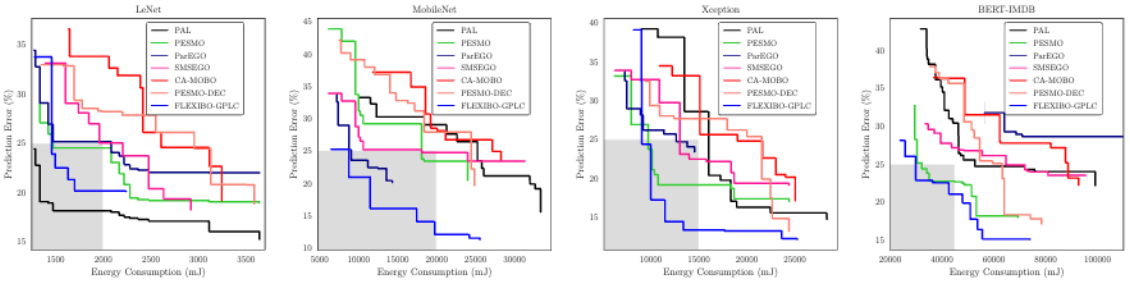


Figure 15: Comparison of Pareto fronts obtained by FlexiBO and other approaches for DNNs of different sizes.

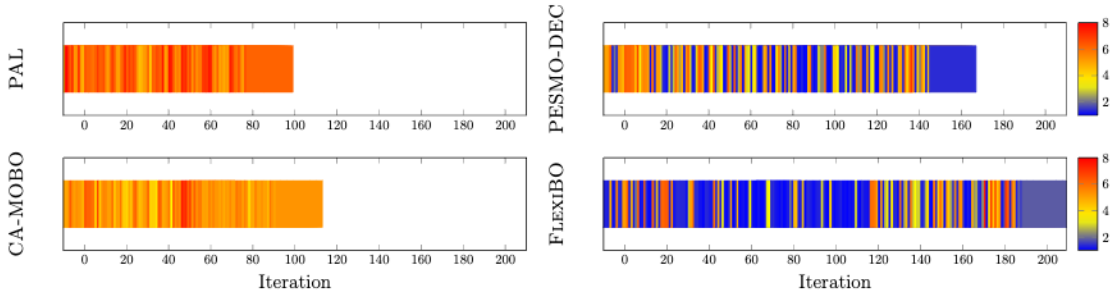


Figure 16: FlexiBO utilizes resources more efficiently than other approaches when the difference of evaluation cost between objectives is high. The colors indicate the evaluation cost. FlexiBO is able to run for more iterations as it expends less of the evaluation budget at each one.

To understand the effect of the surrogate models used on the overall optimization performance, we compare FlexiBO-GPLC and FlexiBO-RFLC with PESMO-DEC (uses GP surrogate model) and PESMO-DEC-RF (uses RF surrogate model) in ResNet. We choose PESMO-DEC for this comparison as it achieves the next best performance after FlexiBO (see Figure 12). The results presented in Figure 19 (left) show that FlexiBO-RFLC has 12.6% lower hypervolume error than PESMO-DEC-RF. However, PESMO-DEC outperforms FlexiBO-RFLC slightly by 0.5%, which indicates that the choice of surrogate model is crucial for MOBO, like, FlexiBO. The reason behind the performance discrepancy of the MOBO algorithms with different surrogate models can be explained by the results presented in Table 8 and Figure 19 (middle and right). Table 8 shows that predictions from

Table 8: Width of the confidence band for the predicted values of the surrogate models for both objectives: energy consumption and prediction error.

Surrogate Model	Confidence Interval for Energy Consumption (mJ)	Confidence Interval for Prediction Error (%)
GP	491.2	1.8
RF	768.4	2.7

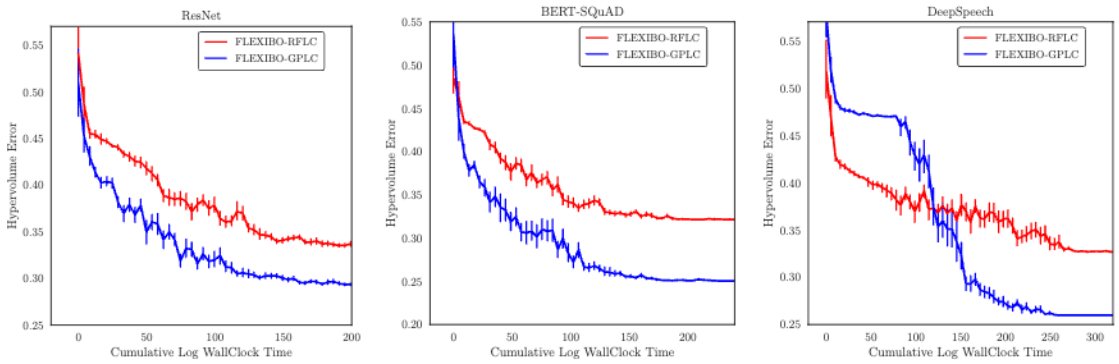


Figure 17: Comparison of hypervolume error obtained by FLEXIBO with different surrogate models.

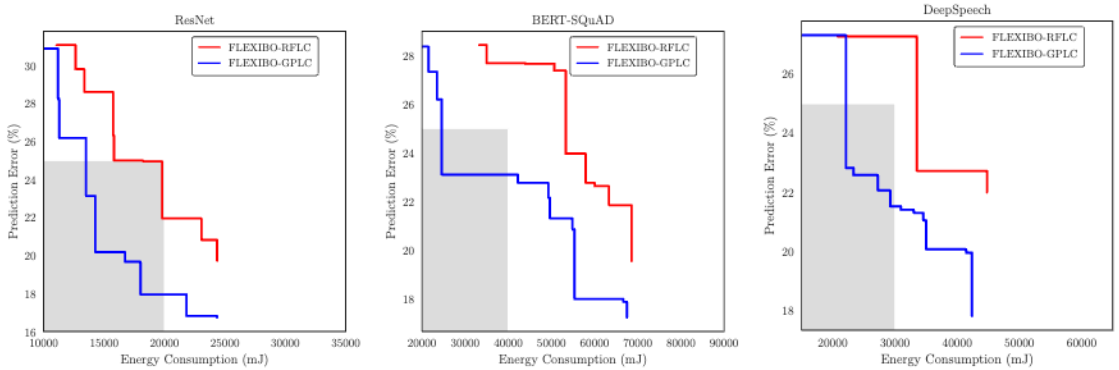


Figure 18: Comparison of Pareto fronts obtained by FLEXIBO with different surrogate models.

RF surrogate models have a wider confidence band (higher uncertainty) than GP predictions for a certain MOBO approach. From Figure 19 (middle and right), we note that as the number of designs used to train a surrogate model increases, the prediction accuracy for RF does not improve much compared to GP, particularly for the prediction error of the DNN objective. This explains why GP surrogate models based MOBO techniques perform better than RF surrogate models based MOBO techniques. Given the limited resources available for the experiment, models that achieve higher accuracy using fewer training samples are better candidates for optimization.

7. Conclusion

In this work, we proposed a novel cost-aware acquisition function for Bayesian multi-objective optimization called FLEXIBO. Instead of evaluating all objective functions, FLEXIBO automatically chooses the one that provides the highest benefit, weighted by the evaluation cost. We showed the promise of our approach through an extensive and thorough evaluation of seven different DNN architectures over a large design space on resource-constrained hardware platforms. Our experimental results show that FLEXIBO performs better than

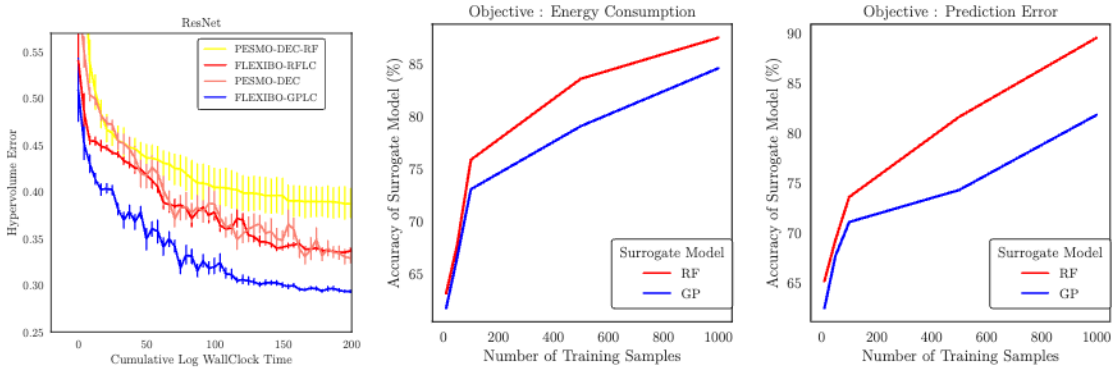


Figure 19: (left) Comparison of hypervolume for the Pareto fronts obtained by FLEXIBO and PESMO-DEC when different surrogate models are used in ResNet. (middle) Comparison of surrogate models' prediction accuracy for energy consumption. (right) Comparison of surrogate models' prediction accuracy for prediction error of the DNN.

current state-of-the-art approaches in most cases, both in terms of the quality of the obtained Pareto fronts and the cost necessary to obtain them.

Acknowledgements

This work has been supported, in part, by National Science Foundation (Awards 1813537, 2007202, 2107463, and 2233873), as well as Google and Chameleon Cloud (provided cloud resources for the experiments). We are grateful to all who provided feedback on the earlier versions of this work, including Luigi Nardi (and several members of his group at Lund), Mohammad Ali Javidian, Md Abir Hossen, several members of ABLE Research Group at CMU, and anonymous reviewers of AutoML'22.

Appendix A. Theoretical Analysis

In this section, we analyze the sample complexity of FLEXIBO. Let us assume that the maximum iteration within budget θ_T is T. By extending the theory from PAL, we derive the convergence rate of our proposed FLEXIBO algorithm. (Zuluaga et al., 2013) demonstrated that the critical quantity governing the convergence rate is given by the following:

$$\gamma_T = \max_{\boldsymbol{y}_1 \ldots \boldsymbol{y}_T} I(\boldsymbol{y}_1 \ldots \boldsymbol{y}_T; \boldsymbol{f}), \tag{13}$$

which indicates the maximum reduction of uncertainty achievable by sampling T designs. For FLEXIBO, the maximum reduction of uncertainty corresponds to the maximum change of the volume of the Pareto region ΔV and the above equation can be written as:

$$\gamma_T = \max_{\boldsymbol{y}_1 \ldots \boldsymbol{y}_T} \Delta V(\boldsymbol{y}_1 \ldots \boldsymbol{y}_T; \boldsymbol{f}), \tag{14}$$

Similar to (Srinivas et al., 2012; Zuluaga et al., 2013), we also establish γ_T as the key quantity in bounding the hypervolume error η in our analysis. The following theorem is our main theoretical result.

Table 9: List of symbols and their descriptions.

Symbol	Description
t	Number of iteration
P_R	Pareto region
T	Total number of iterations
$\mathfrak{F}_{pess}$	Pessimistic Pareto front
n	Number of objectives
V_t	Volume of P_R at iteration t
μ	Posterior mean
σ	Posterior standard deviation
$\mathfrak{U}$	Non-dominated points set
$\theta_{t,i}$	Evaluation cost of an objective f_i
$\boldsymbol{x}$	A design
r	A reference point
$\mathfrak{X}$	Design space
ΔV_t	Change of volume of the P_R at iteration t
f	An objective
$R_t(\boldsymbol{x})$	Uncertainty region of a point $\boldsymbol{x}$ at iteration t
β_t	Scaling parameter value at iteration t
$\mathfrak{M}$	Surrogate model
S_i	Evaluated points set for objective f_i
$\mathfrak{F}^*$	Optimal Pareto front
$\mathfrak{X}^*$	Pareto-optimal set
N_0	Number of initial samples
η	Pareto hypervolume error
$\mathfrak{hv}$	Pareto hypervolume
θ_T	Total objective evaluation cost
δ	Probability
γ_T	Maximum information gain
α	Acquisition function
Δ^n	n-simplex
$V(\Delta^n)$	Volume of an n-simplex
k	Co-variance
y_i	Actual value of an objective f_i
v	Measurement noise
$\hat{\mathfrak{F}}^*$	Approximate optimal Pareto front

Theorem 1. *Let $\delta \in (0,1)$. FlexiBO running with $\beta_t = 2/9\log(n|\mathfrak{X}|\pi^2 t^2/6\delta)$ would achieve a maximum hypervolume error of η of the Pareto front obtained inside total cost θ_T with probability $1-\delta$.*

$$\eta \leq \frac{\sqrt{n}a^{n-1}}{(n-1)!}\left\{a^n - 2\Big(\frac{C_1\beta_T\gamma_T}{2}\Big)^{n/2}\frac{(\frac{\theta_T}{\theta_x}(n+1))^{1/2}}{n!}\right\} \tag{15}$$

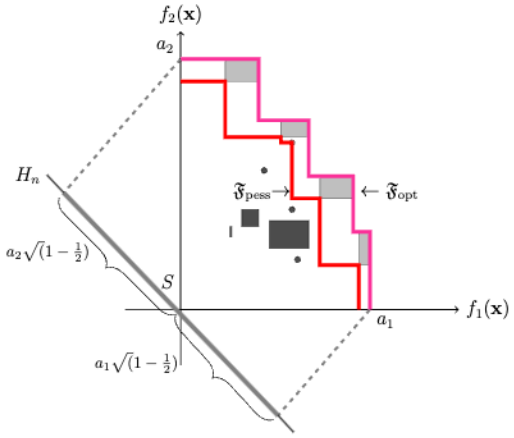


Figure 20: Example of hypervolume error bound for two objectives.

Here, $a_i = \max_{\boldsymbol{x}\in\mathfrak{X}_{\text{m}},1\leq i\leq n} \sigma_i(\boldsymbol{x})\sqrt{\beta_1}$, $C_1 = \frac{8}{\log(1+\sigma^{-2})}$, $\theta_x = \frac{\sum_{i=1}^{n}\theta_i}{n}$ *and* γ_T *It depends on the type of surrogate because the predicted uncertainty can differ depending on a model's ability to handle noisy measurements.*

This indicates that by specifying δ and a total budget θ_T, FLEXIBO can be configured to achieve a hypervolume error η with confidence $1-\delta$.

Proof. Initially, using Lemma 1 and 2, we show how the change of volume of the Pareto region $\sum_{t=1}^{T}\Delta V_{t,i}$ is related to the total budget θ_T:

$$\sum_{t=1}^{T}\Delta V_{t,i} \leq 2\Big(\frac{C_1\beta_T\gamma_T}{2}\Big)^{n/2}\frac{(\frac{\theta_T}{\theta_x}(n+1))^{1/2}}{n!}$$

Now, we relate hypervolume error η and θ_T. Let $1_n = (1,...,1)^T$ and let e_i denote the i^{th} canonical base vector. (Zuluaga et al., 2013) that considers a_i to be the maximum value for each $f_i(x)$, with probability $1-\delta$. Here, $a_i = \max_{\boldsymbol{x}\in\mathfrak{X}_{\text{m}},1\leq i\leq n} \sigma_i(\boldsymbol{x})\sqrt{\beta_1}$ obtained from the width of the confidence regions, as shown in Figure 20. We obtain this by replacing the co-variance term $k_i(\boldsymbol{x},\boldsymbol{x})$ used for measuring the width of the confidence region only for GP in Lemma 12 by (Zuluaga et al., 2013) with variance $\sigma_i^2(\boldsymbol{x})$ to extend our proof for both GP and RF surrogate models. PAL (Zuluaga et al., 2013) also showed that the projection a_i, where $1\leq i\leq n$, onto the hyperplane H_n is an n-simplex S_n has a volume of $\frac{\sqrt{n}a^{n-1}}{(n-1)!}$. Hypervolume error η depends on the distance between the boundaries defined by $\mathfrak{F}_{pess}$ and $\mathfrak{F}_{opt}$ at any iteration that is bounded by $\frac{\sqrt{n}a^{n-1}}{(n-1)!}V_t$ (Zuluaga et al., 2013). At any iteration t, V_t can be written as the difference between the initial volume of the Pareto region V_1 and the sum of change of volume ΔV_t. At iteration T, hypervolume error η can be written as the following:

$$\eta \leq \frac{\sqrt{n}a^{n-1}}{(n-1)!}\left\{V_1 - \sum_{t=1}^{T}\Delta V_{t,i}\right\}$$
$$\leq \frac{\sqrt{n}a^{n-1}}{(n-1)!}\left\{a^n - 2\Big(\frac{C_1\beta_T\gamma_T}{2}\Big)^{n/2}\frac{(\frac{\theta_T}{\theta_x}(n+1))^{1/2}}{n!}\right\} \text{ where } V_1 = a^n$$

□

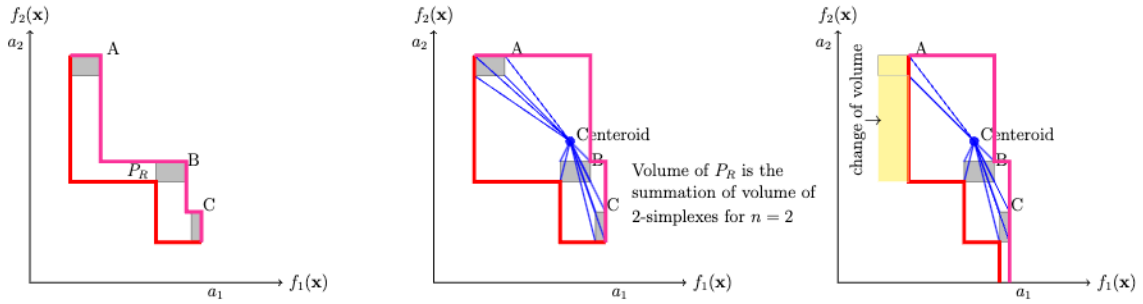


Figure 21: Example Pareto region P_R for $n = 2$ objectives (left). The Pareto region is a sum of n-simplexes (middle). The change of volume after evaluation across objective f_1 is shown by the yellow region (right).

Lemma 1. *Given* $\delta \in (0,1)$ *and* $\beta_t = 2/9\log(n|\mathfrak{X}_\mathfrak{m}|\pi^2t^2/6\delta)$, *the following holds with probability* $\leq 1-\delta$ *with* $C_1 = \frac{8}{\log(1+\sigma^{-2})}$.

$$\sum_{t=1}^{T}\Delta V_{t,i}{}^2 \leq \frac{4(C_1\beta_T\gamma_T)^n}{(n!)^2}\frac{n+1}{2^n}, \tag{16}$$

Proof. The change of volume of the Pareto region at any iteration t would be across only one objective f_i, where $1 \leq i \leq n$. Therefore, we need to determine the change of volume $\Delta V_{t,i}$ for f_i only. Let us assume that m is the centroid of the Pareto region P_R. If we add each vertex of the uncertainty region $R_t(\boldsymbol{x})$ of each design $\boldsymbol{x}$ with centroid m an n-simplex is formed. So, the Pareto region P_R can be shown as the sum of all these n-simplexes similar to Figure 21 (middle). When a design $\boldsymbol{x}_t$ is evaluated across an objective, the volume of the Pareto region is reduced by the volume of two n-simplexes as shown by the yellow region in Figure 21 (right). Volume of an n-simplex is given by $\frac{s^n}{n!}\sqrt{\frac{n+1}{2^n}}$, where s is the length of the side.

$$\Delta V_{t,i} \leq 2V(\Delta^n)$$
$$\leq \frac{2s^n}{n!}\sqrt{\frac{n+1}{2^n}}$$

By using the width of the uncertainty region $2\beta_t^{1/2}\sigma_{t-1,i}(\boldsymbol{x}_t)$ across an objective f_i as the side length s we get the following:

$$\Delta V_{t,i} \leq \frac{2(2\beta_t^{1/2}\sigma_{t-1,i}(\boldsymbol{x}_t))^n}{n!}\sqrt{\frac{n+1}{2^n}}$$

where $1 \leq i \leq n$. As β_t is increasing the above equation can be written similar to (Zuluaga et al., 2013) by the following:

$$\begin{aligned}\Delta V_{t,i}^2 &\leq \frac{4(4\beta_T\sigma^2(\sigma^{-2}\sigma_{t-1,i}^2(\boldsymbol{x}_t)))^n}{(n!)^2}\frac{n+1}{2^n} \\ &\leq \frac{4(4\beta_T\sigma^2 C_2\log(1+\sigma^{-2}\sigma_{t-1,i}^2(\boldsymbol{x}_t)))^n}{(n!)^2}\frac{n+1}{2^n},\end{aligned}$$

where $C_2 = \frac{\sigma^{-2}}{\log(1+\sigma^{-2})}$. Applying summation on the above we get

$$\sum_{t=1}^{T}\Delta V_{t,i}^2 \leq \frac{4(4\beta_T\sigma^2 C_2\sum_{t=1}^{T}\log(1+\sigma^{-2}\sigma_{t-1,i}^2(\boldsymbol{x}_t)))^n}{(n!)^2}\frac{n+1}{2^n}$$

With $C_1 = 8\sigma^2 C_2$ we get the following:

$$\begin{aligned}\sum_{t=1}^{T}\Delta V_{t,i}^2 &\leq \frac{4(4\beta_T\sigma^2 C_2\Delta V(\boldsymbol{y}_T;\boldsymbol{f}_T,i))^n}{(n!)^2}\frac{n+1}{2^n} \\ &\leq \frac{4(C_1\beta_T\Delta V(\boldsymbol{y}_T;\boldsymbol{f}_T))^n}{(n!)^2}\frac{n+1}{2^n} \\ &\leq \frac{4(C_1\beta_T\gamma_T)^n}{(n!)^2}\frac{n+1}{2^n}\end{aligned}$$

□

Lemma 2. *Given $\delta \in (0,1)$ and $\beta_t = 2/9\log(n|\mathfrak{X}_{\mathfrak{m}}|\pi^2 t^2/6\delta)$, the following holds with probability $\leq 1-\delta$.*

$$\sum_{t=1}^{T}\Delta V_{t,i} \leq 2\Big(\frac{C_1\beta_T\gamma_T}{2}\Big)^{n/2}\frac{(\frac{\theta_T}{\theta_x}(n+1))^{1/2}}{n!} \;\; for\; T \geq 1 \tag{17}$$

Proof. Similar to Lemma 6 in (Zuluaga et al., 2013), by applying Cauchy-Schwarz inequality on Lemma 1 as $(\sum_{t=1}^{T}\Delta V_{t,i})^2 \leq T\sum_{t=1}^{T}\Delta V_{t,i}^2$, we obtain the following:

$$\sum_{t=1}^{T}\Delta V_{t,i} \leq 2\Big(\frac{C_1\beta_T\gamma_T}{2}\Big)^{n/2}\frac{(T(n+1))^{1/2}}{n!} \;\; for\; T \geq 1 \tag{18}$$

In the worst case, $T \leq \frac{\theta_T}{\theta_x}$, where $\theta_x = \frac{\sum_{i=1}^{n}\theta_i}{n}$.

□

A.1 Runtime Complexity of FLEXIBO

We analyze the run-time complexity of FLEXIBO using Gaussian Processes (GP) and random forests (RF) as surrogate models, separately, in this section. The total complexity of FLEXIBO can be determined by combining complexities of FLEXIBO from modeling, Pareto region construction, and sampling stages.

Let $|\mathfrak{X}| = q$ be the total number of designs in the design space. However, we only consider $|\mathfrak{X}_m| = m$ designs sampled by Monte-Carlo sampling in this approach. We also consider $N_0 + t$ designs to train the surrogate models at each iteration t. Let us consider $s = m + N_0 + t$. Note that by design, $s << q$. As a result, our FLEXIBO algorithm is significantly faster.

Modeling. In the modeling stage, only a small subset of designs is used to train the surrogate models at each iteration. Training a GP with s number of designs takes $\mathcal{O}(s^3 + ms^2)$ (Rasmussen, 2003) time. Training an RF with s designs takes $\mathcal{O}(n_t n_v s^2 \log s)$ time where n_t is the number of trees, and n_v is the number of features used at each level. Determining the uncertainty region of each design $\boldsymbol{x} \in \mathfrak{X}_\mathfrak{m}$ takes an additional $\mathcal{O}(m)$ time. Therefore, the total complexity of the modeling stage for using GP surrogate model is $\mathcal{O}(s^3 + ms^2 + m)$ and for RF surrogate model is $\mathcal{O}(n_t n_v s^2 \log s + m)$.

Pareto Region Construction. In the Pareto region construction stage, we initially determine the non-dominated designs in the design space and later use the non-dominated designs $\boldsymbol{x} \in \mathfrak{U}$ to construct the Pareto fronts. The complexity of finding the non-dominated designs is $\mathcal{O}(m^2)$. The complexity of constructing the Pareto fronts is similar to the complexity of determining the number of designs on the boundary of the convex hull, which can be performed in $\mathcal{O}(m \log m)|$ time if the number of objectives $n = 2$. When $n > 2$, the Pareto fronts are constructed in $\mathcal{O}(m(\log m)^{n-2} + m \log m)$ time (Kung et al., 1975).

Sampling. In the sampling stage, FLEXIBO determines the next sample $\boldsymbol{x}_t$ and objective $f_{t,i}$ for evaluation. To do so, FLEXIBO computes the acquisition function $\alpha_{t,i}(\boldsymbol{x})$ for each design in $\mathfrak{X}_\mathfrak{m}{}^*$ and selects the maximum. To compute $\alpha_{t,i}(\boldsymbol{x})$ across an objective f_i, FLEXIBO needs to compute the volume of the Pareto region P_R by updating the uncertainty values of $\boldsymbol{x}$ in $\mathfrak{X}_\mathfrak{m}{}^*$. This would take $\mathcal{O}(m)$ time as $|\mathfrak{X}_\mathfrak{m}|^* = m$ in the worst case. After measuring the selected design $\boldsymbol{x}_t$ across objective $f_{t,i}$, we update the evaluated designs set S_i and objective evaluation cost $\theta_{t,i}$. All of these are done in constant time and we can safely ignore them in our analysis. Therefore, the total run-time complexity of FLEXIBO in the sampling stage is $O(m)$, regardless of the surrogate model.

Overall. Finally, we determine the overall complexity of FLEXIBO using GP and RF for n objectives by combining the complexities of the three stages discussed above. When GP is used as the surrogate model, the total complexity of FLEXIBO for $n = 2$ objectives is $\mathcal{O}(s^3 + ms^2 + m^2 + m \log m + 2m)$ and for $n \geq 3$ objectives the total complexity is $\mathcal{O}(s^3 + ms^2 + m^2 + m(\log m)^{n-2} + m \log m + 2m)$. To simplify these expressions, we consider $s = m$. Now, the total complexity for $n = 2$ objectives using GP surrogate model is approximately $\mathcal{O}(s^3 + s^2 + s \log s + s)$ and for $n = 3$ objectives is $\mathcal{O}(s^3 + s^2 + s(\log s)^{n-2} + s \log s + s)$.

Similarly, when RF is used as a surrogate model, the total complexity of FLEXIBO for $n = 2$ objectives is $\mathcal{O}(n_t n_v s^2 \log s + m^2 + m \log m + 2m)$ and for $n \geq 3$ objectives is $\mathcal{O}(n_t n_v s^2 \log s + m^2 + m(\log m)^{n-2} + m \log m + 2m)$. After simplification the complexity for $n = 2$ objectives can be written as $\mathcal{O}(s^2 \log s + s^2 + s \log s + s)$ and for $n = 3$ objectives the complexity can be rewritten as $\mathcal{O}(s^2 \log s + s^2 + s(\log s)^{n-2} + s \log s + s)$.

Upon further simplification, we observe that the complexity of FLEXIBO with the GP surrogate model is approximately $\mathcal{O}(s^3)$ and for the RF surrogate model the complexity is $\mathcal{O}(s^2 \log s)$, where s is significantly lower than the total number of designs q.

References

Abdolshah, M., Shilton, A., Rana, S., Gupta, S., & Venkatesh, S. (2019a). Cost-aware multi-objective bayesian optimisation..

Abdolshah, M., Shilton, A., Rana, S., Gupta, S., & Venkatesh, S. (2019b). Multi-objective bayesian optimisation with preferences over objectives..

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

Belakaria, S., Deshwal, A., & Doppa, J. R. (2019). Max-value entropy search for multi-objective bayesian optimization. In *Advances in Neural Information Processing Systems*, pp. 7825–7835.

Belakaria, S., Deshwal, A., & Doppa, J. R. (2020). Max-value entropy search for multi-objective bayesian optimization with constraints..

Cai, E., Juan, D.-C., Stamoulis, D., & Marculescu, D. (2017). Neuralpower: Predict and deploy energy-efficient convolutional neural networks..

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware..

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

Cao, Y., Smucker, B. J., & Robinson, T. J. (2015). 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.

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

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

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

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

Deb, K., & Sundar, J. (2006). 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.

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

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding..

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

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

Fonseca, C. M., Paquete, L., & López-Ibánez, M. (2006). An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE international conference on evolutionary computation*, pp. 1157–1163. IEEE.

Gadepally, V., Goodwin, J., Kepner, J., Reuther, A., Reynolds, H., Samsi, S., Su, J., & Martinez, D. (2019). Ai enabling technologies: A survey..

Guo, T. (2017). Towards efficient deep inference for mobile applications..

Halawa, H., Abdelhafez, H. A., Boktor, A., & Ripeanu, M. (2017). Nvidia jetson platform characterization. In *European Conference on Parallel Processing*, pp. 92–105. Springer.

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

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

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

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

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

Hernández-Lobato, J. M., Hoffman, M. W., & Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pp. 918–926.

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

Iqbal, M. S., Kotthoff, L., & Jamshidi, P. (2019). Transfer Learning for Performance Modeling of Deep Neural Network Systems. In *USENIX Conference on Operational Machine Learning*, Santa Clara, CA. USENIX Association.

Jamshidi, P., & Casale, G. (2016). 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.

Jamshidi, P., Velez, M., Kästner, C., & Siegmund, N. (2018). 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.

Jamshidi, P., Velez, M., Kästner, C., Siegmund, N., & Kawthekar, P. (2017). 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.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient Global Optimization of Expensive Black-Box Functions. *J. of Global Optimization*, *13*(4), 455–492.

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

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

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

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

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.. Citeseer.

Kung, H.-T., Luccio, F., & Preparata, F. P. (1975). On finding the maxima of a set of vectors. *Journal of the ACM (JACM)*, *22*(4), 469–476.

LeCun, Y., & Cortes, C. (2010). MNIST handwritten digit database..

LeCun, Y., et al. (2015). Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, *20*(5), 14.

Lee, E. H., Perrone, V., Archambeau, C., & Seeger, M. (2020). Cost-aware bayesian optimization..

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., & Kavukcuoglu, K. (2017). Hierarchical representations for efficient architecture search..

Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search..

Lokhmotov, A., Chunosov, N., Vella, F., & Fursin, G. (2018). 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*, p. 6. ACM.

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

Mozilla (2019). `https://commonvoice.mozilla.org/en/datasets`.

Nair, V., Yu, Z., Menzies, T., Siegmund, N., & Apel, S. (2018). Finding faster configurations using flash.. IEEE.

Nardi, L., Koeplinger, D., & Olukotun, K. (2019). 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.

Paria, B., Kandasamy, K., & Póczos, B. (2018). A flexible framework for multi-objective bayesian optimization using random scalarizations..

Pei, K., Cao, Y., Yang, J., & Jana, S. (2017). Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pp. 1–18.

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

Pereira, J. A., Martin, H., Acher, M., Jézéquel, J.-M., Botterweck, G., & Ventresque, A. (2019). Learning software configuration spaces: A systematic literature review..

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

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

Ponweiser, W., Wagner, T., Biermann, D., & Vincze, M. (2008). 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.

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

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text..

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer.

Reuther, A., Michaleas, P., Jones, M., Gadepally, V., Samsi, S., & Kepner, J. (2019). Survey and benchmarking of machine learning accelerators. In *2019 IEEE high performance extreme computing conference (HPEC)*, pp. 1–9. IEEE.

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

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

Roijers, D. M., Zintgraf, L. M., & Nowé, A. (2017). Interactive thompson sampling for multi-objective multi-armed bandits. In *International Conference on Algorithmic DecisionTheory*. Springer.

Rouhani, B. D., Mirhoseini, A., & Koushanfar, F. (2016). 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.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, *115*(3), 211–252.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.

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

Shapiro, A. (2003). Monte carlo sampling methods. *Handbooks in operations research and management science*, *10*, 353–425.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. W. (2012). Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, *58*(5), 3250–3265.

Sun, Y., Wu, M., Ruan, W., Huang, X., Kwiatkowska, M., & Kroening, D. (2018). Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pp. 109–119.

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

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

Wang, Z., & Jegelka, S. (2017). Max-value entropy search for efficient bayesian optimization..

Whatmough, P. N., Zhou, C., Hansen, P., Venkataramanaiah, S. K., Seo, J.-s., & Mattina, M. (2019). Fixynn: Efficient hardware for mobile computer vision via transfer learning..

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., & Keutzer, K. (2019). 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.

Zela, A., Klein, A., Falkner, S., & Hutter, F. (2018). Towards automated deep learning: Efficient joint neural architecture and hyperparameter search..

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

Zintgraf, L. M., Roijers, D. M., Linders, S., Jonker, C. M., & Nowé, A. (2018). 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.

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

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

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