Evaluating Bayesian Optimization for Electronic High-Speed Design Applications

Jan Krummenauer, Nesrine Kammoun
Cross-Domain Computing Solutions
Robert Bosch GmbH
Leonberg, Germany
{Jan.Krummenauer,Nesrine.Kammoun}@de.bosch.com

Juergen Goetze
Information Processing Lab
TU Dortmund
Dortmund, Germany
Juergen.Goetze@tu-dortmund.de

Abstract*— Bayesian Optimization (BO) is an increasingly popular method for automating costly design optimization tasks in many fields of science and engineering. In this paper, we evaluate the application of such surrogate model-based optimization methods in the field of electronic high-speed design. For this purpose, we present a design of a bias-tee circuit required for power transmission over a communication link in an automotive environment. Based on a low-fidelity version of the costly-to-evaluate design task, we analyze the sampling efficiency of BO. Besides this, we discuss the handling of categorical variables in BO, which are frequently encountered in engineering tasks. Our experiments show that with suitable variable handling, BO outperforms other state-of-the-art optimization methods significantly. Lastly, we discuss additional benefits of applying surrogate model-based optimization methods, such as direct access to feature importance metrics, possibilities to use novel surrogate models, and the option to apply transfer learning to optimization.

Keywords — Bayesian optimization, high-speed design, electronic design automation, optimization benchmarks, feature importance, surrogate modeling, Bayesian neural networks, transfer learning.

I. INTRODUCTION

Bayesian Optimization (BO) is a state-of-the-art surrogate model-based optimization (SMBO) method often applied to very time-consuming and costly optimization problems. The most popular application of BO is neural architecture search (NAS) [1], but it is also increasingly applied in other domains, such as material science, chemistry [2, 3], or electronic design automation (EDA) [4, 5, 6].

Electronic design tasks often involve long simulation runtimes, especially when costly 3D simulations are required. In printed circuit board (PCB) design for example, such expensive simulations are used for signal integrity (SI), power integrity (PI), or electromagnetic compatibility (EMC) design. In this context, we evaluate the application of BO for optimization tasks in the field of electronic high-speed design, for which we present a design of a wideband bias-tee circuit for automotive applications. In this particular design, there are several options for circuit elements that must be selected, while suitable PCB layout parameters must be defined in parallel. To analyze the parasitic layout effects on the high-speed characteristics, the design evaluation involves a computationally intensive 3D simulation. Thus, the design should be optimized with as few simulative evaluations as possible, which demands a high *sampling efficiency* of an applied optimization method.

Bayesian Optimization is associated with such a high sampling efficiency [7, 8]. However, applying BO in an engineering context often presents a unique set of challenges. On the one hand, the given design task must be formulated as an optimization problem. Such a problem setup typically poses several obstacles such as defining a suitable optimization target, dealing with multiple variable types, or automating design variations. On the other hand, the optimization method itself raises application-oriented challenges. Since BO is typically applied to expensive optimization tasks, the validation of its sampling efficiency is difficult, as a large number of optimization runs is needed for statistically relevant conclusions. Also, choices in the optimization experiment setup can affect this sampling efficiency, such as different approaches for handling categorical variables. Due to the large number of optimization runs required to analyze these effects, they are often neglected, which can lead to incorrect assumptions about the effectiveness of an applied BO method.

In the first part of this paper, we address these application-oriented problems. First, we introduce the highspeed design task and formulate it as an optimization problem. We provide detailed insights into the definition of an optimization objective, the handling of categorical variables, and the required automation steps. Next, we conduct benchmark experiments to evaluate the sampling efficiency of BO in comparison to other state-of-the-art optimization methods. We also examine how choices in the experiment setup, such as categorical variable encodings, impact this optimization performance. By using a low-fidelity version of the costly-to-evaluate high-speed design task, we present an approach on how to evaluate the sampling efficiency in the context of computationally intensive optimization problems. Subsequently, we use insights gained from these benchmark experiments to select the most suitable settings for the high-fidelity design optimization.

In the second part of this paper, we present three additional experiments to demonstrate further advantages of SMBO methods beyond their sampling efficiency. We provide an example of how feature importance metric can be analyzed directly via the surrogate model. Additionally, we present an application of an advanced surrogate model, which can be used to give more informative predictions in an optimization experiment. Furthermore, we show how transfer learning in BO can further improve its sampling efficiency. The respective results serve as an outlook on future opportunities in this field.

To summarize, our main contributions:

- The sampling efficiency of BO is compared to other state of the art black-box optimization methods in benchmark experiments. An approach is presented on how to evaluate the sampling efficiency of optimization algorithms for computationally expensive design tasks via highly comparable low-fidelity versions.
- Using an exemplary highspeed PCB design, it is shown how a simultaneous choice of circuit and

layout design parameters can be formulated as an optimization problem. In addition, necessary automation steps for applying BO to the design are presented.

- Additional benefits of SMBO methods are highlighted from an application-oriented perspective, which provides an outlook on future opportunities in this domain.

Section II introduces theoretical background of the BO method and discusses current applications in science and engineering. In section III, we present a detailed setup of the high-speed design task, the optimization objective, and necessary steps for the design automation. Lastly, section IV presents the results of the optimization experiments, while section V highlights additional benefits of applying SMBO methods.

II. Bayesian Optimization Method

Bayesian Optimization [9, 10] is an iterative black-box optimization (BBO) method that consists of three main steps: Black-box function evaluation (data collection), surrogate model training, and acquisition function maximization (see Figure 1). These steps are executed iteratively until either a maximum number of evaluations (budget) is reached, or a required optimization target is achieved.

The highspeed design task introduced in section III corresponds to the black-box function $y(\boldsymbol{x})$. Next, we will introduce surrogate models and acquisition functions, which are commonly applied in BO.

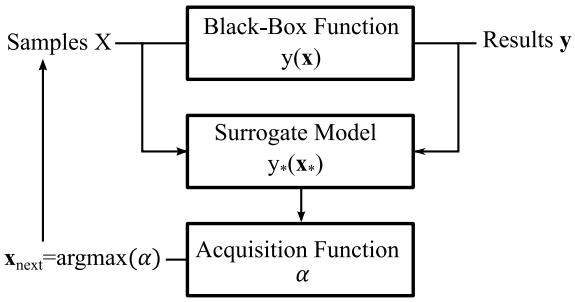

Figure 1: Schematic representation of the Bayesian Optimization method.

A. Probabilistic surrogate model

A probabilistic surrogate model $y_*(\boldsymbol{x}_*)$ is used to predict a mean $\bar{y}_*(\boldsymbol{x}_*)$ and an uncertainty, represented by a standard deviation $\sigma(\boldsymbol{x}_*)$, of a black-box function. $\boldsymbol{x}_*$ is an unknown sample position, while $X = (\boldsymbol{x}_i \mid i = 0, \dots, n)$ and $\boldsymbol{y} = y(X)$ describe already evaluated samples.

The most common surrogate models applied in BO are Gaussian Processes (GP). However, it is also possible to use other probabilistic machine learning models such as Bayesian Neural Networks (BNN), of which we will present a suitable application in section V.B.

1) Gaussian Processes

Using a noise-free Gaussian Process regression model [11] requires first the selection of a prior mean $m_{\text{prior}}(\boldsymbol{x})$ and a prior kernel function $k_{\text{prior}}(\boldsymbol{x})$ (also known as covariance function). $m_{\text{prior}}(\boldsymbol{x})$ typically models an offset in the data. For simplicity, however, we assume that the given data has a zero-mean ($m_{\text{prior}}(\boldsymbol{x}) = 0$), which can be implied for standardized data. $k_{\text{prior}}(\boldsymbol{x})$ describes correlation between samples. Most common kernel functions are the radial basis function (RBF) and Matern kernels, which model spatially-exponentially decreasing correlations. Moreover, it is possible to use other kernel types, such as periodic kernels, or to combine multiple kernels mathematically.

After choosing priors, the hyperparameters of the kernel function are fitted via the Log-Marginal-Likelihood-Maximization method using observed (training) data $(X, \boldsymbol{y})$. Subsequently, the posterior (trained) GP can be derived and is defined as [11]:

$$\boldsymbol{y}_* | X_*, X, \boldsymbol{y} \sim \mathcal{N}(\bar{\boldsymbol{y}}_*, K_{\boldsymbol{y}_*})) \quad (1)$$

with:

$$\bar{\boldsymbol{y}}_* = K(X, X_*)^T \cdot K(X, X)^{-1} \cdot \boldsymbol{y} \quad (2)$$

$$K_{\boldsymbol{y}_*} = K(X_*, X_*) - K(X, X_*)^T \cdot K(X, X)^{-1} \cdot K(X, X_*) \quad (3)$$

Where K describes covariance matrices sampled from the kernel function at known locations X and unknown locations X_*. $\bar{\boldsymbol{y}}_*$ is the prediction for the mean of the black-box function, while $K_{\boldsymbol{y}_*}$ is the resulting covariance matrix of $\boldsymbol{y}_*$. The standard deviation $\boldsymbol{\sigma}$ can be calculated by taking the square-root of the diagonal of $K_{\boldsymbol{y}_*}$:

$$\boldsymbol{\sigma}(X_*) = \sqrt{\text{diag}(K_{\boldsymbol{y}_*})} \quad (4)$$

The most computational-intensive part of the GP training is the calculation of the inverse covariance matrix $K(X, X)^{-1}$. Thereby, the runtime of standard BO methods typically scales with $\mathcal{O}(n^3)$ to the number of evaluations n.

2) Bayesian Neural Networks

Recent studies suggest the use of Bayesian Neural Nets as surrogate models for BO [12, 13]. Benefits of these approaches are a linear scaling runtime, which is especially important when observing a large quantity of samples n. In addition, the data formats at the inputs and outputs have a larger flexibility, which enables including auxiliary information for optimization.

A downside of using these models in BO is a higher computational effort from the start on (for a small number of samples n). Also, BNNs are parametric models, which add additional hyperparameters for the model complexity to choose before the optimization. GPs in contrast are inherently non-parametric models, which intrinsically add complexity with increasing amount of training data [14].

B. Acquisition functions

After training a surrogate model, an acquisition function α is applied. This function uses the surrogate model predictions to search for informative new sample positions where the black-box function should be evaluated in the next iteration. In this process, the tradeoff between exploration (evaluating $y(\boldsymbol{x}_*)$ at $\boldsymbol{x}_*$ where $\sigma(\boldsymbol{x}_*)$ can be reduced) and exploitation (evaluating $y(\boldsymbol{x}_*)$ at $\boldsymbol{x}_*$ where $\bar{y}_*$ improves over the current best value found) is handled.

Common acquisition functions are probability of improvement (PI), expected improvement (EI), and upper confidence bound (UCB) [15]. In practice, EI is often a default choice in many BO frameworks and is formulated as:

$$EI(\boldsymbol{x}) = \mathbb{E}(\max(\bar{y}_* - y_{\text{best}}, 0)) \quad (5)$$

Where $\mathbb{E}$ is the expected value and y_{best} is the best observation so far. A more detailed definition is given in [16]. The next sample $\boldsymbol{x}_{\text{next}}$ is found by maximizing α:

$$\boldsymbol{x}_{\text{next}} = \text{argmax}(\alpha(\boldsymbol{x})) \quad (6)$$

This method typically results in a new sample position, which significantly increases the informational content of the acquired dataset.

C. Application of BO in engineering and science

BO methods are widely applied in various domains of science. For example, in chemistry or material design [17, 18, 12], BO is used discover new materials or molecules. These optimization problems often show a very comparable structure to engineering problems with long simulation runtimes and mixed variable spaces [19]. In electronic engineering, BO is increasingly applied e.g. in chip design [20, 21] to tune parameters of chip architectures or RF-circuits. Further applications include analog circuit designs [22, 23, 24], SI [25], or EMC [26] design tasks.

However, application-oriented studies often lack a convincing validation for the effectiveness of the applied BO approach. This is typically due to the optimization problems being computationally intensive, which often permits only a limited number of optimization runs. As a consequence, conclusions about a methods efficiency are often drawn based on very small sample sizes or even single optimization runs.

In Section IV, our experiments show that, similar to other BBO techniques, the performance of BO is stochastic. Hence, it is crucial to have a sufficiently large sample size of optimization trials when comparing the sampling efficiency of optimization methods.

Since achieving this large number of optimization trials is typically not possible on the computationally intensive optimization problems, we introduce an approach that utilizes low-fidelity versions of computationally intensive experiments to evaluate the effectiveness of applied optimization techniques in a highly comparable environment.

III. High Speed Design Task & Design Automation

In this section, a design of a broadband bias-tee filter circuit is introduced, to which we will later apply the optimization methods and evaluate their performance.

A. Design task introduction

In an automotive environment, a camera sensor unit is connected via an RF-coax-cable to an electronic control unit (ECU). With this connection, a broadband video signal is transmitted to the ECU by a so called "Gigabit multimedia serial link" (GMSL) [27]. In addition, the power for the camera sensor unit must be transferred over the same connection, which requires a DC bias on the transmission line. Therefore, a filter circuit must be placed at both ends of the transmission line to separate DC and RF signal contents. Such a filter circuit is called a Power-over-Coax (PoC) filter, or more generally a bias-tee.

The PoC filter module will be used in multiple instances in several high-volume Advanced Driver Assistance System (ADAS) products, so there is a high interest in finding optimized solutions. An overview of such a filter setup is given in Figure 2a), with a more detailed representation of the PoC-circuit given in Figure 2b).

For the design of this filter circuit, there are a lot of choices for inductive components, resistors, and layout parameters. We will introduce these parameters, which will be the input variables $\boldsymbol{x}$ for the optimization problem in the next section III.B.

The connected SerDes components apply strict specifications to the RF- path given as S-parameter limits for the transmission and return loss (S_{21},S_{11}). The optimization goal is to comply with a best possible margin to these limits. We explain this criterium, and the design of a suitable target function assigning a scalar score for optimization in section III.C. This score will be the output $y(\boldsymbol{x})$ of the optimization problem.

In section III.D, we outline necessary automation steps required for the evaluation of this design, implemented in a state-of-the-art pythonic toolchain. This automation is crucial for applying any automated optimization approach, underscoring the increasingly important role of EDA in PCB-design.

B. Design variables and variable encodings

The optimization problem has 6 input design variables $\boldsymbol{x}$, which are shown including their boundaries in Figure 2c:

- L_1, L_2, L_3 : Inductors out of a set of 45 possible inductive devices $\boldsymbol{S}_{\text{L}}$ consisting of coils and ferrite beads. Each device is described by its S-parameters and has an individual footprint. The choice to place an electric short is also possible.

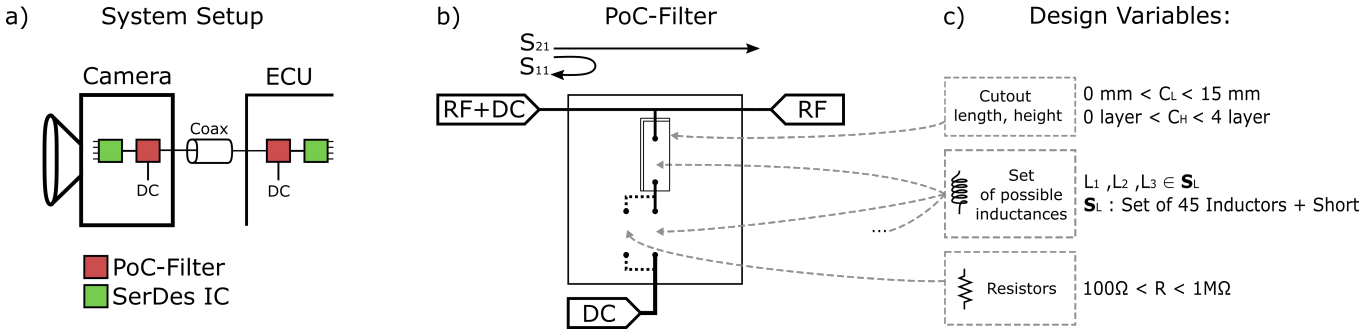

Figure 2: Setup of the Power over Coax Filter design task. a) shows the system environment consisting of a camera sensor connected to an ECU. b) shows an exemplary filter design setup with corresponding design variables shown in c).

- R : Value of a damping resistor which is placed in parallel to large inductors.
- C_L, C_H: Length and height of a ground layer cutout, placed below the inductive devices. The cutout starts at the RF-path and has a fixed width.

While R, C_L and C_H can be treated as standard continuous or discrete variables in this design, the inductive devices represent categorical variables as they can be described by multiple important physical descriptors. For the categorical variables, a numerical encoding must be defined, which must be chosen with caution in BO [28].

A typical approach is to either use One-Hot-Encodings (OHE) or Integer Encodings (IE). In OHE, the physical descriptors of the categorical variable are neglected. The categorical variable with number of choices m is mapped into an m -dimensional space, with its choices distributed at uniform positions on orthogonal axes. Thereby all categorical options have the same Euclidean distance to each other, which results in uniform spatial correlations in BO. In contrast to that, in an IE a single physical descriptor is used to define an order for the categorical options along one axis. This results in smaller Euclidean distances between physically similar choices and thus higher spatial correlation between similar samples in BO.

In the case of the inductors, multiple physical descriptors can be derived from their S-parameters, which are exemplarily shown in Figure 3. Possible descriptors for the shown inductors are for example the resonance frequency f_r, the maximum insertion loss $S_{21\mathrm{max}}$, metrics for the broadband behavior, etc. Each of these descriptors can be used to define an IE.

There are also more advanced methods for encoding categorical variables in BO, which also allow considering multiple physical descriptors simultaneously for variable encodings [2, 3, 29]. However, benchmark experiments presented in [29] indicate that simpler encoding approaches, such as using IEs in combination with standard BO frameworks, often achieve a competitive sampling efficiency to the more complex approaches when dominant physical descriptors are considered for the encoding. Furthermore, it is shown that the standard BO frameworks have a faster runtime, which makes them easier to use.

This is why, in chapter IV.A we focus on comparing OHE and IEs based on two descriptors in particular: f_r, and the area under the S_{21} curve ($S_{21-\mathrm{AUC}}$), which acts as combined measure of bandwidth and damping. The goal is, to identify the optimal variable encoding for the inductors in low-fidelity benchmarks, which can then be applied in high-fidelity design optimization.

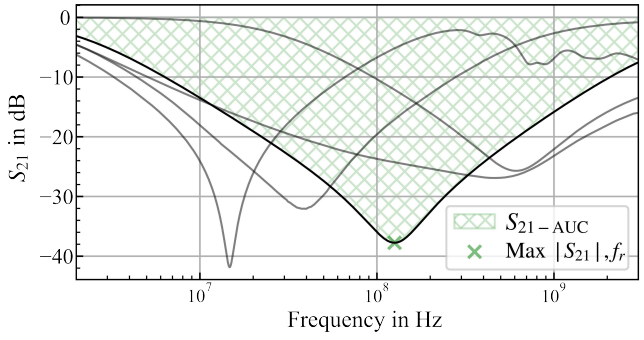

Figure 3: S_{21} of some inductive devices which can be applied to the filter circuit. The curves can be characterized with multiple physical descriptors indicated by the hashed area and the marker. These physical descriptors can be used to find suitable variable encodings for the inductors.

C. Optimization objective

The optimization objective described in this section is used as the scalar output of our black box function $y(\boldsymbol{x})$. Figure 4 shows the given GMSL limits for the S-parameters in red. The black line shows an exemplary result of a design evaluation. To obtain an optimization goal G we calculate the sum of the distances between the evaluation results $\boldsymbol{S}_{11}, \boldsymbol{S}_{21}$ and their limits adjusted by a weighting function:

$$\boldsymbol{d}_{11} = \boldsymbol{S}_{11\mathrm{lim}} - \boldsymbol{S}_{11}\ ;\ \boldsymbol{d}_{21} = \boldsymbol{S}_{21} - \boldsymbol{S}_{21\mathrm{lim}} \qquad (7)$$

$$G = \sum_{\mathrm{i}=1}^{f_L} w_{11}(d_{11_\mathrm{i}}) + w_{21}(d_{21_\mathrm{i}}) \qquad (8)$$

Where $\boldsymbol{d}$ describes the distance of the S-Parameters to their limits and f_L corresponds to the length of the S-Parameters frequency vector. The weighting functions w_{11} and w_{21} are chosen so that limit violations are weighted negatively, while limit compliance will contribute positively to G:

$$w_{11}(d) = \begin{cases} m_1 + |d| \cdot 0.1, & \text{for } m_1 < d \\ |d| \cdot 1, & \text{for } m_2 < d \le m_1 \\ (|d| - m_2) \cdot 10, & \text{for } 0 < d \le m_2 \\ (|d| - m_2) \cdot 100, & \text{for } d \le 0 \end{cases} \qquad (9)$$

$$w_{21}(d) = \begin{cases} |d| \cdot 1\,, & \text{for } m_3 < d \\ (|d| - m_3) \cdot 10, & \text{for } 0 < d \le m_3 \\ (|d| - m_3) \cdot 100, & \text{for } d \le 0 \end{cases} \qquad (10)$$

m_2 and m_3 are defined as margins close to the limits, which also result in a negative contribution to G if passed (we choose m_2 = 5 dB and m_3 = 0.5 dB). m_1 is a margin applied

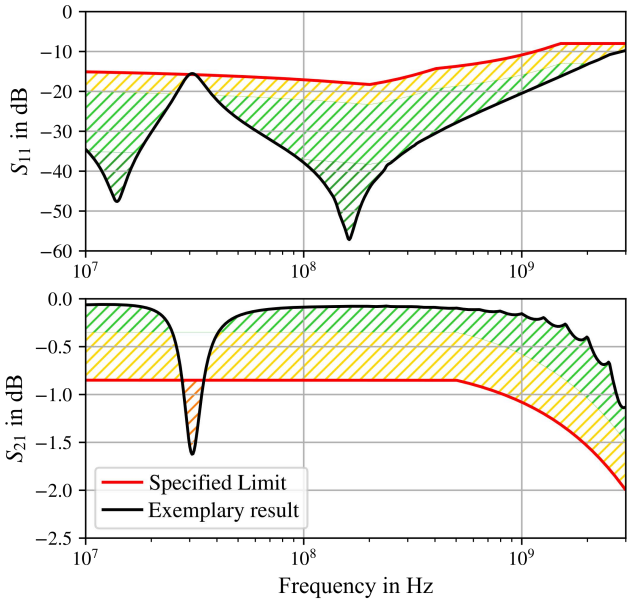

Figure 4: Exemplary representation of a design evaluation result. The colored, hashed areas depict the weighting function, with green areas contributing positively to G, while orange and red areas contribute negatively to G.

to the return loss, limiting positive contributions of deep peaks (here m_1 = 20 dB; The best insertion loss is inherently limited to 0 dB).

The weighting is visualized in Figure 4. Evaluation results crossing the orange and yellow regions contribute negatively to G, while outcomes that lie in the green regions contribute positively to G. The objective is to maximize G and thereby the green region.

G is a scalar value here, for the use as a single objective. For a multi-objective approach, G could be modified to a vector, e.g. generating multiple goals for different frequency regions.

D. Design Automation

A manual evaluation of this design is a time-consuming and highly repetitive task. After deciding for a sample $\boldsymbol{x}$, the evaluation process consists of the steps shown in Figure 5. We automated this design process using PyAEDT [30] and Scikit-RF [31] as explained below:

- PCB layout adjustments:
  The PCB is generated, including right footprints for the selected inductive parts. Then, a cutout is inserted in the ground (GND) plane below the inductors. This process is automated using PyAEDT.
- PCB layout simulation:
  In a next step, the parasitic effects of the PCB are simulated using Ansys HFSS. The simulation involves pre- and post-processing tasks, such as port placements, simulation sweep setups, net naming, etc., which are also implemented using the python interface.
- Device connection:
  Here, the S-parameters of the inductive devices are cascaded to the simulated PCB. This step also involves pre-processing effort, such as interpolating S-parameters along a predefined frequency axis, which is why we implemented this process using the Scikit-RF toolbox.

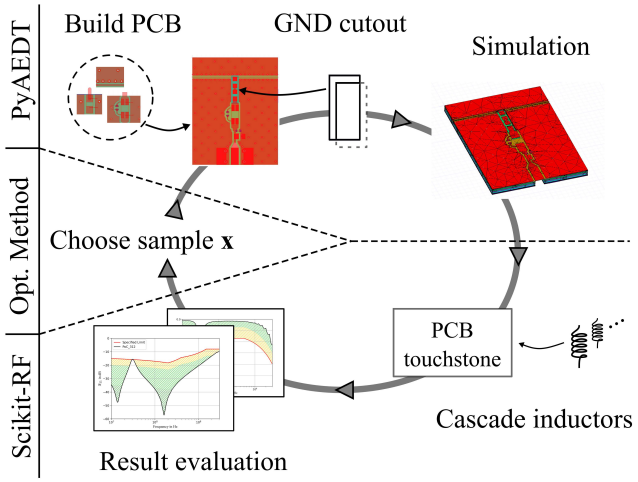

Figure 5: Overview of the required automation process for the evaluation of the filter module. The sample is chosen by an optimization method, the PCB is built and simulated via PyAEDT. Scikit-RF is used for preprocessing, cascading the devices S-parameters, and to calculate the optimization goal G.

- Result evaluation:
  In a last step, the S-parameter results of the PCB with placed inductors is evaluated and the previously described optimization objective is computed (see section III.C).

The runtime of evaluating a single sample is 9 minutes using the full wave 3D simulation (HFSS solver). The simulations are conducted on a workstation with an Intel Xeon Gold 6128 CPU and a Nvidea Quadro RTX 4000 GPU. Thus, the cumulative runtime of several hundred simulations required for optimization is in the order of several days, which emphasizes the importance of using as few simulation steps as possible.

E. Low-fidelity benchmark designs

In the previous section, we introduced the design automation required to evaluate the bias-tee design in a high-fidelity setting. However, benchmarking and comparing optimization methods is impractical on optimization problems involving such computationally intensive evaluations due to the large amount of necessary optimization trials. Therefore, we introduce two low-fidelity versions of this design, which we use to run benchmark experiments in chapter IV and chapter V:

- In a first low-fidelity version of the design task (LF1), the placement of two inductors (corresponding to two categorical input variables) is considered on a fixed PCB layout. Thereby, the 3D simulation is evaluated only once and evaluations in the benchmarks only require cascading the S-Parameters of the inductors to the fixed PCB, resulting in a short evaluation runtime. This low-fidelity version of the design enables us to evaluate the optimization methods and variable encoding techniques in a search space, which is highly comparable to the high-fidelity design.
- In another low-fidelity setting (LF2), we cascade the inductors directly in a bias-tee configuration without any layout considered. This low-fidelity function is also fast to evaluate and is used in section V.B and section V.C.

IV. Optimization Experiments

To compare the sampling efficiency of BO to other state-of-the-art optimization methods and to analyze the impact of categorical variable encodings, we first evaluate benchmark experiments on toy functions. After that, we compute similar benchmarks on the low-fidelity version (LF1) of the design task to further analyze the performance of the considered methods on a function, which is comparable to the high-fidelity experiment. The method with the highest sampling efficiency in the benchmarks is then applied to the high-fidelity version of the design.

In the benchmark experiments, we evaluate the sampling efficiency of the following algorithms:

- BO using the GPyOpt [32] framework with a GP surrogate model and a Matern52 kernel function. Expected Improvement is used as the acquisition function. To show the impact of different categorical variable encodings on the sample efficiency, we test OHE and different physical descriptor-based IEs.

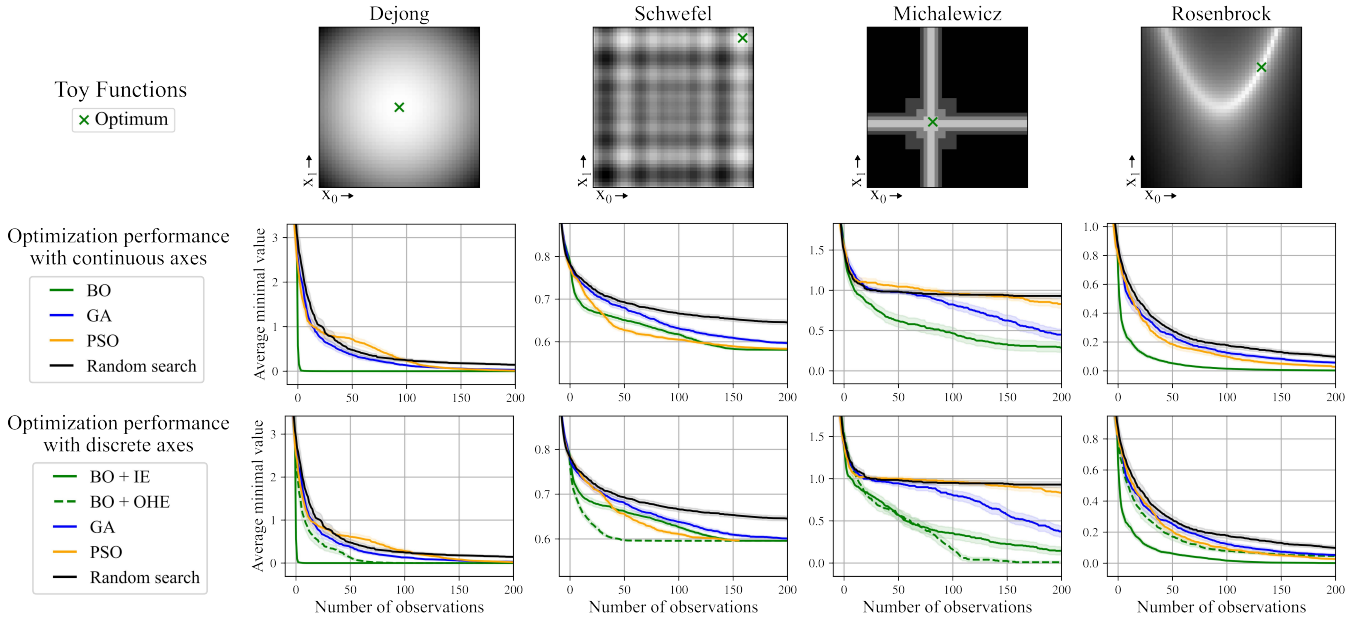

Figure 6: Benchmark experiments on toy functions. The top row depicts analyzed toy functions with their global optimum highlighted with a green x. The middle row shows the optimization performance plots on the functions with continuous axes. The bottom row shows the performance on functions with discretized axes, where the impact of the variable encoding techniques can be observed (IE vs. OHE). 95% confidence intervals are given in the shaded area along the curves.

- Genetic algorithm (GA) and particle swarm optimization (PSO), which we both implemented using the pymoo [33] framework. For both algorithms, the population size was set to 20. For other hyperparameters, we used the default values suggested by the framework.
- Random search is used as baseline metric.

For the benchmarks we run 200 trials of each algorithm on every function, which results in small-sized 95% confidence intervals. We always show a minimization of the function values in the experiments (if the objective is to maximize, its negative is shown). All optimization methods are initialized with 10 random samples before the start of strategic picks. We defined a maximum budget (maximum number of observations) per trial of 200, which corresponds to 10% of the size of the search space when discretized axes are considered.

A. *Sample efficiency benchmarks*

Figure 6 shows optimization benchmark experiments on toy functions. Representations of the functions are shown in the top row. The second row presents the average optimization performance on the toy functions having two continuous axes, while in the third row all axes are discretized. Using discretized functions, we can compare the effect of using ideal IEs or OHE for categorical variable encodings in BO.

On all toy functions, the BO method shows the best average optimization performance. On the functions with continuous axes, BO outperforms the other methods significantly in 3 of 4 experiments. Only on the Schwefel function, which has a lot of local optima, GA and PSO perform competitively.

When considering discretized axes, we observe similar performances to the continuous setup. However, the choice of variable encoding shows a significant impact on the performance of BO. OHE leads to an improved performance on the Schwefel function, which has a higher degree of spatial complexity than the other functions. On the Dejong and Rosenbrock functions, which are smoother, IE outperforms OHE.

This experiment shows on the one hand, that BO indeed often shows a higher sampling efficiency in comparison to other state-of-the-art methods. On the other hand, when dealing with categorical variables, the selection of a suitable encoding is crucial to achieve this superior optimization performance.

Next, we evaluate the optimization methods using the low-fidelity benchmark version of the design (LF1, see chapter III.E), where the choice of two inductors is evaluated on a fixed layout. Thereby, we can analyze the performance of BO using different encodings for the categorical input variables before running the computationally intensive optimization.

Figure 7 shows the benchmark results of the optimization runs on the low-fidelity function. Again, BO outperforms the GA, PSO, and the random search baseline. In terms of

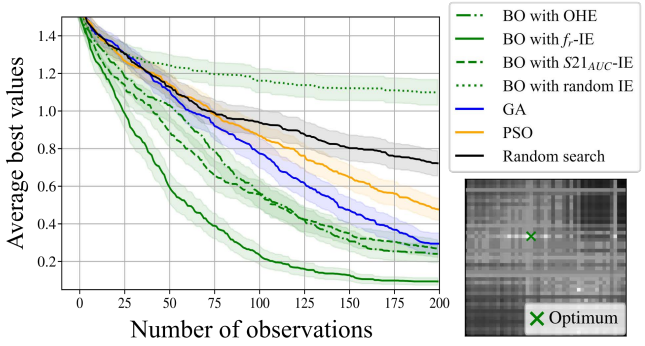

Figure 7: Optimization performance comparison on a low-fidelity benchmark function (LF1), which is comparable to the high-fidelity design introduced in chapter III. The bottom right image represents the LF1 benchmark function with a f_r-IE applied to the categorical variables, resulting in a well-structured search space.

sampling efficiency, the best BO approach outperforms random search by a factor of 5, PSO by a factor of 3 and GA by a factor of 2.5.

The variable encoding again shows a significant impact on the optimization performance. For instance, a random choice of an IE reduces the performance of the BO method to a point where it is even less effective than random search. This can be explained by the random structure induced by the random IE to the search space. The best performance is shown by the IE utilizing the f_r descriptor, outperforming the other tested physical descriptor S_{21-AUC} and OHE. The image in the bottom right corner shows the function with the f_r-IE applied to the categorical input variables. This encoding results in a relatively smooth and well-structured function, which based on our previous experiments and [28] increases the probability of achieving a good BO performance.

To emphasize the stochastic behavior of the discussed algorithms and to highlight the need of sufficiently large sample sizes of optimization trials when comparing optimization methods, Figure 8 depicts an explosion plot of all trials of two optimization methods applied in the previously shown experiment (see Figure 7). The green curves show all optimization trials of BO (with the f_r-IE), the blue curves illustrate the results of the GA method.

While the best and worst runs of both optimization methods are comparable, on average, BO is more effective than the GA. This highlights that optimization methods cannot be compared based on single optimization runs or using a too small number of trials. In addition, BO does not automatically ensure a high sampling efficiency in single trials. Instead, BO *increases the probability* of achieving a high sample efficiency in a trial.

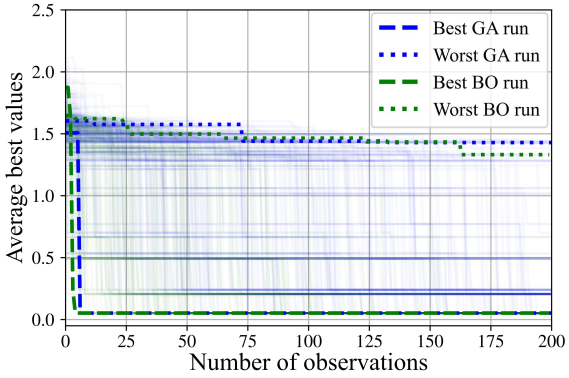

Figure 8: Explosion plot of the GA and BO optimization runs on the low-fidelity function LF1 (see Figure 7). The best and worst runs are shown as dashed and dotted curves. All other runs are shown with a weaker opacity.

B. *High-fidelity design optimization*

As indicated above, a comparison of optimization methods is not viable on the high-fidelity design task, as the large runtime does not allow a large enough number of optimization trials. However, since the benchmark experiments were conducted on a benchmark function (LF1) that closely resembles its high-fidelity version, we can expect a similar optimization performance on the high-fidelity optimization experiment [34]. The benchmark results indicate that the BO method with f_r-IEs performs significantly better than the other methods, making it the best choice for the high-fidelity optimization.

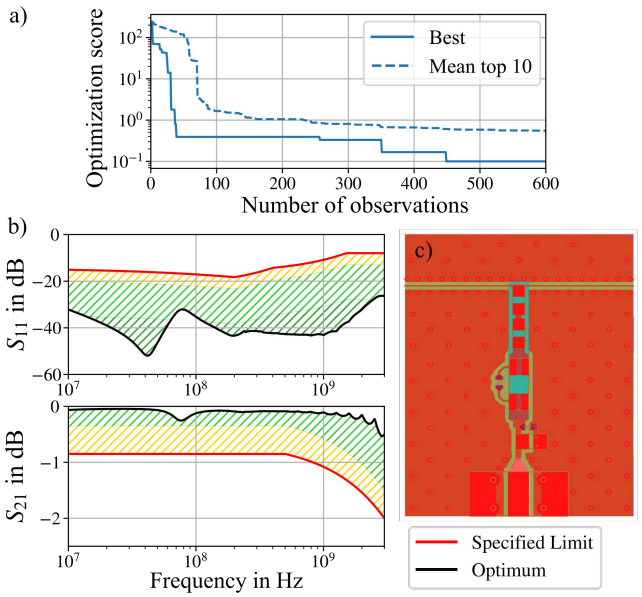

Figure 9: a) shows the optimization score and the average score of the top 10 found values over the number of observations. The best score is here scaled to 0.1, corresponding to the best goal G). b) shows the S-parameter of the optimum sample, with the corresponding PCB-layout shown in c).

Figure 9 shows the optimization results of this method applied to the high-fidelity optimization problem. Figure 9a) indicates, that the best value is found after 450 evaluations. The top 10 values found in the trial also stop improving within this region, indicating that the search space is sufficiently explored. The best score was achieved with the layout shown in Figure 9. The optimum parameter configuration proposes the use of two identical ferrite beads for the inductors L_0 and L_1 (top and middle position) and a larger inductor for L_2 (bottom position). A damping resistor R in parallel to L_2 was chosen to 10kΩ. The top 10 results provide a broader image of optimal designs:

- Other combinations of devices may also be used (with their footprint impacts considered in simulation).
- S_{11} peaks at low frequencies can be limited by choosing a smaller R.
- Cutout dimensions should be chosen as large as possible.

V. Additional Benefits of SMBO Methods

In the previous chapter, we highlighted the sample efficiency of the BO method and presented its application to the high-fidelity design optimization. Besides this, BO offers other advantages, which we emphasize in this section. Section V.A presents how design guidelines can be derived based on feature sensitivity metrics. After that, in section V.B, it is shown how advanced surrogate models enable more informative surrogate model predictions. Finally, section V.C demonstrates the option to use transfer learning in BO, which can further enhance its sampling efficiency.

A. *Feature Importance*

Feature importance metrics can be used to analyze the impact of design variables on the optimization score. In a BO experiment, they can directly be computed after an optimization run using the surrogate model. Suitable methods for that are:

- Feature permutation [35], which involves iteratively permutating a feature of a test dataset while measuring the decrease in the surrogate model's prediction score, such as R^2.
- Model intrinsic measures, such as mean decrease of impurity (MDI) [36] in random forest models.
- Feature elimination [37], where comparable to feature permutation, a decrease in a test score is computed. Instead of iteratively permutating a feature, it is completely removed from the dataset. This requires a retraining of the surrogate model.

For the evaluation shown in Figure 10 we apply the feature elimination method measuring a decrease in R^2 score. We use the data obtained by the BO trial on the high-fidelity design shown in section IV.B. Further, we include a random feature as a baseline, which should not decrease the surrogate model performance if removed. To consider statistical deviations, we applied a 10-fold cross-validation to the measurements. Also, the frequency range is divided into 3 parts to evaluate the importance of the features for different frequencies:

- Low f: 2 MHz to 100 MHz
- Mid f: 100 MHz to 1 GHz
- High f: 1 GHz to 3.5 GHz

Figure 10 shows the calculated feature importance metrics of the design variables (see section III.B) for insertion loss and return loss scores. In general, we can observe that for low frequencies, L_0, L_1 and R choices are the most important parameters. At mid to high frequencies, the L_0 choice (which is the inductance closest to the RF-path) is of highest importance. The layout parameters have no impact at the low frequency range, but their importance increases at mid and high frequencies. The impact of the L_2 inductor is only noticeable in the low frequency insertion loss scores, which can be explained by a wider design flexibility for this inductor.

These observations can be used to derive design guidelines, such as:

- The inductor responsible for high-frequency damping should be placed nearest to the RF-path (L_0).
- The other inductors affect the filter performance mostly in the low and mid frequency range.
- The cutout has the highest impact in the mid and high frequency range and should be chosen as large as possible.
- Modifying the damping resistor R can improve the filter characteristic at low and mid frequencies (since it is placed in parallel to large inductors in this design).

These design guidelines match with the know-how of an experienced designer for this circuit. Extracting such guidelines automatically enables less experienced designers to understand significant correlations for a specific design task.

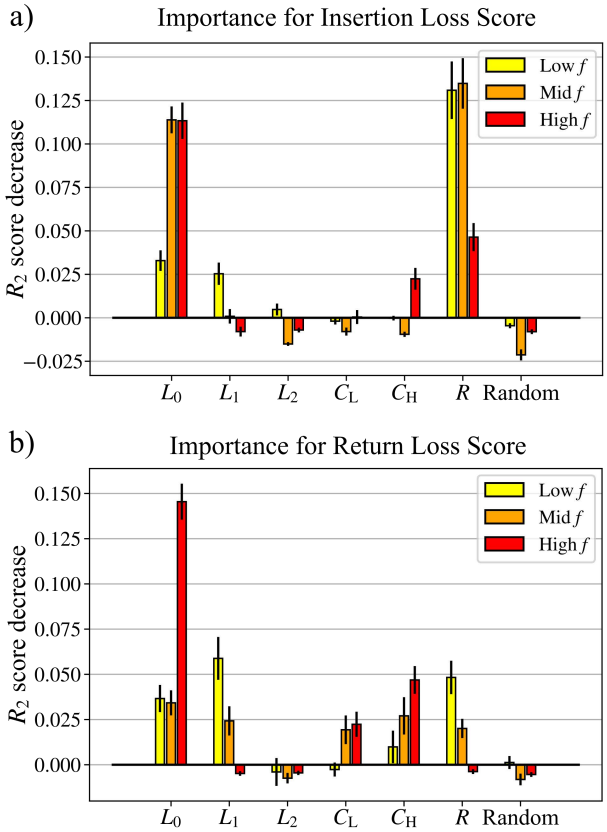

Figure 10: Feature importance calculated using the BO surrogate model. a) shows the feature importance on the insertion loss scores and b) shows the feature importance for the return loss scores.

B. *Surrogate model utilization*

More advanced surrogate models such as BNNs [13] (see section II.A), offer higher flexibility for the input and output data formats in optimization settings. Besides the option to include higher dimensional inputs, e.g. image data, as auxiliary information, BNNs can directly be used to predict a vectorial output format. For the high-speed design introduced in section III, the surrogate model could for example predict resulting S-parameter spectra.

In terms of sampling efficiency, benchmark experiments presented in [13] suggest that BNNs perform comparable to GP based methods when the auxiliary information is applied at the output. However, the more informative model outputs of the BNNs have the advantage that they make the acquisition process more explainable. Additionally, the designer obtains a model that allows a more in-depth exploration of the design space.

Figure 11 shows predictions of a BNN used as surrogate model for BO applied on a low-fidelity version of the design task. In this low-fidelity setting, we used three choices for inductive devices as inputs. The output is calculated by cascading these devices to a bias-tee without a layout or other components considered (LF2, see chapter III.E). The figure shows BNN predictions for the S_{11} spectrum for a validation sample. The predictions are recorded during the BO run, in which the surrogate model is trained with an increasing number of observations n.

The figure shows, that the model-fit on the validation sample improves with increasing number of observations

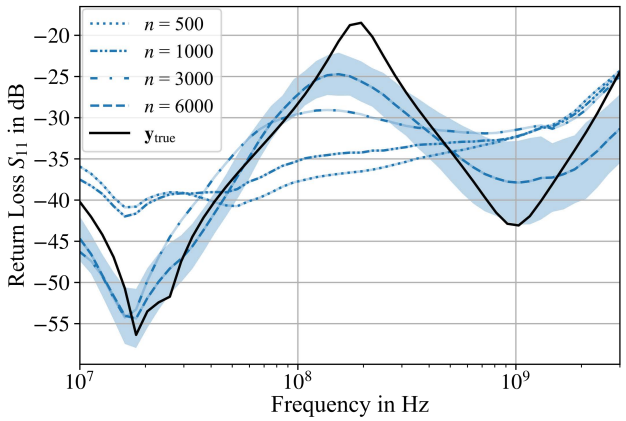

Figure 11: BNN predictions for a S-parameter spectrum over varying number of BO observations n. The black curve shows a S_{11} vector of a validation sample. The blue curves show the predicted mean of the BNN trained on different training data sizes n. For $n = 6000$ the shaded area indicates the standard deviation of the prediction.

(which is also observed in the general validation score on a larger validation dataset). For small numbers of observations (at n = 500, and n = 1000) the BNN cannot predict resonance points in the spectrum. However, after n = 6000 observed samples, the overall fit improves significantly, and the resonance frequencies can be derived from the prediction.

For this experiment, we used an "ensemble" BNN model suggested by [13] in the default settings. The model fit could be further improved by using other BNN configurations, e.g. with a higher number of parameters, other BNN types or convolutional layers applied.

Figure 12 shows a runtime comparison using a BNN surrogate model vs. a GP surrogate model for BO. The continuous curves show the average stepwise runtime of the methods, over the number of observations n. The shaded regions show raw runtime measurements.

For the GP, we observe a noisy, nonlinear increasing runtime, while for the BNN the increase is linear (see section II.A). The shaded curve of the BNN has a large margin, which is caused by a complete retraining of the model at every 10th observation (see [13]). By changing this retraining frequency and the hyperparameters of the BNN itself, the gradient of the average runtime over n can be modified.

The experiment (with default settings applied to the frameworks) shows that after 1250 observations, the training

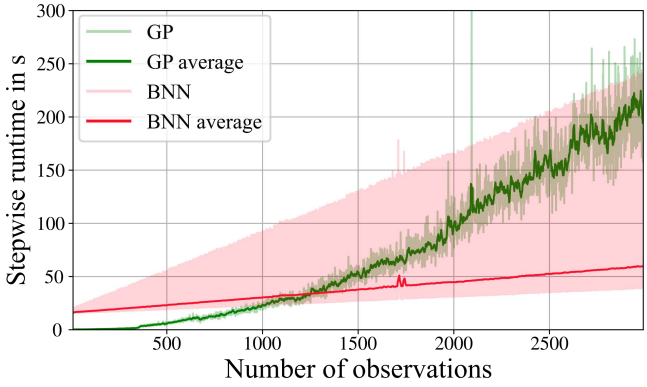

Figure 12: Comparison of the stepwise BO runtime between a GP and a BNN surrogate model.

of the BNN has a shorter runtime than the GP training. The cumulative runtime of the GP based method surpasses the BNN after 2000 observations, from where on the use of a BNN would be preferable.

The experiment in Figure 11 shows, that a large quantity of observations is required to get a sufficient fit of the surrogate model to predict a vectorial output. This, combined with the runtime behavior shown in Figure 12, makes the BNN-based methods preferable for optimization experiments with budgets above 1000 observations.

For optimization experiments with smaller budgets the GP-based methods are preferable due to the faster runtime in this region and less hyperparameters to choose for the surrogate model. However, with increasing computational resources or cluster-based evaluation options for other optimization tasks, BNNs will be a preferable option as a surrogate model due to their greater flexibility.

C. Transfer learning capability

A surrogate model-based approach enables transfer learning (TL) for optimization. This can further improve the sampling efficiency in BO as suggested in numerous studies [38, 39, 40].

In our experiments TL would enable to utilize low-fidelity data as prior information for high-fidelity optimization. This approach is similar to multi-fidelity optimization approaches such as [41, 42]. However, these multi-fidelity methods are often not applicable to EDA tasks because they require a continuously tunable fidelity parameter. EDA settings often allow only few fidelity options, e.g. using 2.5D or 3D simulations. For such problems, a TL approach using previously collected data of varying fidelity would be preferable.

Figure 13 shows an experiment, where we evaluate a simple TL approach of providing a GP surrogate model with a prior mean, which is derived from observations sampled in a previous experiment. This approach is comparable to the MHGP method in [40]. The known data was sampled in a previous experiment from the so-called *source* function, while the unknown black-box function that is to be optimized is called the *target* function.

For the experiment in Figure 13 we use both low-fidelity benchmark versions of the high-speed design, which are introduced in section III.E. Low-fidelity function LF2, which completely neglects any layout influences, acts as source (I.a). The target function (I.c) is the low-fidelity function LF1, where the layout remains fixed for all inductor combinations. Thereby, we obtain two similar functions for benchmark experiments, where we can compare the TL approach to the best performance shown in the previous benchmarks (see Figure 7).

In the TL experiment we assume that the source function is completely known and generated a so-called "mean mask" (I.b) based on it. The mask elevates values that are larger than the source-functions mean value (black-regions), compared to values smaller than the mean (white-regions). We then applied this mask as a prior mean to the GP model used in the BO method. This prior should guide the BO-acquisitions on the target function to the lower-than-mean regions indicated by the mask.

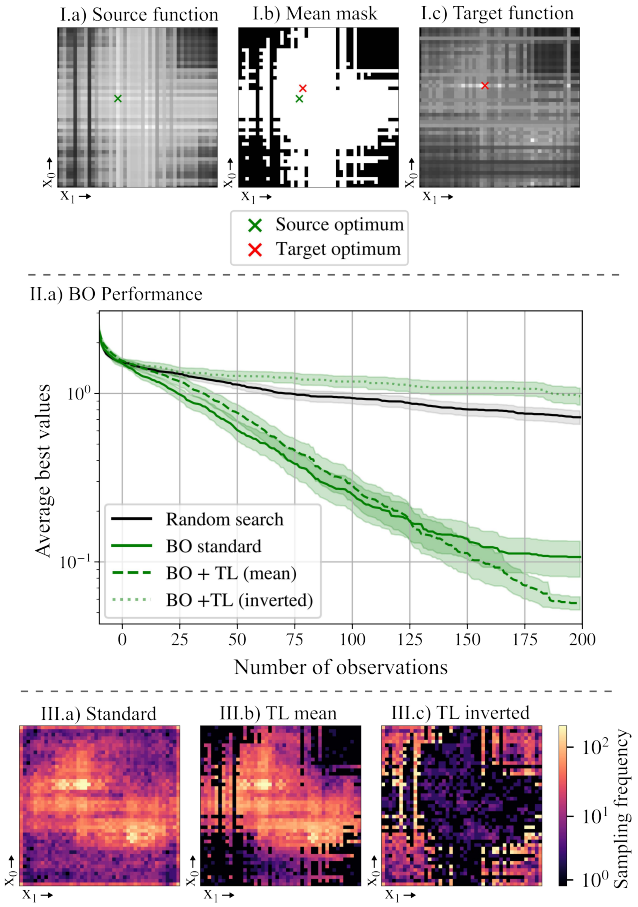

Figure 13: Transfer Learning experiment. I) depicts an exemplary source function, a mask based on the source function and the target function. II) shows the optimization performance of two different TL-masks applied as a BO prior mean in comparison to the standard performance. III) shows the heatmap of chosen samples by the three approaches indicating the effect of the masks used as prior mean. The color-scale indicates how often a certain sample was chosen in the benchmark.

Figure 13 II.a) shows the BO performance with the TL applied compared to the standard approach without TL. We also tested an inverted mean mask to demonstrate the general influence of the masking. As expected, the inverted mask deteriorates the BO performance substantially, as it masks out the optimal regions of the target function. The mean mask performs comparable to the standard approach; however, it outperforms the standard approach beyond 150 observations, finding the global optimum more frequently than the standard approach. This can be caused by the reduced need for exploration in the TL approach, which could lead to more significant performance improvements in higher dimensions.

Figure 13 III) shows heatmaps of the analyzed approaches, with the brighter colors indicating more frequent choices by the BO algorithm of certain samples in the benchmarks. The heatmaps clearly show the impact of the masks applied to the prior mean. III.b) shows that with the application of the mean-mask the exploration in the corner and edge regions is significantly reduced compared to the standard approach (III.a).

This experiment demonstrates the effectiveness of the TL approach, outlining future opportunities to further enhance the sampling efficiency of BO.

VI. Conclusions

In this paper, we presented various benefits of applying state-of-the-art Bayesian Optimization methods to costly simulation tasks in the field of high-speed design. This not only illustrates the possibility of accelerating PCB design tasks using BO as a sampling efficient optimization method, but also indicates additional possibilities offered by the surrogate model-based approach.

First, we evaluated the often-cited sampling efficiency of BO in comparison to other common optimization methods in benchmark experiments. In the conducted experiments, we demonstrate the superior optimization performance of BO on different functions, as well as the impact of categorical variable encodings. In addition, we emphasize that for single optimization runs, it is not guaranteed that BO outperforms other methods due to the stochastic nature of the algorithms. However, BO *increases the probability* of achieving a superior performance in single optimization trials.

Using the low-fidelity benchmark function derived from the high-fidelity filter design task, we can show that BO outperforms the other optimization methods on average by a factor of up to 3 and the random search baseline by a factor of 5. Based on these results, we subsequently applied the best optimization method to the high-fidelity design. Using the fully automated design process, we found an optimal design for the bias tee circuit after 450 evaluations.

Besides the high sampling efficiency of BO, we demonstrate additional advantages of surrogate model-based optimization methods. On the one hand, the surrogate model can further be utilized to extract feature importance metrics and design guidelines. On the other hand, more advanced surrogate models, such as BNNs, make it possible to directly predict auxiliary information, e.g. spectral data. This enables further exploration of design spaces and makes acquisition decisions explainable. We illustrate that these advanced surrogate models are especially preferable in optimization settings with budgets above 1000 evaluations. The last experiment demonstrates that model-based optimization methods enable transfer learning, which can further increase the sampling efficiency of BO methods. Together with the other benefits presented, this provides insight into future directions in the application of SMBO methods.

References

[1] F. Hutter, Automated machine learning : methods, systems, challenges, Cham, Switzerland: Springer, 2019.

[2] F. Häse, M. Aldeghi, R. J. Hickman, L. M. Roch and A. Aspuru-Guzik, "Gryffin: An algorithm for Bayesian optimization of categorical variables informed by expert knowledge," *Applied Physics Reviews*, vol. 8, no. 3, 2021.

[3] R. Gómez-Bombarelli, J. N. Wei, D. Duvenaud, J. M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. D. Hirzel, R. P. Adams and A. Aspuru-Guzik, "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules," *ACS Cent. Sci.*, vol. 4, no. 2, p. 268–276, January 2018.

[4] P. Chen, B. Merrick and T. Brazil, "Bayesian Optimization for Broadband High-Efficiency Power Amplifier Designs," *IEEE Transactions on Microwave Theory and Techniques*, vol. 63, p. 4263–4272, 2015.

[5] H. Torun, M. Swaminathan, A. Davis and M. Bellaredj, "A Global Bayesian Optimization Algorithm and Its Application to Integrated System Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, p. 792–802, 2018.

[6] Y. Ma, Z. Yu and B. Yu, "CAD Tool Design Space Exploration via Bayesian Optimization," *2019 ACM/IEEE 1st Workshop on Machine Learning for CAD (MLCAD)*, p. 1–6, 2019.

[7] R. Turner, D. Eriksson, M. McCourt, J. Kiili, E. Laaksonen, Z. Xu and I. Guyon, "Bayesian Optimization is Superior to Random Search forMachine Learning Hyperparameter Tuning:Analysis of the Black-Box Optimization Challenge 2020," *Proceedings of Machine Learning Research*, vol. 133, 2021.

[8] S. Greenhill, S. Rana, S. Gupta, P. Vellanki and S. Venkatesh, "Bayesian Optimization for Adaptive Experimental Design: A Review," *IEEE Access*, vol. 8, pp. 13937-13948, 2020.

[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148-175, 2016.

[10] R. Garnett, Bayesian Optimization, Cambridge University Press, 2023.

[11] C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2005.

[12] F. Häse, L. M. Roch, C. Kreisbeck and A. Aspuru-Guzik, "Phoenics: A Bayesian Optimizer for Chemistry," *ACS Central Science*, vol. 4, p. 1134–1145, September 2018.

[13] S. Kim, P. Y. Lu, C. Loh, J. Smith, J. Snoek and M. Soljačić, "Deep Learning for Bayesian Optimization of Scientific Problems with High-Dimensional Structure," *Transactions on Machine Learning Research (TMLR)*, 2022.

[14] P. Orbanz and Y. W. Teh, "Bayesian Nonparametric Models," *Encyclopedia of machine learning*, vol. 1, 2010.

[15] M. Hoffman, E. Brochu and N. De Freitas, "Portfolio allocation for Bayesian optimization," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI'11)*, Arlington, 2011.

[16] D. R. Jones, M. Schonlau and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, p. 455–492, 1998.

[17] S. Diwale, M. K. Eisner, C. Carpenter, W. Sun, G. C. Rutledge and R. D. Braatz, "Bayesian optimization for material discovery processes with noise," *Molecular Systems Design & Engineering*, vol. 7, p. 622–636, 2022.

[18] Q. Liang, A. E. Gongora, Z. Ren, A. Tiihonen, Z. Liu, S. Sun, J. R. Deneault, D. Bash, F. Mekki-Berrada, S. A. Khan, K. Hippalgaonkar, B. Maruyama, K. A. Brown, I. I. I. John Fisher and T. Buonassisi, "Benchmarking the performance of Bayesian optimization across multiple experimental materials science domains," *npj Computational Materials*, vol. 7, November 2021.

[19] Y. Zhang, D. W. Apley and W. Chen, "Bayesian Optimization for Materials Design with Mixed Quantitative and Qualitative Variables," *Scientific Reports*, vol. 10, March 2020.

[20] A. Yazdanbakhsh, C. Angermueller, B. Akin, Y. Zhou, A. Jones, M. Hashemi, K. Swersky, S. Chatterjee, R. Narayanaswami and J. Laudon, "Apollo: Transferable Architecture Exploration," in *Workshop on ML for Systems at the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[21] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng and D. Zhou, "An Efficient Bayesian Optimization Approach for Automated Optimization of Analog Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, p. 1954–1967, June 2018.

[22] S. A. Abdelaal, A. Hussein and H. Mostafa, "A Bayesian Optimization Framework for Analog Circuits Optimization," *15th International Conference on Computer Engineering and Systems (ICCES)*, December 2020.

[23] B. He, S. Zhang, F. Yang, C. Yan, D. Zhou and X. Zeng, "An Efficient Bayesian Optimization Approach for Analog Circuit Synthesis via Sparse Gaussian Process Modeling," *IEEE Design, Automation and Test in Europe Conference (DATE)*, March 2020.

[24] J. Huang, S. Zhang, C. Tao, F. Yang, C. Yan, D. Zhou and X. Zeng, "Bayesian Optimization Approach for Analog Circuit Design Using Multi-Task Gaussian Process," *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021.

[25] R. Medico, D. Spina, D. Vande Ginste, D. Deschrijver and T. Dhaene, "Machine-Learning-Based Error Detection and Design Optimization in Signal Integrity Applications," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 9, p. 1712–1720, September 2019.

[26] R. S. Rezende, M. Hadžiefendić, J. Hansen and R. Schuhmann, "Multi-Output Variable-Fidelity Bayesian Optimization of a Common Mode Choke," in *IEEE International Joint EMC/SI/PI and EMC Europe Symposium*, 2021.

[27] Maxim Integrated, "Gigabit Multimedia Serial Links for ADAS," [Online]. Available: https://www.maximintegrated.com/.../gigabit-multimedia-serial-links-for-adas.pdf.

[28] J. Krummenauer, N. Kammoun, B. Stein and J. Goetze, "Impact of Categorical Variable Encodings on Bayesian Optimization Performance," *IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2022.

[29] J. Krummenauer, N. Kammoun, B. Stein and J. Goetze, "Encoding categorical variables in physics-informed graphs for Bayesian Optimization," *IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, 2022.

[30] Ansys, "PyAEDT," May 2023. [Online]. Available: https://aedt.docs.pyansys.com.

[31] A. Arsenovic, "scikit-rf: An Open Source Python Package for Microwave Network Creation, Analysis, and Calibration," *IEEE Microwave Magazine*, vol. 23, p. 98–105, 2022.

[32] Gpyopt-Team, *GPyOpt: A Bayesian Optimization framework in Python*, 2016.

[33] J. Blank and K. Deb, "Pymoo: Multi-Objective Optimization in Python," *IEEE Access*, vol. 8, p. 89497–89509, 2020.

[34] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, p. 67–82, 1997.

[35] C. Molnar, Interpretable Machine Learning, 2 ed., 2022.

[36] G. Louppe, L. Wehenkel, A. Sutera and P. Geurts, "Understanding variable importances in forests of randomized trees," *Advances in neural information processing systems*, 2013.

[37] E. Romero and J. M. Sopena, "Performing Feature Selection With Multilayer Perceptrons," *IEEE Transactions on Neural Networks*, vol. 19, p. 431–441, March 2008.

[38] Y. Li, Y. Shen, H. Jiang, W. Zhang, Z. Yang, C. Zhang and B. Cui, "TransBO: Hyperparameter Optimization via Two-Phase Transfer Learning," *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, p. 956–966, August 2022.

[39] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro and D. Sculley, "Google Vizier: A Service for Black-Box Optimization," *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, p. 1487–1495, 2017.

[40] P. Tighineanu, K. Skubch, P. Baireuther, A. Reiss, F. Berkenkamp and J. Vinogradska, "Transfer Learning with Gaussian Processes for Bayesian Optimization," *25th International Conference on Artificial Intelligence and Statistics*, 2022.

[41] J. Bergstra, D. Yamins and D. Cox, "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," *30th International Conference on Machine Learning (ICML 2013)*, June 2013.

[42] S. Falkner, A. Klein and F. Hutter, "BOHB: Robust and Efficient Hyperparameter Optimization at Scale," *35th International Conference on Machine Learning (ICML 2018)*, 2018.

[43] M. Cairnie and C. DiMarino, "Bayesian Optimization of PCB-Integrated Field Grading for a High-Density 10 kV SiC Power Module Interface," *IEEE Transactions on Power Electronics*, vol. 37, p. 7590–7603, July 2022.

[44] Z. Kiguradze, J. He, B. Mutnury, A. Chada and J. Drewniak, "Bayesian Optimization for Stack-up Design," *IEEE International Symposium on Electromagnetic Compatibility, Signal and Power Integrity (EMC+SIPI)*, July 2019.