electronics

Article

Mixed-Variable Bayesian Optimization for Analog Circuit Sizing through Device Representation Learning

Kostas Touloupas * and Paul Peter Sotiriadis

Department of Electrical and Computer Engineering, National Technical University of Athens, 15780 Athens, Greece
* Correspondence: ktouloupas@mail.ntua.gr

Abstract: In this work, a deep representation learning method is proposed to build continuous-valued representations of individual integrated circuit (IC) devices. These representations are used to render mixed-variable analog circuit sizing problems as continuous ones and to apply a low-budget black box Bayesian optimization (BO) variant to solve them. By transforming the initial search spaces into continuous-valued ones, the BO's Gaussian process models (GPs), which typically operate on real-valued spaces, can be used to guide the optimization search towards the global optimum. The proposed Device Representation Learning approach involves using device simulation data and training a composite model of a Variational Autoencoder (VAE) and a dense Neural Network. The latent variables of the trained VAE model serve as the representations of the integrated device and replace the discrete-valued parametrizations of particular devices. A thorough explanation of the proposed methodology's mathematical formulation is given and example sizing applications on real-world analog circuits and integrated devices underline its efficiency.

Keywords: analog circuit sizing; optimization; Bayesian; representation learning; variational autoencoder

check for updates

Citation: Touloupas, K.; Sotiriadis, P.P. Mixed-Variable Bayesian Optimization for Analog Circuit Sizing through Device Representation Learning. *Electronics* **2022**, *11*, 3127. https://doi.org/10.3390/electronics11193127

Academic Editors: Fábio Passos, Nuno Lourenço and Ricardo Martins

Received: 31 August 2022
Accepted: 23 September 2022
Published: 29 September 2022

1. Introduction

The economic and social impact of the device miniaturization in Integrated Circuits (ICs), described by Gordon Moore in his famous heuristic law [1], is undeniable. Individuals harness the benefits of an interconnected world, where modern power efficient computational and communication ICs enable unprecedented capabilities and yield considerable profits for investors. These benefits, however, come at the cost of increased design times and manufacturing difficulties [2]. In practice, successful IC design requires well studied, established procedures which include, among others, the time consuming extensive verification of circuits. Often, the need to decrease the time-to-market leads to inefficient and error prone designs [2].

Device miniaturization, in reality, has had more impact on the work of analog designers, compared to digital ones [2]. The main reason for this issue is the nature of the circuits themselves; digital circuitry that operates on signals having distinct levels is more easy to abstract and reason about, in comparison to analog designs that requires expert skills to take into account the physics of each fabrication process [3]. In addition, the sizing of individual devices in analog ICs involves selecting values from a continuous range, rendering the design space virtually infinite. The resulting analog performance metrics, on the other hand, are closely correlated to both the device sizes and low-level circuit aspects, such as voltages and currents, making it difficult to establish abstractions and to determine a good compromise between trade-offs [4]. The aforementioned problems, and the resulting productivity gap in analog design [3], can be addressed by means of dedicated automation tools that facilitate the job of analog designers.

Efforts to establish means of automation in the analog design cycle include mostly sizing automation approaches [5–10]. In this setting, the sizing procedure of a given analog circuit topology is formulated as a black-box optimization problem, where the design space includes all the unknown design variables and the optimization goals and constraints rely on the desired performance metrics. To evaluate potential solutions, these approaches use a simulation-in-the-loop method where parametrized testbenches are updated at each optimization step and they are simulated using commercial simulators in order to attain their goodness, i.e., their deviation from the desired performance metrics. The optimization algorithm is the main choice for such automation tools. Typically, Evolutionary Algorithms (EAs) [11] are used to derive optimal sizing solutions. However, EAs are population-based methods that produce good approximations for the global optima when a large number of evaluations have taken place. A low simulation budget alternative is the Bayesian optimization (BO) algorithm [12], which derives good approximations of the optima with relatively fewer evaluations than EAs [12]. BO has been applied to analog circuit sizing, both in the multi-objective [5] and in the single-objective settings [8].

BO is an iterative surrogate modelling procedure that uses Gaussian processes (GPs) [13] to model the objective and constraint functions of the optimization problem. GPs, however, operate only on continuous spaces and, subsequently, hinder the application of BO in problems that include discrete variables. In practice, many real-world circuits include devices, such as integrated inductors, which are parametrized by both discrete and continuous variables. A partial remedy to this situation is allowing GPs to operate on a continuous space and having BO propose promising (query) points that are continuous [5]. These query points can be altered prior to the simulation, by rounding the discrete-valued variables to the closest integers. This relaxation approach, however, may lead to systematic biasing towards some discrete values on top of others. An alternative approach, where no brute-force rounding takes place is much needed.

Recently, the topic of latent space optimization [14–17] has emerged in the field of Machine Learning (ML) as a promising approach to solving optimization problems that are hard to formulate, either because of the large dimensionality of the search space or due to the representation of the input one. This approach involves a latent variable generative model [18] $G : \mathbb{Z} \rightarrow \mathbb{X}$ that maps vectors from a space $\mathbb{Z}$ to the actual search space of the optimization $\mathbb{X}$. By choosing $\mathbb{Z}$ to be of low dimensionality and continuous and using it as the search space, one can map the initial optimization problem into a new, low-dimensional, and continuous one. Motivated by the above, in this paper we present an approach for learning continuous, data-driven representations of integrated devices as latent codes and we use them to render mixed-variable sizing problems continuous. The resulting continuous optimization problem is then solved using a BO variant.

We use a variational autoencoder (VAE) [19] as the generative model for learning the device representations and, to ensure that the latent space is well structured, we embed a label-guiding supervised predictor network that maps latent codes to device geometric sizes. Instead of training the generative network to map geometric sizes to latent codes, we embed domain-specific simulation data to $\mathbb{Z}$. The latent codes are used as inputs in the predictor net, which is trained simultaneously with the VAE to predict device sizes. During optimization, the search is conducted on the latent space, and the geometric features are acquired by using the predictor model that maps latent codes to actual geometries.

The paper is organized as follows. Section 2 explains the simulation-based optimization framework for analog circuit sizing employed in this work. In Section 3, the employed BO algorithm is discussed. Section 4 presents theoretical background on latent space optimization and the VAE and demonstrates the proposed approach. In Section 5, the proposed approach is applied for learning representations of integrated inductors and capacitors. Finally, Section 6 presents two sizing examples using two LNA topologies and Section 7 concludes the paper.

2. Automatic Sizing Framework

In this section, the framework used for automatic sizing and optimization of analog and RF circuits is explained and its advantages over other approaches are discussed.

Automating the procedure of analog circuit sizing, while at the same time optimizing the circuit's performance is a subject that has been addressed by both heuristic rules and optimization approaches. However, the latter have demonstrated increased generalization and seem to be a more viable solution for analog Electronic Design Automation (EDA) tools. These methods cast the sizing procedure as a minimization problem, where the devices of the circuit under test are updated until some imposed specifications are met. They are composed of two basic ingredients: an optimization algorithm and an evaluation engine. As shown in Figure 1, the core of this approach is an iterative procedure, which repeats the steps of searching for potentially promising solutions, using optimization algorithms, and evaluating them. Depending on the choice of evaluation engine, these methods are distinguished between simulation-based [20] and model-based ones [21].

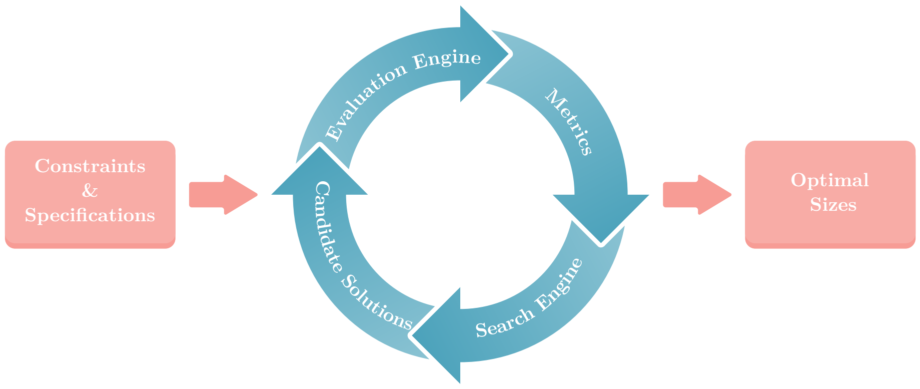


Figure 1. Optimization-based flow of analog circuit sizing.

The model-based methods evaluate the goodness of each parametrization of the circuit using analytical equations, which are derived by hand analysis. A particular example of this family of methods is the transconductance-over-current g_m/I_d methodology [22], where transistors' behavior is implicitly taken into account by replacing the overdrive voltage with the technology-independent g_m/I_d metric. By building analytical models of the circuit's performance using the g_m/I_d as design variable, one may define optimization problems that result in sized circuits, such as in [21]. A limitation of the g_m/I_d is that, being a small-signal metric that describes the inversion level of each transistor, it does not give insights into the large signal behavior of the circuit. In addition, the formulation of the design equations using g_m/I_d as a free-parameter may require a substantial effort from the designer and even become intractable when the variable space is large, which is contradictory with the effort of automating the sizing process.

Instead, in this work, we use off-the-shelf commercial simulators as evaluation engines for the optimization procedure. We developed a high-level Python framework that automates the process of simulation triggering, result parsing and metric calculation, using the Cadence Spectre simulator. The designer is able to define constraints and optimization goals for the simulator outputs and the simulation takes place at each loop of the iterative procedure of Figure 1. This black-box approach is applicable to any type of circuit and requires only the existence of parametrized testbenches, without need for analytical derivation of the circuit's equations. It is worth noting that the designer should impose any constraint necessary for the circuit to be functional, in order for the optimization algorithm to produce reasonable results.

3. Employed Bayesian Optimization

In this section, the functionality of BO, along with its GP surrogate models are discussed.

3.1. Gaussian Processes

Gaussian processes (GPs) [13] belong to the family of non-parametric models. They build a distribution over the function space and perform Bayesian inference to approximate an unknown black-box function. GPs are highly studied topic with their popularity stemming from the fact that (1) they rely on few *hyperparameters* to approximate the given dataset, which are determined in a systematic manner, (2) they yield uncertainty estimates about their predictions, and (3) they use closed-form expressions for all operations required to perform inference.

Let us now discuss GPs in mathematical terms. Consider a dataset $\mathcal{D}$ comprised of n d-dimensional parameter vectors $X = \{\mathbf{x}_i\}_{i=1}^n$ and a corresponding 1-dimensional observation value set $\mathbf{y} = \{y_i\}_{i=1}^n$, derived by measuring an unknown black-box function $f : \mathbb{R}^d \to \mathbb{R}$. In the general case, we assume that the observations are corrupted by an uncorrelated additive noise source such that it holds

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, n, \tag{1}$$

where ϵ_i is the noise in the i-th measurement. In the GP context, we say that the actual function f in Equation (1) is modeled by a GP $\tilde{f}$, such that

$$y_i = \tilde{f}(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, n. \tag{2}$$

Here, the random noise is considered to follow a zero mean Gaussian distribution, such that $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, where σ is the standard deviation.

The values of $\tilde{f}$ at any point $\mathbf{x}^\star \in \mathbb{R}^d$ are 1D random variables that follow Gaussian distributions. In practice, the distribution $\tilde{f}(\mathbf{x}^\star)$ serves as an uncertainty estimate for the GP model's prediction. The aforementioned property leads to the conclusion that a GP can be considered as a stochastic process of infinitely many random variables, which together form a probability distribution over function f [13]. For every positive integer n, any $(n \times 1)$ vector $\mathbf{f} = [\tilde{f}(\mathbf{x}_i)]_{i=1}^n$, which is comprised out of a subset of these random variables, follows a multivariate Gaussian distribution [13], i.e.,

$$\mathbf{f} = [\tilde{f}(\mathbf{x}_1,) \ldots, \tilde{f}(\mathbf{x}_n)]^T \sim \mathcal{N}(\boldsymbol{\mu}, K). \tag{3}$$

Vector $\boldsymbol{\mu}$ is the GP mean, defined by a mean function $m : \mathbb{R}^d \to \mathbb{R}$, and K is the covariance matrix, constructed by a kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ [18], such that $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Functions $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ define uniquely the GP model and constitute design choices. The mean function imposes a bias on the function values and it can be used to describe our prior beliefs about the unknown black-box function. Often, in cases when no prior information about f is available, the mean function is set to a constant μ. The kernel function is a measure of similarity between the GP outputs of any two input points and determines the model's behavior, such as periodicity, smoothness, etc. [23].

Popular kernel functions include the squared exponential and Matèrn kernel families [13]. The Matèrn 5/2 kernel is employed in this work, since it relaxes the smoothness of the exponential ones. Its function is given by

$$k_{\nu=5/2}(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \left(1 + \sqrt{5}r + \frac{5}{3}r^2\right) e^{-\sqrt{5}r}, \tag{4}$$

where r is given by

$$r = \left(\sum_{k=1}^{d} \frac{(x_{i,k} - x_{j,k})^2}{\lambda_k^2}\right)^{1/2}. \tag{5}$$

Here, the parameters λ_k are the length-scales of the kernel and correspond to the degree of variation of the GP with respect to the dimensions of the input point [13].

Having established the kernel and mean functions of the GP model, one must adapt the model to the observed dataset $\mathcal{D}$. This involves learning the kernel's hyperparameters, i.e., length-scales, variance, which are grouped into a vector $\boldsymbol{\theta}$. To do so, one needs to maximize the marginal likelihood of the observations in $\mathcal{D}$, i.e., determine the values of the hyperparameters, such that the observations $\mathbf{y}$ become likely, under the GP. Arranging the noise-corrupted measurements $\mathbf{y}$ in a vector, and taking into account the additive Gaussian noise in Equation (2), this vector follows a Gaussian distribution with mean and covariance

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, K + \sigma^2\mathbb{I}). \tag{6}$$

The learning process for a GP consists of minimizing the negative log-marginal likelihood [12],

$$L(\boldsymbol{\theta}) = \frac{1}{2}(\mathbf{y} - \boldsymbol{\mu})^T(K + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \boldsymbol{\mu}) + \frac{1}{2}\log(|K + \sigma^2\mathbb{I}|) + \frac{n}{2}\log(2\pi). \tag{7}$$

This expression is used to learn parameters $\boldsymbol{\theta}$, using off-the-shelf optimizers, such as the limited-memory BFGS [24].

After adapting the GP model to the data, one can make predictions about a point $\mathbf{x}^\star$ that does not belong in $\mathcal{D}$. This is achieved by utilizing the GP's predictive distribution, which is defined by

$$\begin{aligned} \mu_{\tilde{f}|\mathbf{y}}(\mathbf{x}^\star) &= \boldsymbol{\mu} + \mathbf{k}^{\mathrm{T}}(K + \sigma^2\mathbb{I})^{-1}(\mathbf{y} - \boldsymbol{\mu}) \\ \sigma^2_{\tilde{f}|\mathbf{y}}(\mathbf{x}^\star) &= k(\mathbf{x}^\star, \mathbf{x}^\star) - \mathbf{k}^{\mathrm{T}}(K + \sigma^2\mathbb{I})^{-1}\mathbf{k} \end{aligned}. \tag{8}$$

Here, $\mathbf{k}^{\mathrm{T}}$ is a $(1 \times n)$ vector with values $k(\mathbf{x}_i, \mathbf{x}^\star)$ for $i = 1, \ldots, n$ and K is the $n \times n$ kernelvmatrix.

By using the Gaussian distribution defined in Equation (8), one can produce 1D distributions which serve as predictions, for pointwise inputs $\mathbf{x}^\star$. It is important to note that, although the predictive distribution in Equation (8) is given for the pointwise input $\mathbf{x}^\star$, one can define and use a multivariate predictive distribution as well. Considering a set of $k > 1$ unseen points $\mathbf{x}_1, \ldots, \mathbf{x}_k$, the predictive distribution $p\big(\big[\tilde{f}(\mathbf{x}_1), \ldots, \tilde{f}(\mathbf{x}_k)\big)\big]\big)$ has a covariance matrix

$$\mathrm{Cov}(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j) - \mathbf{k}^{\mathrm{T}}_{\mathbf{X},\mathbf{x}_i}(K + \sigma^2\mathbb{I})^{-1}\mathbf{k}_{\mathbf{X},\mathbf{x}_j}, \tag{9}$$

where $1 \le i < j \le k$ and the $(1 \times n)$ vector $\mathbf{k}^{\mathrm{T}}_{\mathbf{X},\mathbf{x}_j} = \big[k(\mathbf{x}_i, \mathbf{x}_j)\big]_{i=1}^{n}$, while the mean vector remains the same as in Equation (8). Sampling from this joint predictive distribution results in a k-dimensional output vector $\tilde{\mathbf{f}}$.

3.2. Bayesian Optimization

Bayesian optimization (BO) [12] is a sample efficient method to solve global optimization problems, particularly aiming expensive-to-evaluate cost functions. In the unconstrained regime, a real valued, unknown function f is provided, and BO learns a fast-to-evaluate surrogate model from past evaluations. It selects next query points for evaluation sequentially, by balancing exploration and exploitation to find the global optimum. The BO framework consists of two main components; the probabilistic surrogate model that aims to approximate f and an *acquisition function* $a : \mathbb{S} \to \mathbb{R}$ [12] that provides a score of utility for evaluating a candidate query point, based on the probabilistic model. In most approaches, the surrogate model is a GP, which is trained and used for predictive point evaluations as discussed in the previous subsection.

BO works in iterations and its pseudocode is given in Algorithm 1. Starting from an initial set of evaluations, BO incrementally builds a GP model based on historical data, and selects a new query point as the one that optimizes the acquisition function. This is an auxiliary optimization problem, but since the acquisition function is fast to evaluate, off-the-shelf optimization methods, such as CMA-ES [12] and BFGS [24], can be used. After the selection of the query point $\mathbf{x}^\star$, the black-box function is evaluated to acquire the corresponding measurement value $y^\star$. The data pair $\{\mathbf{x}^\star, y^\star\}$ is appended to the historical data and BO proceeds with training the models once again.

Acquisition functions take as inputs the trained GP model and produce estimates about the goodness of each point in the search space. To do so, they rely on the predictive distributions of Equations (8) and (9) and implement a procedure to translate these distributions to a scalar-valued measure. Typically, they are designed such that their optimization yields the best candidate point for evaluation, at a particular BO iteration. Since in the context of black-box optimization there is no universally accepted 'best' way to explore the search space, there exist several acquisition functions that work in different ways. In the single-objective case, the most popular acquisition functions are the Lower Confidence Bound (LCB), the Expected Improvement (EI), and the Probability of Improvement (PI) [25]. All of these functions define the goodness of each point using the pointwise predictive distribution of the GP model in Equation (8).

Algorithm 1: BO algorithm

```
Input  : Initial samples $N_{init}$, number of iterations $T_{max}$, variable space $\mathbb{S}$
Output: Global minimum $\mathbf{x}_{best}$
Create randomly a set $\mathbf{X}$ of $N_{init}$ initial samples from $\mathbb{S}$
Evaluate $\mathbf{X}$ to acquire observations $\mathbf{y}$
for $i = 1, \ldots, T_{max}$ do
    Adjust GP models using Equation (7)
    $\mathbf{x}^\star \leftarrow \text{argmax}_{\mathbf{x}\in\mathbb{S}}\, \alpha(\mathbf{x}, \mathbf{y})$
    Evaluate $\mathbf{x}^\star$ to acquire $y^\star$
    Update archive $\mathbf{X} \leftarrow \mathbf{X} \cup \{\mathbf{x}^\star\}$, $\mathbf{y} \leftarrow \mathbf{y} \cup \{y^\star\}$
end
Find $\mathbf{x}_{best}$ from $\mathbf{X}$, $\mathbf{y}$
```

In contrast, this work utilizes the Thompson Sampling (TS) [12] acquisition function, which addresses the exploration–exploitation trade-off by drawing random samples from the GPs' posterior distributions and selecting a query vector by optimizing on these samples. The samples are drawn over a large number of input points, to provide multiple vectors of functions values, which, intuitively, can be thought as sampled functions from the GP models. However, there is no way to yield exact, analytical expressions for the samples of GP models [13]. Typically, TS samples the values of the functions at a predefined, finite-length vector of input points [10], and determines the query point by finding the best value among them. However, in the case of high-dimensional variable spaces, this quantization approach is not practical since exponentially many points need to be considered to achieve good coverage. Therefore, in this work, we use analytic approximations of the posterior samples using Random Fourier Features (RFF) [26], instead of relying on quantization. RFFs are a set of basis cosine functions $\phi(\mathbf{x}) = [\phi_i(\mathbf{x}]_{i=1}^{M}$ that approximate the GP's kernel function as $k(\mathbf{x}, \mathbf{x}') \approx \phi(\mathbf{x})^T\phi(\mathbf{x}')$ and can be also used to to approximate a sample from the GP as a linear model [26]

$$f_{sample}(\mathbf{x}) \approx \phi(\mathbf{x})^T\boldsymbol{\theta}, \tag{10}$$

where θ is an M-dimensional vector drawn from a Gaussian distribution. In this work, at each iteration, $N_s \geq 1$ approximate analytic samples are created through RFF, for each of the objective and the constraint functions. Each set of sampled constraint and objective functions is utilized in an auxiliary optimization problem, which is solved using a genetic algorithm (GA). Background on the use of RFFs in BO can be found in [5], where it was applied in the case of multiple objectives.

4. Device Representation Learning

In this section, the proposed method to derive continuous representations of integrated devices using deep VAE models is presented. For completeness, a subsection with a summary about the functionality of the employed VAE model is also given.

4.1. Variational Autoencoders

VAEs [19] are generative models, i.e., provided a dataset $\mathcal{D} = \{\mathbf{x}_i \in \mathbb{X}\}_{i=1}^{N}$ they can produce synthetic samples that follow approximately the distribution of the inputs $\{\mathbf{x}_i\}_{i=1}^{N}$. They rely on an encoder–decoder architecture and fall into the category of latent variable models [18]. These assume that, the—potentially complex—distribution of $\mathcal{D}$ which we wish to approximate, can be better explained using some *hidden* or *latent* variables $\mathcal{Z}=[\mathbf{z}_i]_{i=1}^{N}$. By defining a joint distribution over the observed variables in $\mathbf{x}$ and the unobserved latent variables $\mathbf{z}$, one can acquire the distribution of the observable ones $p_{\boldsymbol{\theta}}(\mathbf{x})$ by marginalization:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}. \tag{11}$$

Note that the distributions in the above expression stem from a flexible parametric model $p_{\boldsymbol{\theta}}$, with parameters $\boldsymbol{\theta}$. Therefore, $p_{\boldsymbol{\theta}}(\mathbf{x})$ can be expressed using simpler distributions as components. The VAE latent variable model defines the joint distribution of the observable and the latent variables as

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}). \tag{12}$$

Here, $p_{\boldsymbol{\theta}}(\mathbf{z})$ is the prior distribution and controls the behavior of the latent variables. The model's likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ describes the mapping from latent variables to observable ones. In addition, $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is termed the posterior distribution of the model.

Given the above, the generational procedure of the VAE consists of two steps [19]:

1. Sampling a latent variable vector $\mathbf{z}_i$ from the prior distribution $p(\mathbf{z})$;
2. Using the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z} = \mathbf{z}_i)$ to generate an observable vector.

The inference process of a latent variable model involves determining the latent variable value given an input data point $\mathbf{x}$, and it is formulated by the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

In practice, the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ stems from non-linear activation functions of neural nets, which makes it intractable to compute. Thus, brute maximum likelihood to train the VAE model is not possible [19]. To circumvent this issue, VAEs build a parametric approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to the true posterior using an encoder network $f_{\boldsymbol{\theta}} : \mathbb{R}^N \to \mathbb{R}^{2\times M}$. Parameters $\boldsymbol{\phi}$ are the are the weights and the biases of the network and the approximate posterior is formulated as

$$\begin{gathered}\Big(\boldsymbol{\mu}, \log(\boldsymbol{\sigma})\Big) = f_{\boldsymbol{\phi}}(\mathbf{x}),\\ q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}\Big(\mathbf{z}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma})\Big).\end{gathered} \tag{13}$$

Learning the optimal VAE parameters requires the maximization of the Evidence Lower Bound (ELBO), which is given by

$$\mathcal{L}_{VAE}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big] - \mathrm{D}_{KL}\Big(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{z})\Big), \tag{14}$$

where D_{KL} is the Kullback–Leibler (KL) divergence, which is a non-negative measure of similarity between distributions. Low KL-divergence values indicate a good resemblance between the given distributions. The maximization of Equation (14) with respect to both $\boldsymbol{\theta}, \boldsymbol{\phi}$ induces the maximization of the log-likelihood, ensuring greater reconstruction likelihood, as well as the minimization of the KL divergence between the approximate and the true posterior.

The approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is the encoder net which takes as inputs vectors from $\mathcal{D}$ and provides their latent representation whereas the likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is the decoder net, which maps latent representations back to their original form. A depiction of the encoder–decoder VAE architecture is given in Figure 2. Since neural nets are deterministic models, the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is implemented as in Equation (13), i.e., as a multivariate Gaussian distribution with diagonal covariance matrix with learned mean and standard deviation functions. Latent vectors $\mathbf{z}$ are produced by sampling from this distribution, as shown in Figure 2. Typically, the prior $p(\mathbf{z})$ is an isotropic, unit-variance multivariate Gaussian distribution with zero mean, which results in an analytic expression for the KL divergence term [19] in Equation (14) as

$$\mathrm{D}_{KL}\Big(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \parallel p_{\boldsymbol{\theta}}(\mathbf{z})\Big) = \frac{1}{2}\sum_{i=1}^{M}\Big[\mu_i^2 + \sigma_i^2 - \log(\sigma_i^2) - 1\Big], \tag{15}$$

with $\boldsymbol{\mu} = [\mu_i]_{i=1}^M$ and $\boldsymbol{\sigma} = [\sigma_i]_{i=1}^M$ being the outputs of the encoder network.

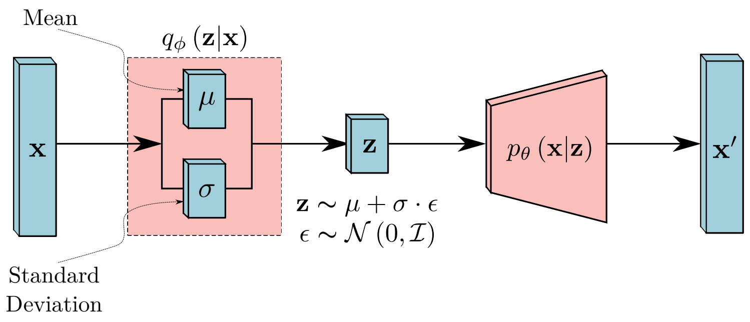

Mean
$q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{x}$
μ
σ
$\mathbf{z}$
$\mathbf{z} \sim \mu + \sigma \cdot \epsilon$
$\epsilon \sim \mathcal{N}(0, \mathcal{I})$
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Standard Deviation

Figure 2. Illustration of the VAE architecture.

4.2. Latent Space Optimization

Latent space optimization [14–17] involves the substitution of the original search space of an optimization problem, $\mathbb{S}$, with the latent space of a deep generative model, $\mathbb{Z}$. Thus, the optimization algorithm performs its search for the global optimum in the latent space and, before evaluation, the latent vectors are transformed to their original representation and fed to the simulator. This technique has been proven to be helpful in two main cases; by choosing a low latent-space dimensionality, one may mitigate the effects of the curse of dimensionality. In addition, in the case of mixed-variable input spaces, generative models can produce continuous valued latent representations, effectively rendering the original problem a continuous-valued one.

To perform latent space optimization, one must choose and adapt a generative model to the optimization search space. In our work, this is completed by the VAE model. Input vectors that reside in $\mathbb{S}$ are used as training data and the VAE represents them in $\mathbb{Z}$, by learning to reconstruct them. An optimization algorithm, such as BO, is then used to search over $\mathbb{Z}$ for optimal variables, which are then reconstructed by the decoder network and eventually fed to the simulator for evaluation.

In this setting, the structure of the VAE's latent space is of utter importance. In fact, the VAE must learn to map input vectors that yield similar outputs close-by to the latent space, such that the optimization algorithm's exploration is not deteriorated. In the case of analog circuit sizing optimization problems, learning a VAE model directly from input (device sizes) and output (performance metrics) data is not practical. This is because the model will not be able to generalize to new, unseen topologies and processes. Additionally, the process of learning such a model involves simulations that could otherwise be used to optimize the circuit in the first place. In contrast, our approach to latent space optimization involves learning multiple VAE models, each one for a particular PDK model of an integrated device. Thus, devices that are parametrized by discrete variables can be represented by continuous valued ones, under a trained VAE model. These continuous representations, then, can be used in the optimization setting, replacing the original discrete variables, and rendering the problem continuous. An illustration of the proposed procedure is shown in Figure 3.

To produce a continuous representation for an integrated device, and by taking into account that the VAE learns a manifold of the input data in such a manner that 'similar' inputs are mapped to 'close-by' latent codes, we use data that enable such a similarity comparison to take place. An i.i.d. sampling of the structured combinatorial design space of a device can be used to define similarity via the Euclidean distance, but this may not be useful in real-life cases, where a slight variation of a device's geometry yields completely different behavior. For this reason, we choose to use simulation data, tailored to each specific device. We argue that by using current-voltage characteristics or frequency responses, over some predefined ranges, functional similarity between integrated devices can be implicitly captured by the VAE model. In the following section, for example, we use frequency response data to learn a latent space representation for spiral inductors and integrated capacitors.

Since we choose to learn a representation of an integrated device using simulation data, we must incorporate the geometric characteristics of the devices in the model as well, otherwise there will be no way to embed them in an optimization procedure. To this end, besides the unsupervised VAE model, our approach also uses a predictor model $g : \mathbb{Z} \to \mathbb{R}$, which maps latent codes to actual geometries. The predictor model is trained simultaneously with the VAE and its parameters are learned so as to minimize the deviation between the actual and predicted geometries. Therefore, one can use the latent space in an optimization formulation and by passing latent code queries to the predictor, they can have valid geometries to pass to the simulator. Figure 3 shows an illustrative example of this transformation process. This approach has an additional effect on the latent space structure: it enforces an ordering of the latent codes, such that they have clear gradients with respect to geometrical characteristics.

The overall class of the proposed models for device representation learning are, therefore, composed of a VAE part and a predictor network. The simultaneous training of these parts requires the definition of a composite loss function, as well. By denoting the VAE loss in Equation (14) as $\mathcal{L}_{VAE}(\boldsymbol{\phi}, \boldsymbol{\theta})$, and introducing the supervised loss of the predictor as $\mathcal{L}_{sup}(\boldsymbol{\phi}, \boldsymbol{\lambda})$, where $\boldsymbol{\lambda}$ are the parameters of the predictor network, the overall loss function is their summation, i.e.,

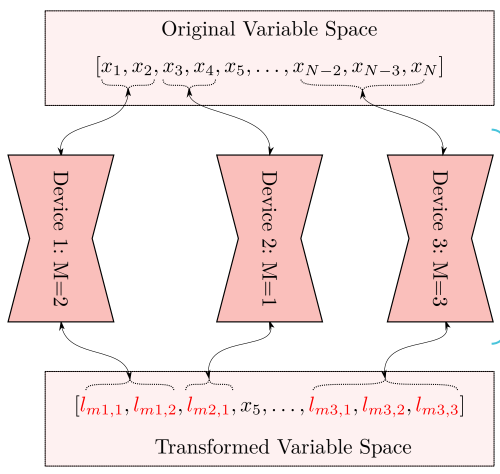
$$\mathcal{L}_{overall} = \mathcal{L}_{VAE}(\boldsymbol{\phi}, \boldsymbol{\theta}) + \mathcal{L}_{sup}(\boldsymbol{\phi}, \boldsymbol{\lambda}). \quad (16)$$

The supervised loss induces changes in the encoder network, and its actual formulation depends on the device characteristics.

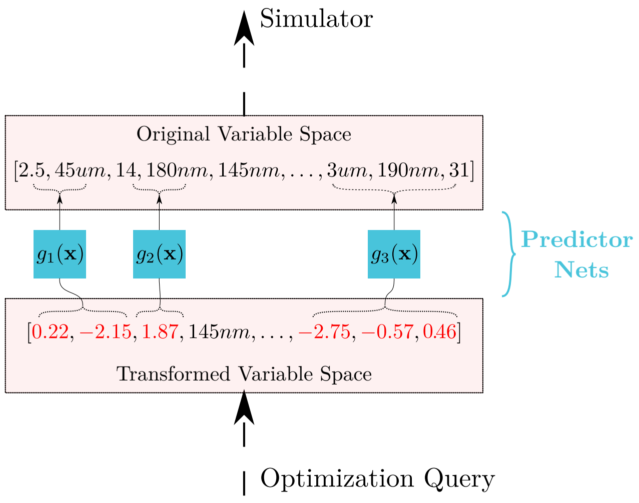
An advantage of using VAEs stems from the additional KL divergence term in Equation (14), which enforces the posterior distribution to resemble a zero mean, unit variance isotropic Gaussian one. In this case, a logical choice for latent variable ranges in an optimization setting is the hypercube $[-3, 3]^M$ with M being the dimensionality of the latent space. This translates to a 3-sigma interval, where 97.7% of all samples from a unit variance Gaussian reside, i.e., it includes almost all latent codes produced by the dataset.

By using this Deep Learning modeling approach, we are able to decouple the functional space of integrated devices from their geometric parameters. This is important because it allows us to create a parametrization where functionally similar devices are close-by in the parameter space, irrespective of the devices' granularity. It is worth noting that this representation learning technique cannot be implemented with the use of linear or other off-the-shelf shallow models alone.

In real analog sizing applications, only a subset of the original variables are discrete, or belong to the class of devices for which we learn continuous representations. Therefore, based on our approach, a new search space will be used in the optimization, where both original, continuous-valued geometric parameters and latent variables reside. Without loss of generality, let us suppose that only variable x_1 is discrete and that the rest of the variables reside in a space $\mathbb{S}$. Let us also assume that the latent variable has $M = 1$ dimension. Then, the new search space for the optimization is defined as $\mathbb{D} = \mathbb{S} \times [-3, 3]$, where $[-3, 3]$ is the space where the latent variable that substitutes x_1 resides. Of course, prior to the evaluation, the latent variable is passed through the predictor model to acquire valid input for the simulator.


Original Variable Space
$[x_1, x_2, x_3, x_4, x_5, \ldots, x_{N-2}, x_{N-3}, x_N]$
Device 1: M=2
Device 2: M=1
Device 3: M=3
Proposed Models
$[l_{m1,1}, l_{m1,2}, l_{m2,1}, x_5, \ldots, l_{m3,1}, l_{m3,2}, l_{m3,3}]$
Transformed Variable Space

(**a**)


Simulator
Original Variable Space
$[2.5, 45um, 14, 180nm, 145nm, \ldots, 3um, 190nm, 31]$
$g_1(\mathbf{x})$
$g_2(\mathbf{x})$
$g_3(\mathbf{x})$
Predictor Nets
$[0.22, -2.15, 1.87, 145nm, \ldots, -2.75, -0.57, 0.46]$
Transformed Variable Space
Optimization Query

(**b**)

Figure 3. The proposed device representation learning scheme within a sizing problem, where the latent variables are shown in red. (**a**) The original variable space of the sizing problem is changed into a transformed one, by mapping sets of variables that belong to devices into continuous latent ones. (**b**) During the optimization, a query from the optimization algorithm is transformed back to the original variable space using the predictor networks, prior to simulation.

5. Applications on Integrated Devices

In this section the application of the proposed device representation learning methodology is demonstrated on a spiral octagonal inductor and a rotative metal capacitor.

5.1. Spiral Inductor

In this subsection, we apply the continuous representation learning technique to integrated octagonal spiral inductors. We use a TSMC 90 nm process and the 'spiral std' model for 3-terminal octagonal, symmetric spiral inductors. The geometry of these devices is parametrized by 3 variables, namely inner radius, inductor width, and number of turns, whereas the spacing between inductor turns is fixed. The inner radius variable is continuous in the range [15, 90] µm, the inductor width one takes values from the set {3, 6, 9, 15} µm and the number of turns is a discrete variable with quarter-turn multiples, in the range $[0.5, 5.25]$. Although this parametrization suits the case of the PDK-provided model, it is important to state that the proposed concept applies to other inductors' models or geometric parametrizations.

In this case, we use frequency response data to train a VAE model. We consider that functional similarity between different inductors can be inferred through 1D vectors of inductance $L(f)$ and quality factor $Q(f)$, over a wide frequency range. To obtain these data, we first define a grid on which we conduct parametric sp analyses, based on the geometric variable ranges described previously. This results in 6000 inductor geometries, which are simulated on a predefined set of 250 frequencies, in the range $[0.1, 100]$ GHz. The inductors are simulated in single-ended fashion and their frequency responses are acquired by the impedance parameters as [27]

$$L(f) = \frac{\text{imag}(Z_{11})}{2\pi \cdot f}, \quad Q(f) = \frac{\text{imag}(Z_{11})}{\text{real}(Z_{11})}. \tag{17}$$

Both inductance and quality factor features for each geometry are 1D vectors with 250 entries.

The overall architecture employed for the derivation of continuous inductor representations is shown in Figure 4. The VAE part of this model is implemented using 1D convolutional filters, and the inputs to this model are tensors of size $B \times 2 \times 250$, where B is the batch size. Both the encoder and the decoder have 3 convolutional layers with kernel size 4, stride 2, and padding 1, while the filter size is shown in the Figure 4. The ReLU non-linearity is used. Two linear layers are used before and after the projection to the latent space, which has $M = 3$ dimensions. It is important to state that the convolutional architecture is preferred, in order to take advantage of the spatial correlations between the entries of the 1D vector inputs.

The predictor part of the model is a Fully Connected Neural Network (FCNN) which takes as inputs the sampled points from the approximate posterior distribution and maps them to the actual geometric characteristics. The FCNN has three layers in total, which are linear with 3, 50, and 25 neurons, and use ReLU activations; this is shown in pink colour in Figure 4. The 25 outputs of the predictor network are utilized as follows: for the number of turns and the inductor width, we use a classification approach and define 20 and 4 distinct outputs, each one of them corresponding to a particular valid geometric value. A single output from the FCNN corresponds to the inductor's inner radius, since it is a continuous-valued variable and its approximation is handled by regression.

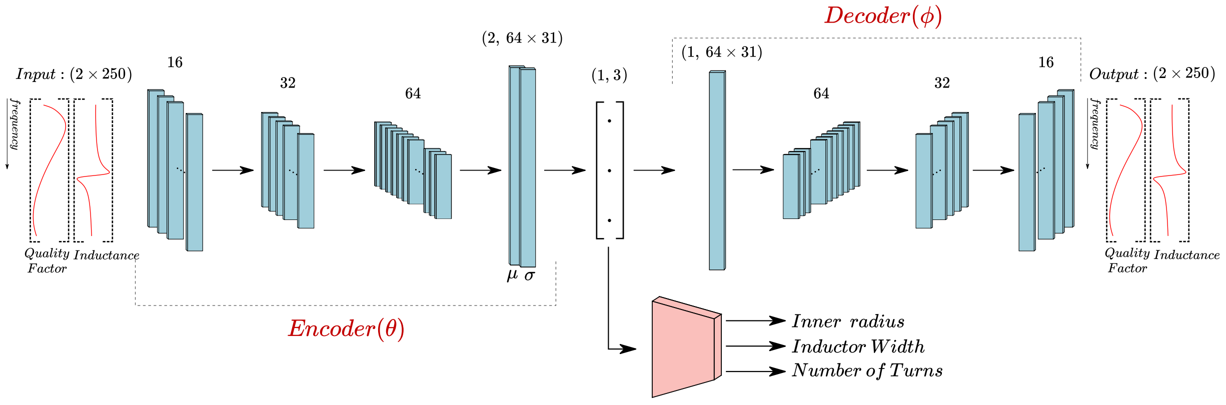


Figure 4. A depiction of the proposed composite model for spiral inductor representation learning. The 1D vectors of inductors' Quality Factor and Inductance frequency behavior are inputs to the 1D convolutional filters of the architecture. The filter sizes are shown as well. The predictor FCNN obtains as an input the latent representation of the inductor's frequency characteristics and yields its geometric sizes.

In order to train the composite model, we need to define its supervised loss as in Equation (16). Let us denote as $\tilde{x}_i$, with $i = 1, \ldots, 25$ the outputs of the predictor net. Outputs $[\tilde{x}_i]_{i=1}^{20}$ correspond to the number of turns, outputs $[\tilde{x}_i]_{i=21}^{24}$ to the inductor's width and output $\tilde{x}_{25}$ to inner radius. For a particular geometry, let us also denote as $\hat{\mathbf{y}} = [y_1, \ldots, y_{25}]$ the ground truth results, where out of the first 20 items all of them are zero, with the exception of the corresponding index of the ground truth number of turns. The same applies for the next four items, and the last one is a continuous variable. Therefore, the ground truth vector $\hat{\mathbf{y}}$ has always 22 zeros, 2 ones, and a real-valued item. The loss function that penalizes deviations from actual geometries is comprised out of three individual losses, L_{NoT} for the number of turns, L_{IW} for the inductor's width and L_{IR} for the inner radius. These are computed via

$$\begin{aligned}
L_{NoT} &= -\sum_{i=1}^{20} y_i \cdot \log\left[\frac{\exp(x_i)}{\sum_{j=1}^{20}\exp(x_j)}\right], \\
L_{IW} &= -\sum_{i=21}^{24} y_i \cdot \log\left[\frac{\exp(x_i)}{\sum_{j=21}^{24}\exp(x_j)}\right], \\
L_{IR} &= (x_{25} - y_{25})^2, \\
\mathcal{L}_{sup} &= L_{NoT} + L_{IW} + L_{IR}.
\end{aligned} \tag{18}$$

We use the squared euclidean loss for the inner radius, and a cross-entropy loss for the two classification-based outputs.

The model was trained using the Adam optimizer for 1000 epochs and a 80–20% training-test split. After training, the test data were mapped to their latent representations, by passing them through the encoder part of the VAE, and the predictor's accuracy was calculated as 94% for number of turns and 96% for inductor width. For the inner radius, the MSE score is 0.11, where the values are normalized in the $[0, 1]$ range. An illustration of the model's operation is also shown in Figure 5, where 500 descaled random samples from the model are depicted. This is completed by sampling from the isotropic, unit variance, zero mean Gaussian distribution $p(\mathbf{z})$ and passing the samples from the decoder network. It is seen that the curves follow a particular trajectory, which resembles the resonant characteristic of the actual inductor's curves.

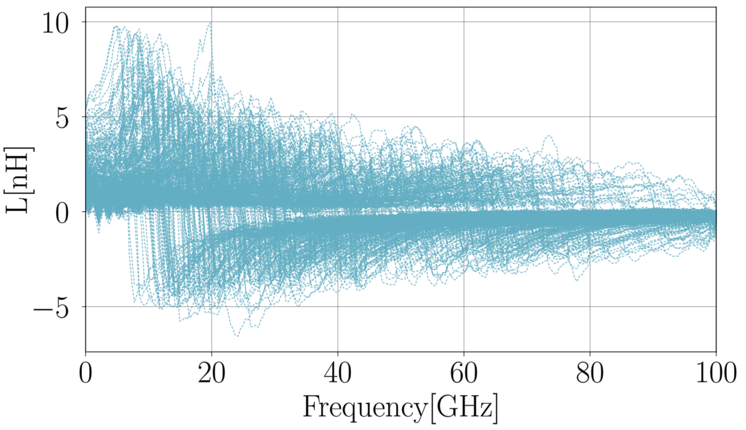


Figure 5. The 500 samples of inductance curves sampled from the generative model.

5.2. Metal Capacitor

In this subsection, the proposed model is used to derive continuous representations for a TSMC 90 nm 'crtmom' rotative metal capacitor. We consider a parametrization with three variables, namely fingers space, number of vertical fingers, and number of horizontal fingers. The width of the fingers is considered fixed, since it contributes relatively little to the frequency behavior of the device. The fingers spacing variable is continuous in the range $[140, 180]$ nm, whereas the rest of the variables take integer values in the range $[6, 200]$.

Similarly to the previous case of spiral inductors, we gather a dataset of 3000 frequency responses in the range of $[0.1, 330]$ GHz. The data are obtained as S_{11} responses from a parametric sp analysis. Instead of following the convolutional approach as in the spiral inductor case, in this case we consider a simpler approach is sufficient. We proceed to select 30 frequencies in the aforementioned range and keep the real and imaginary parts of the S_{11} responses only for them. The VAE model then consists of fully connected encoder and decoder networks, with its input being the concatenated imaginary and real part of each frequency response, i.e., a vector of 60 length. Both the encoder and decoders have three layers with 200, 400, and 600 neurons, with ReLU activations. The chosen latent space dimensionality is $M = 3$ and the predictor network is a three layer FCNN with 50, 50 and three neurons and ReLU activations.

To train the composite, we define its supervised loss $\mathcal{L}_{sup}$ as the summation of three individual losses, L_{HF} for horizontal fingers, L_{VF} for vertical fingers and L_{fspace} for finger spacing. All of them are mean squared losses, defined in a similar way as L_{IR} in Equation (18), and the overall loss is the summation of all of them and the VAE loss. The model is trained for 1000 epochs, using the Adam optimizer. After training, the test data were mapped to their latent representations through the encoder part of the trained VAE and the predictor's mean squared error for all three outputs was 0.13, with the labels being scaled to $[0, 1]$.

6. Sizing Results

In this section, the continuous representations of spiral inductors and metal capacitors are used to automatically size two LNA topologies. For both circuits, we use the composite models discussed in the previous section and the same TSMC 90 nm process. It is worth noting that for device models that are used more than once in a single topology, the same models are utilized, but with different latent variables for optimization.

6.1. Inductively Degenerated LNA

In this example, we consider the LNA topology shown in Figure 6. It is an inductively degenerated single-stage LNA, having three spiral inductors and two metal capacitors, namely C_g and C_d. The supply voltage is 1.2 V and the operating frequency is 2.4 GHz.

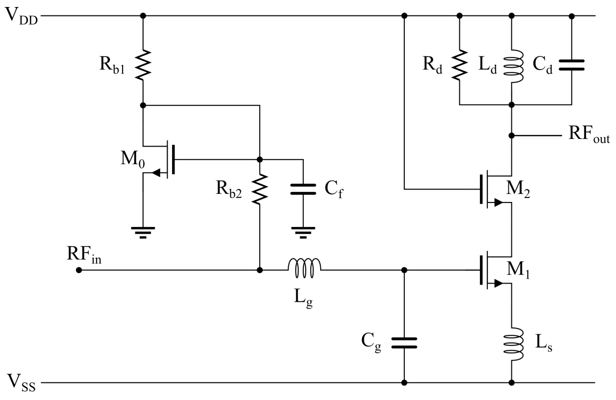


Figure 6. The inductively degenerated LNA considered in this subsection.

This circuit is parametrized by its transistor lengths and widths, the metal resistors' widths and lengths, as well as the geometrics parameters of the capacitors and inductors. The parameters of the inductors in the latent space formulation are substituted with 9 latent variables in total, 3 for each device. Similarly, the parameters of the capacitors amount to 6 latent variables in total. In total, there are 21 variables both in the latent space formulation and in the original variable space. The ranges of the variables, both in their original form and in the transformed-latent space form are given in Table 1.

For automatic sizing, we consider a single objective formulation and wish to minimize the static power consumption P_{dc} of the LNA, while enforcing $IP3 \geq -5$ dBm, $NF \leq 2.5$ dB, $S_{11} \leq -8$ dB, and $S_{22} \leq -8$ dB at the operating frequency at the following corners: ss, sf, fs, and ff, and at working temperatures of -50, 27, and 125 Celsius. Additionally, for the nominal conditions we enforce $S_{21} \geq 21$dB. This amounts to a total of 13 testbenches and 49 constraints $[g(\mathbf{x})]_{i=1}^{49}$. Since there are no matched devices or pairs of devices in the topology of Figure 6, the mismatch effects on the performance of this circuit are not considerable and are not accounted for. However, in the general case, one can account for PVT and mismatch effects by providing testbenches with the appropriate PDK models included to the simulator and defining corresponding constraints in the optimization formulation.

For comparison, we consider the following sizing methodologies:

- The proposed BO algorithm, with the latent space formulation;
- The proposed BO algorithm with the relaxation procedure for integer/discrete variables [5];
- A genetic algorithm operating on the transformed variable space, making use of the proposed models of Section 5 to transform individuals to the original space;
- A genetic algorithm with mixed-variable operators operating on the original variable space.

The hyperparameters of the aforementioned algorithms are chosen as follows: Both BO algorithms have 1200 total evaluations, with $N_S = 8$ RFF samples per iteration and 150 initial samples. The genetic algorithms have 100 individuals per generation, and are allowed to search for 50 generations.

Table 1. LNA Variable Ranges.

Variable	Description	Range
W_M	Transistor Widths	$[1, 120]$ µm
L_M	Transistor Lengths	$[100, 240]$ nm
L_{R1}	Resistor R_{b1} Length	$[0.1, 30]$ µm
L_{R2}	Resistor R_{b2} Length	$[0.1, 30]$ µm
W_{R1}	Resistor R_{b1} Width	$[0.4, 10]$ µm
W_{R2}	Resistor R_{b2} Width	$[0.4, 10]$ µm
IW_{L_g}/L_{L_g1}	Inductor L_g: Width/Latent	$[3, 15]$ µm/$[-3, 3]$
IW_{L_s}/L_{L_s1}	Inductor L_s: Width/Latent	$[3, 15]$ µm/$[-3, 3]$
IW_{L_d}/L_{L_d1}	Inductor L_d: Width/Latent	$[3, 15]$ µm/$[-3, 3]$
$IR_{L_g}L_{L_g2}$	Inductor L_g: Radius/Latent	$[15, 90]$ µm/$[-3, 3]$
IR_{L_s}/L_{L_s2}	Inductor L_s: Radius/Latent	$[15, 90]$ µm/$[-3, 3]$
IR_{L_d}/L_{L_d2}	Inductor L_d: Radius/Latent	$[15, 90]$ µm/$[-3, 3]$
$NT_{L_g}L_{L_g3}$	Inductor L_g: Turns/Latent	$[0.5, 5.25]/[-3, 3]$
NT_{L_s}/L_{L_s3}	Inductor L_s: Turns/Latent	$[0.5, 5.25]/[-3, 3]$
NT_{L_d}/L_{L_s3}	Inductor L_d: Turns/Latent	$[0.5, 5.25]/[-3, 3]$
VF_{C_g}/L_{C_g1}	Capacitor C_g: Vertical Fingers/Latent	$[6, 200]/[-3, 3]$
VF_{C_d}/L_{C_d1}	Capacitor C_d: Vertical Fingers/Latent	$[6, 200]/[-3, 3]$
HF_{C_g}/L_{C_g2}	Capacitor C_g: Horizontal Fingers/Latent	$[6, 200]/[-3, 3]$
HF_{C_d}/L_{C_d2}	Capacitor C_d: Horizontal Fingers/Latent	$[6, 200]/[-3, 3]$
fs_{C_g}/L_{C_g3}	Capacitor C_g: Fingers Spacing/Latent	$[140, 180]$ nm/$[-3, 3]$
fs_{C_d}/L_{C_d3}	Capacitor C_d: Fingers Spacing/Latent	$[140, 180]$ nm/$[-3, 3]$

To account for random fluctuations, we repeat all of the experiments for 10 times. The results of the experiments, with respect to best attained feasible solution, average best feasible solution, standard deviation of best attained feasible solution and success rate are given in Table 2. It is seen that, in the provided simulation budget, both BO formulations outperform the GA ones. The vanilla GA formulation, that works on the original search space, finds feasible solutions two times only, whereas the GA operating in the transformed space, i.e., having only continuous variables, finds feasible solutions 8 out of 10 times. In contrast, both BO formulations find feasible solutions in all experiments.

The efficiency of the proposed approach is underlined by the power consumption results. Among all formulations, the BO that operates in the transformed variable space yields the best solution on average (11.7 mW) and the best solutions out of all formulations and executions (9.5 mW). The BO with relaxation is able to find a single good solution (9.8 mW), but yields in average 13.6 mW of power, which is close to the result of the GA working on the transformed space. The fact that the mixed variable GA formulation has the lowest standard deviation is due to the number of feasible outcomes; only two executions result in feasible results that are taken into account in Table 2.

To study the stability of the resulting circuit parameterizations, we consider the Stern stability factor

$$K = \frac{1 + |\Delta|^2 - |S_{11}|^2 - |S_{22}|^2}{2|S_{21}| \cdot |S_{12}|}, \tag{19}$$

where $\Delta = S_{11}S_{22} - S_{12}S_{21}$. For the circuit to be unconditionally stable, it must hold $K > 1$ and $\Delta < 1$, at all frequencies. We assessed the outcome of the stability criterion on all the feasible and best outcomes of the algorithms examined, on all of the repetitions of Table 2. We considered a frequency range of $[1, 100]$ GHz to measure K and Δ. In all cases, the parametrized circuits are stable. However, this does not hold for all feasible parametrizations encountered during the optimization. An inclusion of these metrics as constraints in the optimization definition would ensure that all feasible solutions are stable. However, the fact that the resulting solutions are stable indicates that constrained circuits are predominantly stable.

Table 2. LNA sizing results—static power consumption.

Formulation	P_{dc}—Best [mW]	P_{dc}—Avg [mW]	P_{dc}—Std [mW]	Success
BO-Latent	9.5	11.7	1.7	10/10
BO-Relaxation	9.8	13.6	2.2	10/10
GA-Latent	11.9	14.3	2.5	8/10
GA	13.8	14.7	1.6	2/10

A graphical illustration of the performance of all formulations in the sizing procedure of the LNA is given in Figure 7, where the evolution of the Constraint Violation (CV) metric is given, against the number of evaluations. The value of the CV is computed as

$$\mathrm{CV}(\mathbf{x}) = \sum_{i=1}^{49} \max[g_i(\mathbf{x}), 0]. \tag{20}$$

Here, it is seen that the BO that works in the transformed space requires roughly 600 evaluations to find feasible solutions. In the case of the BO with the relaxation procedure, all executions find feasible solutions at around 1100 evaluations. The mixed-variable GA's CV seems to stagnate at 4000 evaluations, whereas the transformed space formulation of GA seems to improve its CV even at the end of the experiment.

6.2. Wideband LNA

In this subsection, a wideband, noise cancelling LNA [28] shown in Figure 8 is sized. This topology consists of Common-Gate and a Common-Source-Common-Gate stage, along with a source follower output buffer. It works on a 1.8 V supply and its operating frequency range is $[2, 5]$ GHz.

In a similar way as in the previous example, this circuit is parametrized by its transistor widths and lengths, its metal resistors widths and lengths, and the geometric sizes of the capacitors and inductors. All of the capacitors are rotative metal ones, and the inductors are spiral octagonal ones, making the proposed models in Section 5 applicable. In addition, there are three biasing voltages, namely V_{b1}, V_{b2} and V_{b3} to be selected. Following the guidelines in the original implementation, all capacitors are chosen to be identical and the transistors share the same gate length. In total, there are 34 design variables, both in the original variable space and in the latent space formulation. The ranges of the variables, both latent ones and the original ones, are given in Table 3. The variable space of the optimization will be denoted as $\mathbb{S}$.

For automatic sizing, we consider again a single objective formulation. In order to compare the results of the sizing formulations in a principled way, we consider the following Figure of Merit (FoM) [29] for wideband LNAs and set its maximization as our optimization goal:

$$\mathrm{FoM}(\mathbf{x}) = 20log_{10}\left(\frac{G_{max}(\mathbf{x}) \times IIP3_{max}(\mathbf{x}) \times BW(\mathbf{x})}{(F_{min}(\mathbf{x}) - 1) \times P_{dc}(\mathbf{x})}\right). \tag{21}$$

The term G_{max} indicates the maximum voltage gain of the LNA in V/V, $IIP3_{max}$ is the maximum input-referred third-order intercept point in mW, BW is the LNAs bandwidth and F_{min} is the minimum noise figure in the operating frequency range, in linear units. In our formulation, we consider the bandwidth fixed to 3 GHz. To ensure this, we consider the following constraint function:

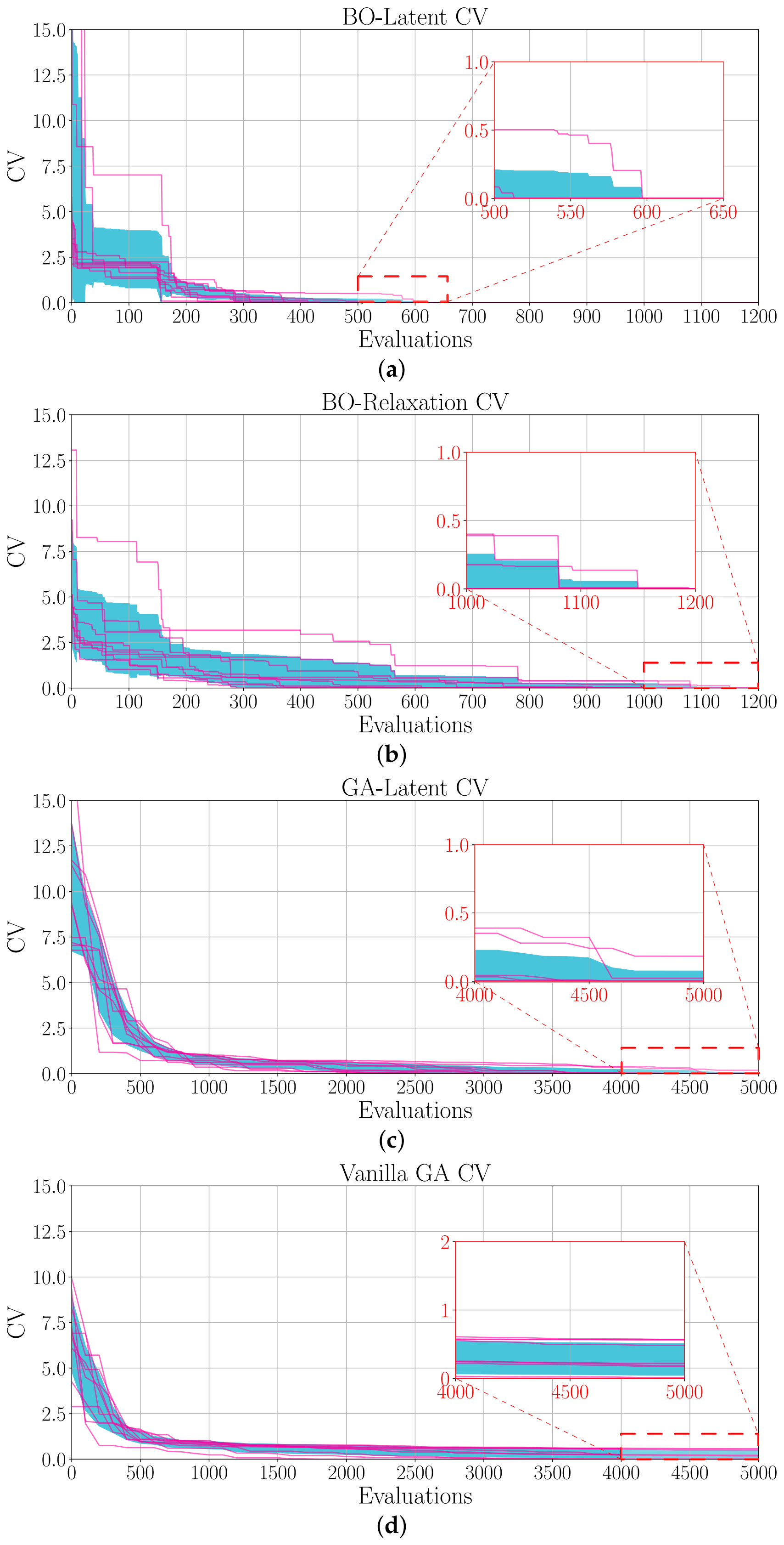


Figure 7. The LNA's CV for the discussed formulations. The blue area indicates the confidence region of ±std, while the purple lines are the curves of each repetition. (**a**) BO-Latent. (**b**) BO-rounding. (**c**) GA-Latent. (**d**) GA.

$$g(\mathbf{x}) = \max_{f \in [2,5]\text{GHz}} \left[S_{11}(\mathbf{x}, f) \right] + 10, \quad \mathbf{x} \in \mathbb{S} \tag{22}$$

with $g(\mathbf{x}) \leq 0$ being the feasibility criterion. Here, function $S_{11}(\mathbf{x}, f)$ is the S_{11} result of the simulation, in dBs. The satisfaction of this constraint ensures that the results will be matched to a 50 Ω input port. Thus, the sizing of this particular LNA topology can be formulated as

$$\begin{aligned} \min \quad & \text{FoM}(\mathbf{x}), \quad \mathbf{x} \in \mathbb{S} \\ \text{s.t.} \quad & g(\mathbf{x}) \leq 0 \end{aligned} . \tag{23}$$

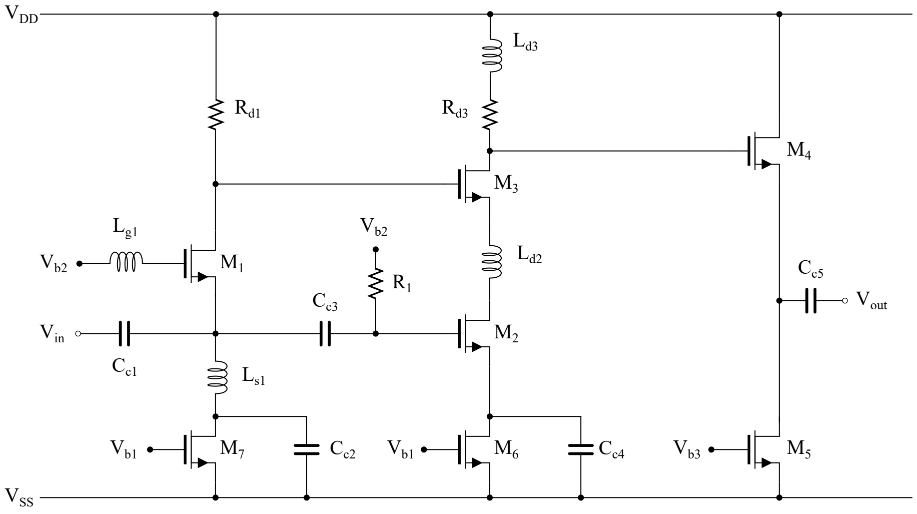


Figure 8. The wideband, noise canceling LNA [28] considered in this subsection.

For comparison, we consider again the same four optimization algorithm cases as in the previous subsection, with the hyperparameters of the algorithms remaining the same. The experiments were executed 10 times to account for random fluctuations.

Table 4 depicts the sizing results using the different formulations, with respect to the objective FoM. Among the considered cases and under the optimization budget given, the proposed BO with the latent space representations of inductors and capacitors yields in average a FoM of 63.9 dBs, whereas the same BO with the relaxation of the discrete variables yields 58 dBs. In addition, the latent space formulation seems to be reliable, since it yields less variance (5.5 dB standard deviation) in comparison to the relaxation approach (9.3 dBs).

Regarding the GA sizing formulations, both the GA that operates in the latent space and the vanilla GA succeed in finding feasible solutions at all experiments. However, the GA-latent formulation is more successful with respect to its resulting FoM, which is roughly 23 dBs higher. Both GAs do not succeed in overpassing BO in the obtained FoM performance, both in averaged results and in terms of the best valued acquired throughout the 10 repetitions. The above highlights the fact that the utilized BO variant is able to approximate the global optimum better, in comparison to the population-based GA, under the given simulation budgets.

The fact that both BO and GA formulations that work with the continuous representations outperform their mixed-variable counterparts underlines the efficiency of our approach. This improvement for the case of BO can be attributed to the fact that there is no brute force rounding of the variables, enabling the GP models to assign correlations to points in the variable space by using their Euclidean distance. In the case of the GA, the results can be attributed to the fact that continuous-only operators work better, and the fact that the latent space of the devices is created in such a way that functionally similar devices are close-by, which assists in the exploration of the variable space. In fact, a simple perturbation of a particular discrete variable may change in a great extent the behavior of the device, such as the spiral inductor.

Table 3. Wideband LNA variable ranges.

Variable	Description	Range
L_M	Transistor Lengths	[100, 240] nm
W_{M1}	M_1 Width	[1, 120] µm
W_{M2}	M_2Width	[1, 120] µm
W_{M3}	M_3 Width	[1, 120] µm
W_{M4}	M_4Width	[1, 120] µm
W_{M5}	M_5 Width	[1, 120] µm
W_{M6}	M_6Width	[1, 120] µm
W_{M7}	M_7Width	[1, 120] µm
L_{Rd1}	Resistor R_{d1} Length	[0.1, 30] µm
L_{Rd2}	Resistor R_{d2} Length	[0.1, 30] µm
L_{Rd3}	Resistor R_{d3} Length	[0.1, 30] µm
L_{R1}	Resistor R_1 Length	[0.1, 30] µm
W_{Rd1}	Resistor R_{d1} Width	[0.4, 10] µm
W_{Rd2}	Resistor R_{d2} Width	[0.4, 10] µm
W_{Rd3}	Resistor R_{d3} Width	[0.4, 10] µm
W_{R1}	Resistor R_1 Width	[0.4, 10] µm
V_{b1}	Volatge V_{b1}	[0.2, 0.5] µm
V_{b2}	Volatge V_{b2}	[0.2, 0.5] µm
V_{b3}	Volatge V_{b3}	[0.2, 0.5] µm
$IW_{L_g1}/L_{L_{g1}1}$	Inductor L_{g1}: Width/Latent	[3, 15] µm/[−3, 3]
$IW_{L_s1}/L_{L_{s1}1}$	Inductor L_{s1}: Width/Latent	[3, 15] µm/[−3, 3]
$IW_{L_d2}/L_{L_{d2}1}$	Inductor L_{d2}: Width/Latent	[3, 15] µm/[−3, 3]
$IW_{L_d3}/L_{L_{d3}1}$	Inductor L_{d3}: Width/Latent	[3, 15] µm/[−3, 3]
$IR_{L_g1}/L_{L_{g1}2}$	Inductor L_{g1}: Radius/Latent	[15, 90] µm/[−3, 3]
$IR_{L_s1}/L_{L_{s1}2}$	Inductor L_{s1}: Radius/Latent	[15, 90] µm/[−3, 3]
$IR_{L_d2}/L_{L_{d2}2}$	Inductor L_{d2}: Radius/Latent	[15, 90] µm/[−3, 3]
$IR_{L_d3}/L_{L_{d3}2}$	Inductor L_{d3}: Radius/Latent	[15, 90] µm/[−3, 3]
$NT_{L_g1}/L_{L_{g1}2}$	Inductor L_{g1}: Turns/Latent	[0.5, 5.25]/[−3, 3]
$NT_{L_s1}/L_{L_{s1}2}$	Inductor L_{s1}: Turns/Latent	[0.5, 5.25]/[−3, 3]
$NT_{L_d2}/L_{L_{d2}2}$	Inductor L_{d2}: Turns/Latent	[0.5, 5.25]/[−3, 3]
$NT_{L_d3}/L_{L_{d3}2}$	Inductor L_{d3}: Turns/Latent	[0.5, 5.25]/[−3, 3]
VF/L_{C1}	Capacitors: Vertical Fingers/Latent	[6, 200]/[−3, 3]
HF/L_{C2}	Capacitors: Horizontal Fingers/Latent	[6, 200]/[−3, 3]
fs/L_{C3}	Capacitors: Fingers Spacing/Latent	[140, 180] nm/[−3, 3]

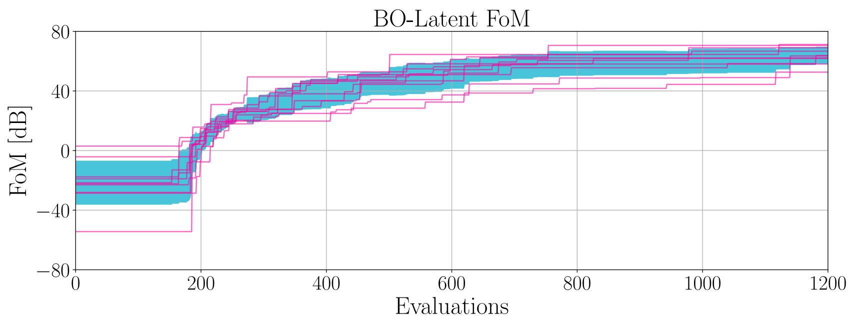
The best and feasible results of the algorithms in Table 4 are checked for their stability using the Stern factor of Equation (19). Using the same frequency range as in the previous subsection, the parameterized circuits are unconditionally stable.

Table 4. Wideband LNA sizing results.

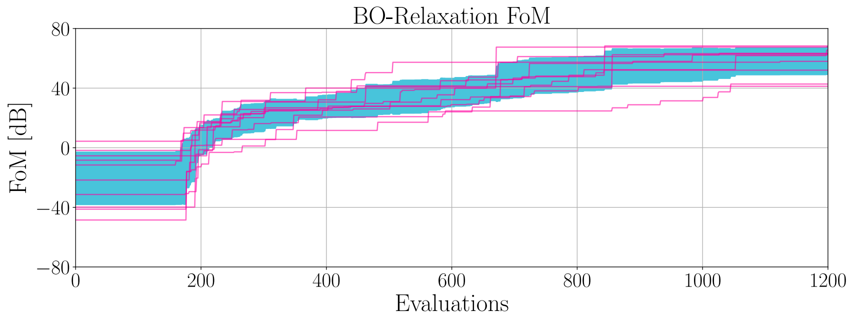
Formulation	FoM—Best [dB]	FoM—Avg [dB]	FoM—Std [dB]	Success
BO-Latent	71.1	63.9	5.5	10/10
BO	68.3	58.4	9.3	10/10
GA-Latent	65.9	58.2	8.5	10/10
GA	49.8	35.6	8.3	10/10

For a graphical comparison, Figure 9 demonstrates the evolution of the LNA's FoM metric for each of the 4 formulations considered, against the number of evaluations. It is seen that the GAs require roughly 800 to 1500 evaluations for reaching a feasible solution, which is seen as an abrupt deviation in the FoM metric in these plots. For the BO cases, the first feasible solutions are found below 200 evaluations. On average, the BO cases require much less evaluations to reach acceptable results, in comparison to the population-based

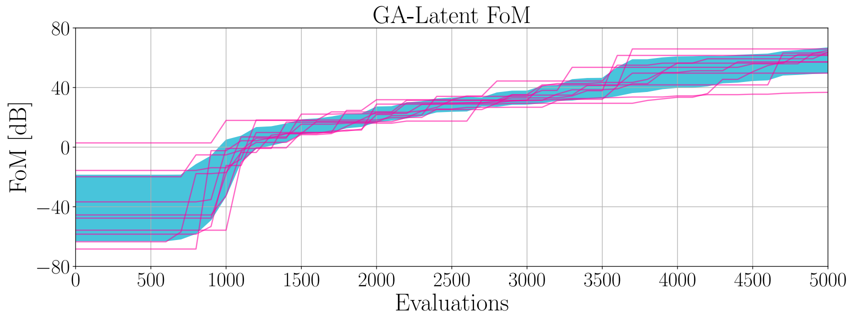
GAs. The BO operating in the transformed variable space has the smallest variance in results, highlighted by the range of the blue region in the respective figure.



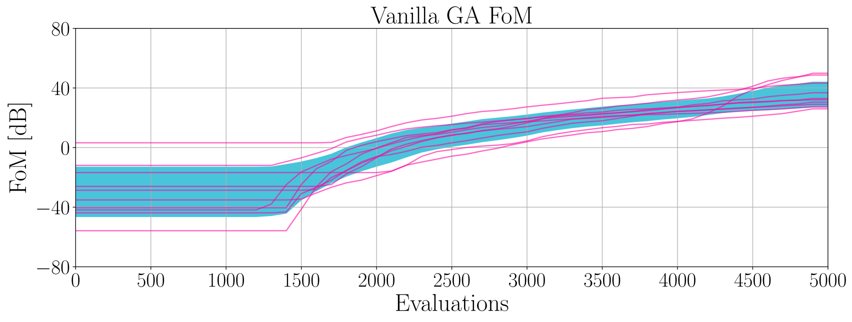
(**a**)



(**b**)



(**c**)



(**d**)

Figure 9. The evolution of the LNA's FoM during the automatic sizing, using the four discussed formulations. The blue area indicates the confidence region of ±std, while the purple lines are the curves of each repetition. (**a**) BO-Latent. (**b**) BO-relaxation. (**c**) GA-latent. (**d**) GA.

7. Conclusions

This paper presented a new approach to low-budget analog circuit automatic sizing by combining a batched version of BO with learned continuous representations of integrated devices. By using simulation data and a composite model of a VAE and a FCNN, continuous representations of integrated devices that are originally parametrized by discrete variables are derived. These representations are used to render circuit sizing problems as continuous optimization ones, which are then solved using black-box optimizers. Experimental results on two real-world circuits suggest that the proposed approach, when combined with a proposed BO variant, outperforms other state-of-the-art approaches to mixed-variable optimization for analog circuit sizing.

Author Contributions: Conceptualization, K.T. and P.P.S.; methodology, K.T.; software, K.T.; validation, K.T. and P.P.S.; formal analysis, K.T.; investigation, K.T.; resources, K.T.; data curation, K.T.; writing—original draft preparation, K.T.; writing—review and editing, P.P.S.; visualization, K.T.; supervision, P.P.S.; project administration, P.P.S.; funding acquisition, K.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research is co-financed by Greece and the European Union (European Social Fund—ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Moore, G.E. Cramming more components onto integrated circuits. *Proc. IEEE* **1998**, *86*, 82–85. [CrossRef]
2. Kahng, A.B.; Lienig, J.; Markov, I.L.; Hu, J. *VLSI Physical Design: From Graph Partitioning to Timing Closure*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2011.
3. White, D. A new methodology to address the growing productivity gap in analog design. In Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 18–21 November 2013.
4. Razavi, B.; Behzad, R. *RF Microelectronics*; Prentice Hall: New York, NY, USA, 2011.
5. Touloupas, K.; Sotiriadis, P.P. LoCoMOBO: A local constrained multi-objective Bayesian optimization for analog circuit sizing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 2780–2793. [CrossRef]
6. Martins, R.; Lourenço, N.; Horta, N.; Yin, J.; Mak, P.-I.; Martins, R.P. Many-objective sizing optimization of a class-c/d vco for ultralow-power iot and ultralow-phase-noise cellular applications. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2019**, *27*, 69–82. [CrossRef]
7. Lourenço, N.; Martins, R.; Horta, N. *Automatic Analog IC Sizing and Optimization Constrained with PVT Corners and Layout Effects*; Springer: Cham, Switzerland, 2017.
8. Lyu, W.; Xue, P.; Yang, F.; Yan, C.; Hong, Z.; Zeng, X.; Zhou, D. An efficient bayesian optimization approach for automated optimization of analog circuits. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *65*, 1954–1967. [CrossRef]
9. Liu, B.; Fernández, F.V.; Gielen, G.G. Efficient and accurate statistical analog yield optimization and variation-aware circuit sizing based on computational intelligence techniques. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2011**, *30*, 793–805. [CrossRef]
10. Toulupas, K.; Chouridis, N.; Sotiriadis, P.P. Local Bayesian Optimization For Analog Circuit Sizing. In Proceedings of the ACM/IEEE Design Automation Conference, San Francisco, CA, USA, 5–9 December 2021.
11. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*; Springer: Berlin/Heidelberg, Germany, 2003.
12. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; Freitas, N.D. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE* **2015**, *104*, 148–175. [CrossRef]
13. Williams, C.K.; Rasmussen, C.E. *Gaussian Processes for Machine Learning*, 3rd ed.; MIT Press: Cambridge, MA, USA, 2006.
14. Tripp, A.; Daxberger, E.; Hernández-Lobato, J.M. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 11259–11272.
15. Antonova, R.; Rai, A.; Li, T.; Kragic, D. Bayesian Optimization in Variational Latent Spaces with Dynamic Compression. In Proceedings of the Conference on Robot Learning, Virtual Conference, 16–18 November 2020.
16. Bombarelli, G.; Wei, J.N.; Duvenaud, D.; Hernández-Lobato, J.M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T.D.; Adams, R.P.; Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **2018**, *4*, 268–276. [CrossRef] [PubMed]
17. Deshwal, A.; Doppa, F.; Deisenroth, M. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 8185–8200.
18. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006.
19. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. In Proceedings of the 2nd International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.

20. Canelas, A.; Póvoa, R.; Martins, R.; Lourenço, N.; Guilherme, J.; Carvalho, J.P.; Horta, N. FUZYE: A Fuzzy *c*-Means Analog IC Yield Optimization Using Evolutionary-Based Algorithms. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *39*, 1–13. [CrossRef]
21. Castagnola, J.L.; Dualibe, F.C.; Laprovitta, A.M.; García-Vázquez, H. A Novel Design and Optimization Approach for Low Noise Amplifiers (LNA) Based on MOST Scattering Parameters and the gm/ID Ratio. *Electronics* **2020**, *9*, 785. [CrossRef]
22. Youssef, A.A.; Murmann, B.; Omran, H. Analog IC design using precomputed lookup tables: Challenges and solutions. *IEEE Access* **2020**, *8*, 134640–134652. [CrossRef]
23. Shawe-Taylor, J.; Cristianini, N. *Kernel Methods for Pattern Analysis*; Cambridge University Press: Cambridge, UK, 2004.
24. Liu, C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
25. Wilson, J.; Hutter, F.; Deisenroth, M. Maximizing acquisition functions for Bayesian optimization. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Volume 31.
26. Rahimi, A.; Recht, B. Random features for large-scale kernel machines. In Proceedings of the 20th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; Volume 20.
27. Drakaki, M.; Hatzopoulos, A.A.; Siskos, S. Cmos inductor performance estimation using z- and s-parameters. In Proceedings of the 2007 IEEE International Symposium on Circuits and Systems, New Orleans, LA, USA, 27–30 May 2007.
28. Kumar, A.R.A.; Sahoo, B.D.; Dutta, A. A Wideband 2–5 GHz Noise Canceling Subthreshold Low Noise Amplifier. *IEEE Trans. Circuits Syst. II Express Briefs* **2018**, *65*, 834–838. [CrossRef]
29. De Souza, M.; Mariano, A.; Taris, T. Reconfigurable inductorless wideband CMOS LNA for wireless communications. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2016**, *64*, 675–685 [CrossRef]