

# Reshaping Bayesian Optimization for Design Space Optimization Towards Accurate and Irredundant Evaluation in EDA Tool Parameter Exploration

Chanhee Jeon and Taewhan Kim

Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea  
{cjeon7, tkim}@snucad.snu.ac.kr

**Abstract**—Finding an optimal tool parameter configuration that achieves optimal design PPA (performance, power, area) through Design Space Optimization (DSO) in Physical Design (PD) has become increasingly important due to the rising complexity of Electronic Design Automation (EDA) tool chains in modern VLSI design. DSO is particularly challenging due to the high dimensionality of tool parameters, the abundance of discrete parameter options, and, most critically, the non-linear relationship between parameters and design PPA. This work overcomes two limitations of the prior state-of-the-art Bayesian Optimization (BO) based DSO methods for EDA tool parameter optimization. The two limitations are (1) a *poor correlation of the similarity computation in BO engine between two sampling points with the actual similarity between the corresponding post-layout PPAs*, resulting in far from the Pareto-optimal PPA exploration; (2) *redundant evaluations occur frequently when discrete parameters are quantized*. Precisely, we overcome limitation 1 by training AE (AutoEncoder) model using PPA outcomes of the prior sample points and using it to reshape the latent parameter space such that similarity in latent vectors aligns with the similarity in PPA, thereby justifying the *accurate* kernel-based similarity function in BO, while we address limitation 2 by reformulating the acquisition function in BO in a way to effectively sample the discrete parameter values in the continuous design space. In the meantime, through experiments, it is shown that using our DSO method with reshaped BO amenable to EDA tool parameter optimization is able to find tool parameter options of 59% larger HyperVolume and up to 16% improvement for single objective (i.e., a weighted sum of PPA) optimization.

**Index Terms**—Design space optimization, Bayesian Optimization (BO), machine learning, physical design.

## I. INTRODUCTION

With the advancement of technology nodes, the role of Electronic Design Automation (EDA) tools has become increasingly critical in Physical Design (PD) flow due to the rising complexity of VLSI chip implementation. Each stage of PD flow is associated with numerous EDA tool parameters, and their option combinations strongly affect chip power, performance, and area (PPA). As the number of parameters grows, their interactions become highly complex and non-linear, making manual parameter tuning infeasible.

Defining a set of all parameter option combinations as a design space, its size can easily reach an unacceptable level. (For instance, a design space of  $10^9$  configurations is common.) In common sense, an exhaustive or a manual exploration of such design space to sample an optimal or even a sub-optimal parameter set is practically impossible. To

address this issue, machine learning (ML)-based design space optimization (DSO) frameworks have emerged in recent years.

One of the earliest efforts to apply ML-DSO in the PD domain is *AutoTuner* [1], which adopts generic hyperparameter optimization algorithms from AI domain into the OpenROAD [2] framework. Though AutoTuner demonstrates its feasibility, it requires thousands of tool executions, raising concerns about its practicality, thus motivating the need for PD domain-specific DSO frameworks.

Following AutoTuner, several PD domain-specific DSO engines have been proposed. *SynTunSys (STS)* [3], [4] employs a pseudo-genetic algorithm and *FIST* [5] utilizes an iterative XGBoost refinement framework while it is shown that reinforcement learning-based methods (e.g., [6]) explore optimization more precisely. Recently, *BOXGB* [7] combines Bayesian Optimization (BO) [8] with XGBoost [9], achieving strong results when optimizing an objective function formulated as a weighted sum of multiple PPA improvements, while *RemoTune* [10] introduces a ‘trust region’ based BO to enable parallel multi-objective optimization. Among them, BOXGB demonstrates stable and effective optimization performance, and RemoTune shows strong results in Pareto-frontier enlargement.

However, both BOXGB and RemoTune merely use BO engine itself with no consideration of fine-tuning towards accurate and effective PD domain space exploration. The followings are two critical BO factors that have not been fine-tuned by the prior state-of-the-art BO-based DSO methods:

1. BO computes similarity using Euclidean distance between parameter vectors. *Since design PPAs are highly non-linear with parameter vectors, Euclidean distance between parameters or even the embedding vectors with reduced dimensions poorly correlates with true PPA similarity, significantly degrading optimization performance.*
2. BO in the PD domain has a tendency to fall into local minima when handling parameter spaces with discrete variables, which are abundant in PD domain. In such case, the discrete parameters must be quantized to a finite set of values; in other words, any continuous values close to a quantized point are mapped to the same discrete option. As a result, *the Gaussian Process Regressor (GPR) used in BO repeatedly samples values near previously chosen*

values, which often leads the optimization to be trapped in local minima.

This work proposes a novel DSO framework that overcomes these two limitations by employing a *modified version of AutoEncoder* (for limitation 1) and an *acquisition function with proximity penalty* (for limitation 2). The key contributions of our work are summarized as:

- **Tightly integrating BO with an AutoEncoder to align the optimization space with the design PPA space, thereby justifying the similarity function in BO:** We propose a modified AutoEncoder in Sec. III-B trained with a novel distortion loss function that reshapes the latent space such that the latent distances align with design PPA differences. We integrate it into our BO engine to compute the similarity score between the sampled points, which is highly correlated to the closeness of the corresponding design PPAs. This preserves the Positive Semi-Definiteness of the RBF kernel while enabling BO to perform similarity computation that meaningfully reflects design PPA behavior.
- **Ensuring robustness to discrete parameters with proximity penalty:** Our proposed BO engine explicitly alleviates the inherent limitation of BO that frequently samples redundant parameters in the exploration space with discrete parameters. By employing so called a proximity penalty term in Sec. III-C to avoid redundant evaluation of equivalent configurations, our DSO framework improves efficiency and ensures robustness in discrete search space.
- **Parallelizing BO with proximity penalty function:** Aside from being robust to the discrete parameters using proximity penalty, it leads the next sampling points to be distant from the previously sampled points. This allows BO to select multiple distant points with high posterior variance or high potential of HyperVolume improvement (EHVI), enabling an effective parallel exploration.

## II. BO-BASED DSO PROBLEM AND MOTIVATIONS

### A. Problem Formulation

Table I shows a list of commonly used EDA tool parameters in PD flow where *LS-PL* indicates the stage of logic synthesis followed by placement while *CTS* and *Route* indicate the stages of clock tree synthesis and routing, respectively. Note that the parameters which appear in multiple stages are adjusted independently at each stage; thus, the same parameter may take different options across different stages.

Consider a design space with parameter set  $P$ :

$$P = \{p_1, p_2, \dots, p_i, \dots\}; p_i = \{\rho_i^1, \rho_i^2, \dots, \rho_i^{j_i}\}, \quad (1)$$

where  $\rho_i^j$  denotes the  $j^{\text{th}}$  option of parameter  $p_i$ , quantized to the range of  $[-1, 1]$  and  $j_i$  is the number of available options of  $p_i$ . For example, parameter ‘‘Clock uncertainty’’ in Table I has an original option list of  $\{0\%, 1\%, 2\%\}$ , which is then quantized to  $\{-0.66, 0, 0.66\}$  to be fitted into  $[-1, 1]$ .

TABLE I  
EXAMPLE OF PD PARAMETERS WITH THEIR OPTIONS (I.E., VALUES) AND PD STAGES. BLUE OPTIONS INDICATE DEFAULT CONFIGURATIONS.

#	Parameter	Option	PD Stage		
			LS-PL	CTS	Route
1	Max fanout	[8, 16, <b>24</b> , 32, 40, 48, 56, 64]	✓	✓	✓
2	Max transition (%)	[10, 12.5, <b>15</b> , 17.5, 20] (%)	✓	✓	✓
3	Clock uncertainty (%)	[ <b>0</b> , 1, 2] (%)	✓	✓	✓
4	Global max density	[0.80, 0.85, 0.90, <b>0.95</b> ]	✓		
5	Clock power effort	[ <b>low</b> , standard, high]	✓		
6	Congestion effort	[low, <b>medium</b> , high]	✓		
7	Legalization instance gap	[ <b>0</b> , 1, 2]	✓		
8	Uniform density	[true, <b>false</b> ]	✓		
9	Wirelength opt level	[none, <b>medium</b> , high]	✓		
10	Opt max density	[ <b>0.80</b> , 0.85, 0.90, 0.95]	✓	✓	✓
11	Reclaim area	[true, false, <b>default</b> ]	✓	✓	✓
12	Power effort	[ <b>none</b> , low, high]	✓	✓	✓
13	Early global route level	[standard, <b>medium</b> , low]	✓	✓	
14	CTS Buffer / Inverter type	[ <b>Option 1</b> , Option 2, Option 3]		✓	
15	CTS max fanout	[12, <b>20</b> , 28, 36, 44, 52]		✓	
16	Skew target (%)	[10, <b>15</b> , 20] (%)		✓	
17	CTS max transition (%)	[10, 12.5, <b>15</b> , 17.5, 20] (%)		✓	
18	Route with timing driven	[true, <b>false</b> ]			✓
19	Fixing DRC effort	[high, <b>medium</b> , low]			✓

TABLE II  
SYMBOLS AND THEIR DEFINITIONS.

Symbol	Definition
$\mathbf{x}, \mathbf{X}$	Enumerated and quantized parameter combination and its set. ( $\mathbf{x} \in \mathbf{X}$ )
$\mathbf{y}, \mathbf{Y}$	PPA evaluated for a corresponding $\mathbf{x}$ and its set normalized with $\mathbf{r}$ . ( $\mathbf{y} \in \mathbf{Y}$ )
$\mathbf{r}$	Reference PPA evaluated with default parameters in Table I
$\mathbf{z}$	Latent vector (dimension-reduced representation of parameter set $\Lambda$ )
$\mathbf{Z}$	Latent space, with $\mathbf{z} \in \mathbf{Z}$ .
$\hat{\mathbf{x}}$	Reconstructed parameter set decoded from $\mathbf{z}$ .
$\mathcal{K}(\cdot, \cdot)$	Kernel function measuring similarity between two points.
$l, \xi$	Hyperparameters that control the sensitivity of $\mathcal{K}$ and <i>proximity penalty</i>

Given the design space  $P$ , a parameter set  $\mathbf{x}$  is sampled by selecting one option per parameter. i.e.,

$$\mathbf{x} = \{\rho_1^{k_1}, \rho_2^{k_2}, \dots, \rho_i^{k_i}, \dots\}; 1 \leq k_i \leq j_i, \quad (2)$$

and evaluated it through PD tool execution as  $\mathbf{y} = f(\mathbf{x})$  where  $\mathbf{y} = [y_{perf}, y_{power}, y_{area}]$  denotes the normalized PPA vector for configuration  $\mathbf{x}$  with respect to a reference PPA  $\mathbf{r}$  in the last three columns on Table IV. The normalization is performed as following:

$$y_m = \frac{\mathbf{r}_m - \mathcal{P}_m(\mathbf{x})}{\mathbf{r}_m}; m \in \{perf, power, area\}, \quad (3)$$

where  $\mathcal{P}_m(\mathbf{x})$  is the PPA value evaluated with parameter set  $\mathbf{x}$  and normalized with target metric  $m$ . The performance metric  $y_{perf}$  is derived from the effective clock period, which is  $t_{clk} - t_{WNS}$  where  $t_{clk}$  is the clock period and  $t_{WNS}$  is the worst negative slack time. The definitions of all symbols used in this work are listed in Table II.

Since PPA metrics (performance, power, area) are inherently in trade-off, PD domain DSO is naturally formulated as a multi-objective optimization problem. Hence, the objective of our proposed DSO is to sample a Pareto-optimal configuration set  $\mathbf{X}^*$ :

$$\mathbf{X}^* = \{\mathbf{x}_1^*, \dots, \mathbf{x}_i^*, \dots\}; \mathbf{x}_i^* = \{\rho_i^{k_1^*}, \rho_i^{k_2^*}, \dots, \rho_i^{k_i^*}\}, \quad (4)$$

that maximizes the HyperVolume (HV) of Pareto frontier, which is a bounded volume size evaluated by  $\mathbf{Y}^* = f(\mathbf{X}^*)$ :

$$HV(\mathbf{Y}^*, \mathbf{o}) = \int_{\mathbb{R}^3} \mathbf{1}\{\exists \mathbf{y}^* \in \mathbf{Y}^* : \mathbf{o} \preceq \mathbf{y} \preceq \mathbf{y}^*\} d\lambda_3(\mathbf{y}); \quad (5)$$

where  $\mathbf{o}$  denotes the origin corresponding to the reference PPA  $\mathbf{r}$  normalized with Eq.3, and  $\lambda_3$  and  $\mathbf{1}\{\cdot\}$  are 3-dimensional Lebesgue measure and an indicator function, respectively. In addition,  $\mathbf{y} \preceq \mathbf{y}^*$  indicates that  $\mathbf{y}^*$  dominates  $\mathbf{y}$ . This is,  $\mathbf{y}$  is no worse in all objectives and strictly better in at least one. In simple words, our goal is to maximize the volume of the region bounded by the origin  $\mathbf{o}$  and Pareto points  $\mathbf{y}^* \in \mathbf{Y}^*$ .

### B. Motivations

Bayesian Optimization (BO) is an iterative strategy for exploring an expensive black-box function. At each iteration, BO constructs a Gaussian Process Regressor (GPR) to approximate the true objective function and guides the exploration to maximize the acquisition function. In this work, following the state-of-the-art researches for PD domain DSO, prior to GPR construction, we reduce the parameter vector  $\mathbf{x}$  into a lower-dimensional latent vector  $\mathbf{z}$ . This alleviates the degradation in exploration performance in a high-dimensional exploration space. However, two critical limitations of BO still remain:

- **(Limitation 1) Latent vector similarity does not reflect design PPA similarity:** After  $n$  iterations, BO accumulates dataset  $\mathcal{D}_n = \{\mathbf{z}_i, \mathbf{y}_i\}_{i=1}^n$  and constructs GPR with a Gaussian prior:

$$\mathbf{y} \sim \mathcal{N}(0, \mathcal{K}_{ij})_{i,j=1}^n; \mathcal{K}_{ij} = \mathcal{K}(\mathbf{z}_i, \mathbf{z}_j) \quad (6)$$

where  $\mathcal{K}(\mathbf{z}_i, \mathbf{z}_j)$  is a kernel function that measures the similarity between  $\mathbf{z}_i$  and  $\mathbf{z}_j$ . A commonly used kernel function is the Radial Basis Function (RBF) [11]:

$$\mathcal{K}(\mathbf{z}_i, \mathbf{z}_j) = \text{RBF}(\mathbf{z}_i, \mathbf{z}_j) = \exp\left(-\frac{\|\mathbf{z}_i - \mathbf{z}_j\|^2}{l}\right) \quad (7)$$

where  $l$  is a hyperparameter that controls the sensitivity of the similarity measure. A larger  $l$  results in slower decay, making distant points appear more similar, while a smaller  $l$  leads to sharper decay.

Though RBF kernel is symmetric and positive semi-definite, the Euclidean distance  $\|\mathbf{z}_i - \mathbf{z}_j\|$  often fails to reflect the true

objective-space distance  $\|\mathbf{y}_i - \mathbf{y}_j\|$ , as specified in Fig. 1(a). This mismatch weakens the validity of using RBF kernel for BO in PD. To address this issue, two choices exist: (*choice 1*) replacing RBF kernel with a more expressive kernel or (*choice 2*) redefining the latent space itself. While non-linear kernels exist, none of them universally guarantee both accurate non-linear mapping and positive semi-definiteness; thus, *choice 1* is infeasible. Hence, we adopt *choice 2*. As illustrated in Fig. 1(b), we propose to use an *AutoEncoder* architecture with distortion loss  $\mathcal{L}_{\text{distort}}$  to reshape  $\mathbf{z}$  such that kernel distances align closely with design PPA distances.

- **(Limitation 2) Redundant sampling in discrete parameter spaces:** BO is prone to repeatedly sample equivalent configurations when parameters are quantized. For example, suppose there is a design space with 10 binary parameters, each quantized to  $-0.5$  and  $0.5$ . If the acquisition function reaches its maximum at  $0.3$  for each dimension, GPR will repeatedly sample vectors with positive entries, all of which quantize to  $0.5$  in all 10 binary parameters. In this case, BO explores only a single configuration, which is only  $1/1024$  ( $= 1/2^{10}$ ) of the entire design space. To resolve this issue, we introduce so called a *proximity penalty* which is able to encourage to sample the latent vectors distant from the previously sampled configurations.

## III. PROPOSED FRAMEWORK

### A. Structure of our BO-based DSO Framework

Fig. 2 shows the overall flow of our proposed DSO framework. Starting from AutoEncoder pre-training, our DSO engine proceeds two main stages: (1) exploration-centric stage (green box) and (2) exploitation-centric stage (blue box). Both stages involve actual PD tool running to obtain design PPA results.

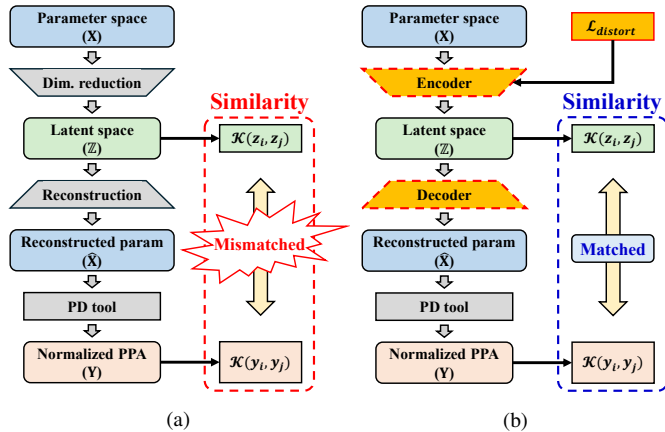


Fig. 1. (a) Conventional parameter search process with dimension reduction. (b) Our proposed parameter search process with AutoEncoder. Our work mitigates the mismatch between  $\mathcal{K}(\mathbf{z}_i, \mathbf{z}_j)$  and  $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j)$  by integrating AutoEncoder trained with  $\mathcal{L}_{\text{distort}}$ , which will be described in Sec. III-B.

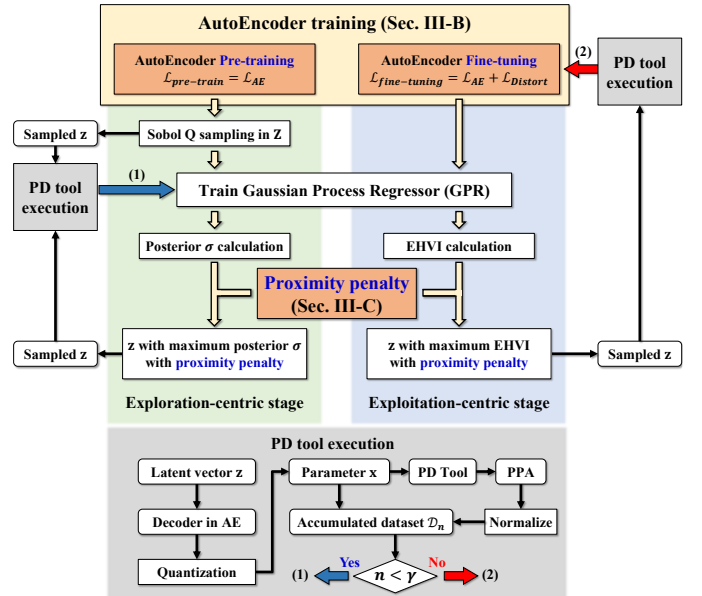


Fig. 2. The overall flow of our proposed DSO framework.

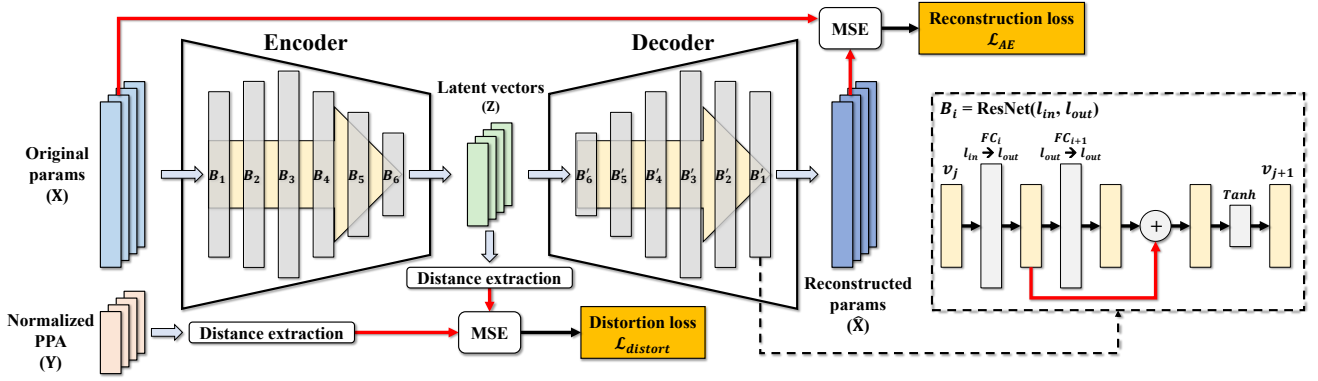


Fig. 3. Overall architecture of AutoEncoder and training loss details. The ResNet block I/O sizes are shown in Table III.

• **Exploration-centric stage:** The purpose of this stage is to expand the GPR’s understanding on the design space. The process iteratively samples latent vectors  $\mathbf{z} \in \mathbf{Z}$  using the Sobol quasi-random sequence (Sobol-Q) [12], which generates samples that approximate a uniform distribution. The latent vector sampled at each iteration  $n = 1, 2, \dots$  is evaluated through the following steps:

1. The decoder in our AutoEncoder reconstructs an enumerated parameter set from the latent vector.
2. The enumerated parameters are quantized into a valid configuration,  $\mathbf{x}$ , which is translated into raw tool options.
3. By executing PD tool with  $\mathbf{x}$ , the resulting PPA vector is normalized and recorded in the accumulated dataset  $\mathcal{D}_n$  together with  $\mathbf{x}$ .
4. If  $n < \gamma$  (a predefined evaluation threshold), the process repeats (blue arrow labeled with (1)); otherwise, the framework proceeds to AutoEncoder fine-tuning (red arrow labeled with (2)).

After accumulating initial samples with Sobol-Q, GPR is trained on  $\mathcal{D}_n$  to compute the posterior mean  $\mu$  and variance  $\sigma^2$ . In this stage, the acquisition function selects latent vectors that maximize only the posterior variance  $\sigma$  combined with proximity penalty (PP) formulated in Sec. III-C. Our proximity penalty offers two key advantages: (1) *robustness to discrete parameters* and (2) *effective parallelization*.

• **Exploitation-centric stage:** This stage aims to aggressively maximize the quantity of HyperVolume  $HV(\cdot)$  in Eq.5. Unlike the exploration-centric stage, this stage fine-tunes our AutoEncoder using the accumulated dataset  $\mathcal{D}_n$  and the loss term that includes the distortion loss  $\mathcal{L}_{distort}$  proposed in Sec. III-B. This ensures that the difference in latent vectors more faithfully represents the differences in their design PPAs.

Within this refined latent space, GPR provides more meaningful similarity evaluations, for which the acquisition function, defined as  $EHVI(\mathbf{z})$  in Eq.11, is augmented with the proximity penalty term. The sampled points are evaluated using the same processing steps (1-4) in the exploration-centric stage.

#### B. AutoEncoder: Align Latent Similarity with PPA Similarity

Fig. 3 shows our proposed AutoEncoder (AE) architecture consisting of 6-layer encoder and 6-layer decoder, both of

TABLE III  
INPUT & OUTPUT DIMENSIONS OF LAYERS IN AUTOENCODER

Encoder		Decoder	
Block name	ResNet name	Block name	ResNet name
$B_1$	ResNet (32, 64)	$B'_1$	ResNet (64, 32)
$B_2$	ResNet (64, 128)	$B'_2$	ResNet (128, 64)
$B_3$	ResNet (128, 64)	$B'_3$	ResNet (64, 128)
$B_4$	ResNet (64, 32)	$B'_4$	ResNet (32, 64)
$B_5$	ResNet (32, 16)	$B'_5$	ResNet (16, 32)
$B_6$	ResNet (16, 12)	$B'_6$	ResNet (12, 16)

which are implemented with ResNet blocks [13] with  $\tanh$  activation function. The encoder maps the original high-dimensional tool parameters to a reduced-dimensional latent space  $\mathbf{Z}$  while the decoder recovers the original parameter space from  $\mathbf{Z}$ .

Our AE is trained in two stages: (1) *pre-training* which is to learn a compact manifold of the parameter space, and (2) *fine-tuning* which is to overcome limitation 1 (i.e., similarity discrepancy) of the prior works described in Sec. II-B.

1. *Pre-training:* Our AE is trained in a self-supervised manner by exclusively using the design parameter vectors. The objective is to minimize the mean squared error (MSE) between the original parameters and their recovered, i.e., decoded ones:

$$\mathcal{L}_{pre-training} = \mathcal{L}_{AE} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 \quad (8)$$

where  $\mathbf{x}_i$  denotes an original parameter and  $\hat{\mathbf{x}}_i$  its reconstructed one.

2. *Fine-tuning:* Our AE is then fine-tuned at the beginning of each iteration of the exploitation stage. Besides the reconstruction loss, we devise a *distortion loss*, denoted by  $\mathcal{L}_{distort}$ , to align latent distances with their corresponding design PPA distances:

$$\begin{aligned} \mathcal{L}_{distort} &= \text{MSE} \left( \text{dist}(\mathbf{Z}, \mathbf{Z}), \text{dist}(\mathbf{Y}, \mathbf{Y}) \right) \\ &= \frac{1}{N} \sum_{i,j} \left( \|\mathbf{z}_i - \mathbf{z}_j\|_2 - \|\mathbf{y}_i - \mathbf{y}_j\|_2 \right)^2 \quad (9) \end{aligned}$$

where  $\mathbf{z}$  are the distorted latent vectors and  $\mathbf{y}$  are their corresponding PPA results. The fine-tuning loss is then defined as:

$$\mathcal{L}_{fine-tuning} = \mathcal{L}_{AE} + \mathcal{L}_{distort}. \quad (10)$$

By incorporating the distortion loss, our fine-tuned AE ensures that the sample similarity in the latent space closely reflects the similarity in PPA results. This alignment allows the kernel function in BO to capture a more reliable similarity correlation among samples.

### C. Acquisition Function with Proximity Penalty

The conventional multi-objective BO selects sampling candidates that have high  $EHVI(\cdot)$  values:

$$\alpha(\mathbf{z}) = EHVI(\mathbf{z}) = \mathbb{E}_{\mathbf{y}}[HV(\mathcal{S} \cup \{\mathbf{y}\}, \mathbf{r}) - HV(\mathcal{S}, \mathbf{r})] = \int_{\mathbb{R}^m} \mathbb{I}_{HV}(\mathbf{y}; \mathcal{S}, \mathbf{r}) \phi_m(\mathbf{y}; \mu, \sigma) d\mathbf{y} \quad (11)$$

where  $\mathcal{S}$  is the set of design PPAs of the previously sampled latent vectors,  $\phi_m$  is the multivariate normal pdf, and  $\mathbb{I}_{HV}$  is the HyperVolume improvement contributed by the predicted design PPA  $\mathbf{y}$  of a latent sample  $\mathbf{z}$ .

However, the conventional BO often selects latent vectors that are decoded to the same quantized parameter set when discrete parameters are employed. Precisely, in real-world DSO, quantized parameters yield discrete PPAs, as illustrated by the yellow histogram bars in Fig. 4, in which the heights indicate the design PPA (i.e.,  $\mathbf{y}$ ) values. Using  $\alpha$  in Eq.11 by BO with no modification shall sample  $\mathbf{z}$  corresponding to the blue dot shown on the left histogram even though the previous samples of three red dots have already found a maximal PPA.

To mitigate this redundancy, we incorporate a *proximity penalty (PP)* defined in Eq.13 into  $\alpha$  to discourage candidates near previously sampled points. Precisely, a new acquisition  $\alpha'$  in Eq.12 penalizes the design space close to the previously sampled points, as indicated by the reposition of the red dots on the blue curve shown on the right side in Fig. 4, by which  $\alpha'$  is able to search for candidates (e.g. the blue dot) with high PPA potential.

Formally, our modified acquisition function  $\alpha'$  is defined as:

$$\alpha'(\mathbf{z}) = \alpha(\mathbf{z}) \cdot \mathbf{PP}; \quad (12)$$

$$\mathbf{PP} = 1 - \max_{\mathbf{p} \in \mathcal{S}} \exp\left(-\frac{\|Dec(\mathbf{z}) - Dec(\mathbf{p})\|^2}{\xi}\right) \quad (13)$$

where  $\xi$  is a hyperparameter that controls the decay rate of the proximity penalty. Similar to the length-scale parameter  $l$

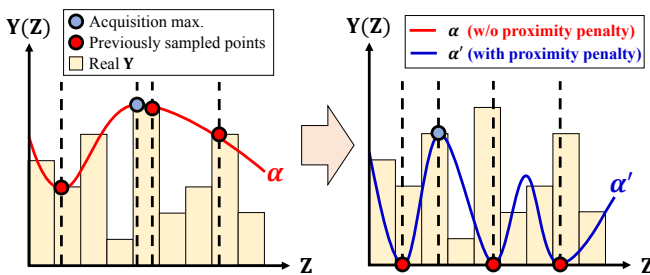


Fig. 4. Effect of the proximity penalty on the acquisition function for quantized parameters. Left: original acquisition function  $\alpha$ , which may sample  $\mathbf{z}$  (blue dot) close to an already sampled one (red dots). Right: modified acquisition function  $\alpha'$  with proximity penalty, which drops down the sampling probability on local space close to the previously sampled points.

in Eq.7, a larger  $\xi$  makes the proximity penalty less sensitive to the distance between the current candidate  $\mathbf{z}$  and previously sampled points.  $Dec(\cdot)$  returns, from a latent vector in AE, the recovered parameter vector of original parameter space by our decoder in AE.  $\mathbf{z}$  is a new sample candidate of latent vector, and  $\mathcal{S}$  is the set of previously sampled latent vectors.

Consequently, if there is a small value of  $\|Dec(\mathbf{z}) - Dec(\mathbf{p})\|$  (i.e.,  $\mathbf{z}$  is close to  $\mathbf{p}$ ) in the proximity penalty term in Eq.13,  $\alpha'(\mathbf{z})$  will exponentially decrease. Our modification of acquisition function into  $\alpha'(\cdot)$  provides two distinct benefits: (1) minimizing redundant sampling, (2) enabling parallel sampling.

## IV. EXPERIMENT RESULTS

### A. Experiment setup

Table IV summarizes the design information and reference PPA of 7 OpenCore [14] circuits. The reference PPA results were obtained by running PD flow with the default parameter settings listed in Table I. All experiments were conducted with ASAP 7nm PDK [15], using *Synopsys Design Compiler* [16] for logic synthesis and *Cadence Innovus* [17] for P&R.

TABLE IV  
INFORMATION OF TESTED DESIGNS AND **reference PPA**.

Design	Clk period (ps)	Util (%)	#Gate	#Net	#FF	$\Gamma_{power}$ (mW)	$\Gamma_{perf}$ (ps)	$\Gamma_{area}$ ( $\mu m^2$ )
I2C_MASTER	250	80	954	859	128	0.97	296	1584.3
SPI	230	80	2,522	2,358	229	2.29	332	3873.1
TV80	440	80	4,947	4,856	357	1.99	602	7064.7
USB_FUNCT	260	80	10,591	10,140	1,737	5.09	316	18616.4
AC97	210	80	12,731	13,045	2,199	18.91	311	22518.8
WB_CONMAX	260	70	28,725	40,121	768	11.95	416	35272.0
AES128	270	80	194,583	159,541	10,688	87.50	326	239840.3

The overall optimization framework is implemented on Botorch [18], and AutoEncoder (AE) is built on Pytorch [19]. The AE is pre-trained for  $10^6$  epochs and fine-tuned for  $10^3$  epochs at each exploitation iteration, using a learning rate of 0.001 with exponential decay of 0.9 for every 10% of the corresponding total epochs. The proximity penalty hyperparameter  $\xi$  is fixed to 0.1 in all experiments.

### B. Effectiveness of Our AutoEncoder Training

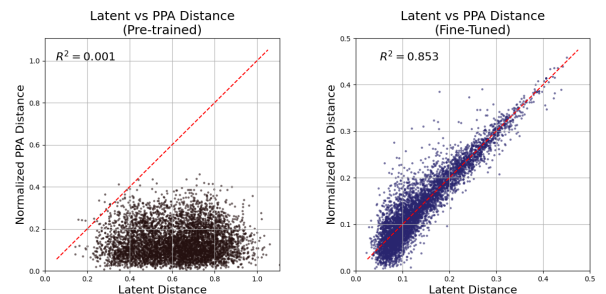


Fig. 5. Correlation between latent distance and normalized PPA distance for design AC97: (a) pre-training and (b) fine-tuning. Fine-tuning significantly improves the alignment between latent space and PPA space, as reflected by the higher  $R^2$  value.

TABLE V  
OPTIMIZATION PERFORMANCE COMPARISON WITH PREVIOUS WORKS. BLUE BOLDED RESULTS ARE THE BEST RESULTS AMONG THE FRAMEWORKS.

Designs	BOXGB [7] (50 iterations)					RemoTune [10] (10 iterations)					Ours (10 iterations)				
	Maximum improvement (%)			Score	HV	Maximum improvement (%)			Score	HV	Maximum improvement (%)			Score	HV
	$y_{power}$	$y_{perf}$	$y_{area}$			$y_{power}$	$y_{perf}$	$y_{area}$			$y_{power}$	$y_{perf}$	$y_{area}$		
I2C_MASTER	9.3	15.5	9.1	31.7	1225.6	9.8	15.9	8.8	34.5	1372.1	<b>11.1</b>	<b>16.6</b>	<b>9.5</b>	<b>36.0</b>	<b>1696.2</b>
SPI	4.7	19.3	8.4	29.9	645.7	<b>12.4</b>	21.7	<b>8.9</b>	37.6	<b>1886.0</b>	11.9	<b>30.1</b>	6.9	<b>42.4</b>	1829.5
USB_FUNCT	7.1	11.7	5.1	23.3	415.1	6.5	<b>19.0</b>	3.3	28.3	410.0	<b>9.0</b>	18.0	<b>5.7</b>	<b>30.1</b>	<b>852.4</b>
AC97	7.7	28.0	10.0	43.2	2065.7	8.5	28.0	9.6	41.7	2128.7	<b>11.6</b>	<b>32.2</b>	<b>10.9</b>	<b>53.4</b>	<b>4002.6</b>
TV80	<b>5.1</b>	11.3	<b>4.0</b>	<b>16.7</b>	<b>146.9</b>	0.4	19.6	1.0	12.1	3.7	1.5	<b>23.3</b>	3.0	12.3	30.6
WB_CONMAX	<b>10.0</b>	12.7	<b>9.3</b>	<b>29.9</b>	<b>1139.8</b>	5.5	15.6	5.7	23.7	433.2	5.7	<b>20.0</b>	4.8	22.5	301.8
AES128	4.3	7.7	6.1	15.4	161.8	6.0	11.0	<b>7.7</b>	22.2	438.1	<b>8.1</b>	<b>14.4</b>	7.0	<b>23.0</b>	<b>509.4</b>
Average	6.89	15.17	<b>7.43</b>	27.15	828.66	7.02	18.69	6.44	28.58	953.12	<b>8.41</b>	<b>22.07</b>	6.83	<b>31.39</b>	<b>1317.50</b>
Normalized	1.00	1.00	<b>1.00</b>	1.00	1.00	1.02	1.23	0.87	1.05	1.15	<b>1.22</b>	<b>1.45</b>	0.92	<b>1.16</b>	<b>1.59</b>

Our AutoEncoder is first pre-trained to compress quantized parameters into low-dimensional latent vectors. However, as noted in *Limitation 1*, latent distances do not initially reflect PPA distances. This is explicitly shown in Fig. 5(a) where  $R^2$  correlation is only 0.001. After fine-tuning with our distortion loss  $\mathcal{L}_{distort}$ , the latent distances align closely with PPA distances, achieving an  $R^2$  score above 0.85, as shown in Fig. 5(b). Thus, our fine-tuned AE successfully overcomes *Limitation 1*, ensuring that the kernel function  $\mathcal{K}$  in BO is justified by meaningful latent similarity.

### C. Effectiveness of Our DSO Framework

Table V shows a comparison of the optimization performance of **BOXGB** [7], **RemoTune** [10], and **our DSO**. The *maximum improvement* metric corresponds to the maximum values of  $y_{power}$ ,  $y_{perf}$ , and  $y_{area}$ , which have already been normalized to the **reference PPA** in Table IV. The *score* is defined as the maximum sum of these improvements while *HV* denotes the HyperVolume defined in Eq.5. For a fair comparison, BOXGB is evaluated with 50 iterations due to its sequential nature whereas RemoTune and our proposed method use 10 iterations each.

On average, our proposed DSO achieves the best performance, achieving 16% and 59% higher single-objective score and HV in comparison with that of BOXGB, and 10% and 38% improvement over RemoTune, which clearly shows that our DSO outperforms the prior state-of-the-art methods in both single- and multi-objective optimization.

### D. Ablation Study

Table VI shows a summary of an ablation study for our BO based DSO. *AE* denotes our AutoEncoder trained with  $\mathcal{L}_{distort}$  to address *Limitation 1*, while *PP* denotes the proximity penalty introduced to address *Limitation 2*.

Individually, both modules improve performance: AE achieves  $3.47\times$  increase in HV, and PP achieves  $2.15\times$  increase compared to Vanilla BO. By combining two modules, our DSO attains  $3.54\times$  higher HV, demonstrating complementary benefits. Notably, for design TV80, our DSO is the only one that achieves all non-zero HVs, highlighting its robustness. These results clearly justify the integration of both AE and PP in terms of improved optimization stability and overall performance.

TABLE VI  
ABLATION STUDY FOR OUR PROPOSED DSO. **PP** STANDS FOR PROXIMITY PENALTY, AND **AE** STANDS FOR AUTOENCODER.

Engine name		<b>BO</b>	<b>BO + PP</b>	<b>BO + AE</b>	<b>Ours</b>
Modules	BO	✓	✓	✓	✓
	PP		✓		✓
	AE			✓	✓
Design		HyperVolume (HV)			
I2C_MASTER		614.5	1289.2	1683.5	<b>1696.2</b>
SPI		624.5	35.7	594.3	<b>1829.5</b>
USB_FUNCT		177.5	1385.6	<b>1794.2</b>	852.5
AC97		753.0	2230.1	3773.3	<b>4002.6</b>
TV80		0.0	0.0	0.0	<b>30.6</b>
WB_CONMAX		290.3	371.1	<b>693.6</b>	301.8
Average		410.0	885.3	1423.2	<b>1452.2</b>
Normalized		1.00	2.16	3.47	<b>3.54</b>

## V. CONCLUSION

We proposed a new BO-based DSO framework that (1) reshaped the optimization space with AutoEncoder trained with a distortion loss to ensure that the kernel similarity in BO should be well aligned with actual PPA similarity, and (2) incorporated a proximity penalty with acquisition function to suppress redundant evaluations in quantized discrete spaces while enabling parallel sampling. Through experiments, it was shown that our DSO framework outperformed the state-of-the-art DSO methods, achieving on average 16% and 10% higher single-objective score as well as 59% and 38% larger HyperVolume over that of BOXGB [7] and RemoTune [10], respectively. In summary, we devised a DSO framework that ensured *accurate* and *irredundant evaluation*, effectively addressing key limitations of prior BO-based DSO methods.

## ACKNOWLEDGEMENT

This work was supported by National Research Foundation by Korea Government (2021-R1A2C2008864), IITP2023-RS-2023-00256081, Samsung Electronics Co., Ltd (IO201216-08205-01, LSI250306-0010, IO241217-11507-01), SAIT (IO250331-12476-01), ISRC (Inter-University Semiconductor Research Center) in SNU, and BK21 Four Program of Education and Research Program for Future ICT Pioneers in SNU. EDA tool was supported by IDEC.

## REFERENCES

- [1] J. Jung, A. B. Kahng, S. Kim, and R. Varadarajan, "METRICS2.1 and flow tuning in the ieee ceda robust design flow and openroad iccad special session paper," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021.
- [2] T. Ajayi and D. Blaauw, "Openroad: Toward a self-driving, open-source digital layout implementation tool chain," in *Government Microcircuit Applications and Critical Technology Conference*, 2019.
- [3] M. M. Ziegler, H.-Y. Liu, G. Gristede, B. Owens, R. Nigaglioni, and L. P. Carloni, "A synthesis-parameter tuning system for autonomous design-space exploration," in *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016.
- [4] M. M. Ziegler, J. Kwon, H.-Y. Liu, and L. P. Carloni, "Online and offline machine learning for industrial design flow tuning:(invited-iccad special session paper)," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021.
- [5] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, and B. Khailany, "FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.
- [6] A. Agnesina, K. Chang, and S. K. Lim, "Vlsi placement parameter optimization using deep reinforcement learning," in *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020.
- [7] C. Jeon, D. Won, J. Yang, K.-M. Choi, and T. Kim, "BOXGB: Design parameter optimization with systematic integration of bayesian optimization and xgboost," in *IEEE/ACM Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024.
- [8] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyperparameter optimization," *Advances in Neural Information Processing Systems (NIPS)*, vol. 24, 2011.
- [9] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *ACM International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.
- [10] S. Zheng, H. Geng, C. Bai, B. Yu, and M. D. Wong, "Boosting vlsi design flow parameter tuning with random embedding and multi-objective trust-region bayesian optimization," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, pp. 1–23, 2023.
- [11] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, pp. 273–297, 1995.
- [12] P. Bratley and B. L. Fox, "Algorithm 659: Implementing sobol's quasirandom sequence generator," *ACM Transactions on Mathematical Software (TOMS)*, pp. 88–100, 1988.
- [13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [14] Albrecht and Christoph. IWLS 2005 benchmarks. [Online]. Available: <http://www.iwls.org/iwls2005/benchmarks.html>
- [15] L. T. Clark, V. Vashishtha, L. Shifren, A. Gujja, S. Sinha, B. Cline, C. Ramamurthy, and G. Yeric, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, pp. 105–115, 2016.
- [16] Synopsys, "Synopsys Design Compiler." [Online]. Available: <https://www.synopsys.com/>
- [17] Cadence, "Cadence Innovus." [Online]. Available: <https://www.cadence.com>
- [18] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems (NIPS)*, 2020. [Online]. Available: <http://arxiv.org/abs/1910.06403>
- [19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2019.