BNN-YEO: an efficient Bayesian Neural Network for yield estimation and optimization

Zhenxing Dou[*]
School of Integrated Circuit Science and Engineering, Beihang University
Beijing, China
zhenxing@buaa.edu.cn

Ming Cheng
Department of Civil, Chemical, Environmental, and Materials Engineering
University of Bologna
Bologna, Italy
ming.cheng3@unibo.it

Ming Jia
XC Micro Technologies
Nanjing, China
jia.ming@xcmicro.com

Peng Wang[†]
School of Integrated Circuit Science and Engineering, LMIB, Beihang University
Beijing, China
wang.peng@buaa.edu.cn

ABSTRACT

Yield estimation and optimization is ubiquitous in modern circuit design but remains elusive for large-scale chips. This is largely due to the mounting cost of transistor-level simulation and one's often limited resources. In this study, we propose a novel framework to estimate and optimize yield using Bayesian Neural Network (BNN-YEO). By coupling machine learning method with Bayesian network, our approach can effectively integrate prior knowledge and is unaffected by the overfitting problem prevalent in most surrogate models. With the introduction of a smooth approximation of the indicator function, it incorporates gradient information to facilitate global yield optimization. We examine its effectiveness via numerical experiments on 6T SRAM and found that BNN-YEO provides 100x speedup (in terms of SPICE simulations) over standard Monte Carlo in yield estimation, and 20x faster than the state-of-the-art method for total yield estimation and optimization with improved accuracy.

KEYWORDS

Bayesian Neural Network, yield estimation, yield optimization

ACM Reference Format:

Zhenxing Dou, Ming Cheng, Ming Jia, and Peng Wang. 2024. BNN-YEO: an efficient Bayesian Neural Network for yield estimation and optimization. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3658242

[*]First author.
[†]Corresponding author

1 INTRODUCTION

Yield estimation and optimization is ubiquitous in modern circuit design and becoming increasingly paramount with the rising cost of advanced-node chip [5, 11]. At nano-scale, the impacts of process variations such as intra-die mismatches and doping fluctuation are substantial, while aleatory uncertainty from quantum tunnelling further complicates one's effort to maximize yield in the hyperspace of numerous design parameters.

Yield estimation and optimization amounts to the capture of rare events in the distribution tail. Such low probability are traditionally approximated by Monte Carlo-based methods, i.e. via a number of SPICE simulation (Simulation Program with Integrated Circuit Emphasis) [13]. But they becomes forbiddingly expensive as the cost of each simulation soars in tandem with the number of transistors integrated on a chip and are unaffordable with limited computational resources and time. As a result, various sampling techniques have been proposed to reduce the total number of simulations, including Importance Sampling (IS)[7], Quasi-Monte Carlo (Quasi-MC) [9], and Latin Hypercube Sampling (LHS) [10].

Surrogate model is an alternative approach to estimate and optimize yield. By constructing a functional relationship between parameters (inputs) and yield (output), it supplements SPICE simulations and thus reduces the total costs. For example, Polynomial is a popular choice and its polynomial chaos expansion with spectral convergence [4]. Meta-model using Low-Rank Tensor Approximation (LRTA) is another apporoach which can be solved efficiently with a robust greedy algorithm and calibrated iteratively with an adaptive sampling method [8]. But as the number of random parameters rises, the surrogate approach would inevitably suffer the "Curse of dimensionality", leading to exceedingly high cost. Meanwhile, model distortion may also occur in polynomials of high degree due to the Runge-Kutta phenomenon and overfitting.

In recent years, hybrid framework that combines sampling and surrogate model have become increasingly popular. Yao et al. [14] proposed a rapid SRAM yield analysis method that utilizes both importance sampling and an online surrogate model using adaptive SPICE simulations. In a later study, Wang et al. [11] introduced a heuristic two-stage Adaptive Yield Optimization (AYO) method, in which Bayesian optimization is employed for optimal design. But its two-stage Monte Carlo introduces meticulous tuning of hyperparameters and thus incurs additional costs. Recently, Yin et al. addressed such issue by proposing a Bayesian yield analysis (BYA) and modified the Gaussian quadrature with the Bernoulli link function [15]. It incorporates an efficient activation learning process by searching for critical regions with extra simulation for each yield estimation. As a result, additional SPICE sampling are needed for yield optimization should the optimization process be too long or the training data set is flawed.

To address the aforementioned challenges in yield estimation and optimization, we propose a Bayesian Neural Network with the following novelty:

- *Bayesian Neural Network as surrogate model.* To the best of the authors' knowledge, this is the first study incorporating Bayesian Neural Network to estimate and optimize yield.
- *Fast optimization without compromise on yield estimation.* Smooth approximation facilities the utilization of gradient information and expedites yield optimization.
- *Overfitting prevention.* A global surrogate model is constructed so that it alleviates the risk of overfitting prevalent in most surrogate models.
- *Total cost reduction.* No additional SPICE simulations are needed in optimization once surrogate model training is completed. Based on SRAM, BNN-YEO is 20x faster than the SOTA yield optimization method BYA.

The remaining sections of this paper are organized as follows: In Section 2 we provide a review of the research background and related techniques in yield prediction and optimization. Section 3 outlines our Bayesian Neural Network for yield estimation and optimization. Section 4 illustrates and discusses the experimental results of our approach along with those of the baseline methods, and conclusions are drawn in Section 5.

2 PROBLEM FORMULATION

Denote $\mathbf{x} = \left[x_1, x_2, \cdots, x_{N_\text{x}}\right]^\text{T}$ and $\mathbf{v} = \left[v_1, v_2, \cdots, v_{N_\text{x}}\right]^\text{T}$ as vectors of size N_x representing design parameters of a circuit and the manufacturing process variation errors, respectively. Uncertainty on the values of the parameters and errors render them as random variables in the related space: $\mathbf{x} \in \mathcal{X}$ and $\mathbf{v} \in \mathcal{V}$. Following standard practice, we assume that the random errors are independent and of standard Gaussian distributions, i.e,

$$p(\mathbf{v}) = (2\pi)^{-\frac{N_\text{x}}{2}} \prod_{j=1}^{N_\text{x}} \exp\left(-\frac{(v_j)^2}{2}\right). \tag{1}$$

Let $(\mathbf{x}^{(i)}, \mathbf{v}^{(j)})$ be given values of design parameters and process variations, where the superscripts i and j represents the i- and j-th sample, respectively. Corresponding circuit performances such as memory read/write time, amplifier gain and etc., $\mathbf{y}^{(i)} = \left[y_1, y_2, \cdots, y_{N_\text{y}}\right]^\text{T}$, can be obtained from SPICE simulation:

$$\mathbf{y}^{(i)} = \mathbf{f}\left(\mathbf{x}^{(i)}, \mathbf{v}^{(j)}\right), \tag{2}$$

where $\mathbf{f}(\cdot)$ represents the SPICE simulator function. If all performance metrics $\mathbf{y}^{(i)}$ satisfies a pre-defined design specifications $\mathbf{y}^*$, the parameters value $\mathbf{x}^{(i)}$ is considered successful. Otherwise, it is categorized as failure design. This binary result can be represented by a indicator function $I\left(\mathbf{x}^{(i)}, \mathbf{v}^{(j)}\right)$, in which a value of 1 suggests success design and 0 otherwise. Denoting the probabilistic density function of process variation errors as $p(\mathbf{v})$, one can define circuit yield for the given values design parameters:

$$P_\text{f}(\mathbf{x}^{(i)}) = \int_{\mathcal{V}} I\left(\mathbf{x}^{(i)}, \mathbf{v}\right) p(\mathbf{v})\, \text{d}\mathbf{v}, \tag{3}$$

Once yield is estimated, it undergoes optimization to find the optimal values of design parameters $\mathbf{x}^*$:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \; P_\text{f}(\mathbf{x}), \quad \text{s.t.} \;\; \mathbf{x} \in \mathcal{X}. \tag{4}$$

There are two major challenges in yield design. One is the computational costs of SPICE simulations in estimating yield. For standard practice such as Monte Carlo (MC) method, yield is often approximated by its sample mean,

$$\hat{P}_\text{f}(\mathbf{x}^{(i)}) = \frac{1}{M} \sum_{j=1}^{M} I\left(\mathbf{x}^{(i)}, \mathbf{v}^{(j)}\right). \tag{5}$$

where $\{\mathbf{v}^{(j)}\}_{j=1}^{M}$ is the sample set for process error and generated in accordance with its distribution $p(\mathbf{v})$. As the number of samples increases ($M \longrightarrow +\infty$), the yield approximation $\hat{P}_\text{f}$ would statistically converge to its true value P_f. However, in practice, failure is often deemed a very small number, slow convergence rate $O(1/\sqrt{M})$ of Monte Carlo would require a very large number of SPICE simulations and thus incurs significant cost. The second challenge is the lack of analytical form of failure probability ($P_\text{f}(\mathbf{x})$) as a function of design parameters. Without it nor its derivative ($\partial P_\text{f}/\partial \mathbf{x}$), the optimization process of yield is at best a guess.

3 BNN-YEO FRAMEWORK

Our method, BNN-YEO, consists of two parts. We first employ Bayesian Neural Network to construct a surrogate model for yield with a relatively small number of SPICE simulations. Once such model is trained, it serves as the objective function in optimization process and information on its gradients with respect to design parameter can be utilized for fast convergence.

3.1 Yield Estimation with Bayesian Neural Network

Bayesian Neural Network (BNN) leverages prior knowledge to build a highly accurate surrogate model of circuit performances ($\mathbf{y}$) with a limited number of SPICE simulations[3, 12]:

$$\mathbf{y} \approx \tilde{\mathbf{y}}(\mathbf{x}, \mathbf{v}; \boldsymbol{\theta}), \tag{6}$$

where $\mathbf{x}$ are design parameters, $\mathbf{v}$ denotes random process error and $\boldsymbol{\theta}$ as the model parameters with a prior distribution $P(\boldsymbol{\theta})$.

Training data $\mathcal{D}$ for the BNN model is often from noisy SPICE simulations:

$$\mathcal{D} \; = \; \{\mathbf{y}^{(i)}\}_{i=1}^{N} \; = \; \left\{\mathbf{f}\left(\mathbf{x}^{(i)}, \mathbf{v}^{(i)}\right) + \boldsymbol{\epsilon}^{(i)}\right\}_{i=1}^{N}, \tag{7}$$

in which N is the size of the train data, errors $\boldsymbol{\epsilon}$ can be assumed as independent Gaussian random variables with zero mean and known standard deviation ($\boldsymbol{\sigma}$). For the given data $\mathcal{D}$ and prior knowledge, the likelihood of observation can be calculated as:

$$P(\mathcal{D} \mid \boldsymbol{\theta}) = \prod_{i=1}^{N_y} \prod_{j=1}^{N} \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} \exp\left[-\frac{\left(\tilde{y}_i\left(\boldsymbol{x}^{(j)}, \boldsymbol{v}^{(j)}; \boldsymbol{\theta}\right) - y_i^{(j)}\right)^2}{2\sigma_{i,j}^2} \right]. \quad (8)$$

Using Bayes' theorem, the posterior likelihood can be obtained:

$$P(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \boldsymbol{\theta})\, P(\boldsymbol{\theta})}{P(\mathcal{D})} \propto P(\mathcal{D} \mid \boldsymbol{\theta})\, P(\boldsymbol{\theta}). \quad (9)$$

Since calculation for the probability of data set $P(\mathcal{D})$ is analytically intractable, thus in practice, the posterior $P(\boldsymbol{\theta} \mid \mathcal{D})$ is often not normalized and is expressed with the symbol $\propto$, i.e., an equality up to a constant.

For a fully connected neural network, there are N_l hidden layers in the surrogate model. Its input and output is $\mathbf{x}$ and $\tilde{\mathbf{y}}$, respectively. Its l-th hidden layer is denoted as $\boldsymbol{z}_l$, ($l = 1, 2, \ldots N_l$):

$$\boldsymbol{z}_l = \phi\left(\boldsymbol{w}_{l-1}\boldsymbol{z}_{l-1} + \boldsymbol{b}_{l-1}\right), \quad \tilde{y} \quad = \boldsymbol{w}_{N_l}\boldsymbol{z}_{N_l} + \boldsymbol{b}_{N_l}.$$

Here $\boldsymbol{w}_l$ are the weight matrices and $\boldsymbol{b}_l$ are the bias vectors. Nonlinear activation function $\phi(\cdot)$ is taken as hyperbolic tangent function in this study. The surrogate model parameters $\boldsymbol{\theta}$ are the concatenation of all the weight matrices and bias vectors.

A commonly used prior for $\boldsymbol{\theta}$ in Bayesian Neural Networks is that each component of $\boldsymbol{\theta}$ follows an independent Gaussian distribution with zero means. For each hidden layer ($l = 1, 2 \ldots N_l$), entries of the weight matrices ($\boldsymbol{w}_l$) and bias vectors ($\boldsymbol{b}_l$) are variances $\sigma_{w,l}$ and $\sigma_{b,l}$, respectively. In such case, one can show that as the width of hidden layers goes to infinity with $O(\sqrt{N_l}\sigma_{w,l})$ fixed for $l = 1, 2, \ldots N_l$[6], prior of the surrogate model becomes a Gaussian process. Since parameters $\boldsymbol{\theta}$ in Bayesian Neural Network are drawn from a family of Gaussian variables, it is similar to adding regularization terms to regression models [1]. Therefore, BNN can effectively prevents overfitting in establishing a global surrogate model.

Specific values of model parameters $\boldsymbol{\theta}$ are those maximizing the posterior likelihood $P(\boldsymbol{\theta} \mid \mathcal{D})$. Denoted as $\boldsymbol{\theta}^*$, it can be obtained by numerical optimization for the logarithm of expression (9), using Adam [2] and L-BFGS algorithm [16]. Finally, the posterior circuit performance, $\tilde{\mathbf{y}}(\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}^*)$, can be computed from its surrogate model. Since this step is SPICE-free, one can conduct numerous Monte Carlo on the surrogate model and estimate yield by its sample mean (5).

For an efficient training of the Bayesian Neural Network, Latin Hypercube Sampling and cross-validation [8] are employed. The former is for effective sampling, while the latter determines the number of layers and width in the network structure. In summary, our Bayesian Neural Network for yield is trained as below via Algorithm 1:

Algorithm 1: Yield Estimation in BNN-YEO.

```
Input  : SPICE sample count $N$, number of network layers
         $N_{l,i}$, and width $d_i$, $i = \{1,2,3\}$, number of Monte
         Carlo realizations $M$, design parameters values $\mathbf{x}$
Output : Bayesian Neural Network $\mathcal{M}$, yield estimation
         $\hat{P}_f(\mathbf{x})$
1 Setup.
    • Conduct Latin Hypercube Sampling on the design
      parameter domain and obtain a set of their potential
      values $\{(\mathbf{x}^{(i)}, \mathbf{v}^{(i)})\}_{i=1}^N$.
    • Perform SPICE simulation and obtain the corresponding
      circuit performances that form the training data set $\mathcal{D}$.
2 Cross-validations.
    • Divide the training data set $\mathcal{D}$ into three subsets
      $\mathcal{V}_i$, $i = \{1,2,3\}$. Each is used as a mock validation subset
      and its corresponding complementary set ($\mathcal{U}_i = \mathcal{D} \backslash \mathcal{V}_i$)
      serves as the training subset.
    • Determine the number of hidden layers $N_l$ and width $d$.
      For each of the three training subset, $\mathcal{U}_i$, BNN model is
      trained for different numbers of hidden layers $N_{l,i}$ and
      widths $d_i$. Compute the average error of the surrogate
      model on the corresponding validation subsets $\mathcal{V}_i$. Select
      the number of layers $N_l$ and width $d$ for the minimum
      average error.
3 Model Training.
    Train the BBN model of layers $N_l$ and width $d$ with full data
    set $\mathcal{D}$.
4 Yield Estimation.
    Approximate yield from the surrogate model,
    $\left\{\tilde{\mathbf{y}}(\mathbf{x}, \mathbf{v}^{(j)})\right\}_{j=1}^M$, by performing Monte Carlo sampling of
    process errors.
        $\hat{P}_f(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M} I\left(\mathbf{x}, \mathbf{v}^{(j)}\right).$
```

3.2 Yield optimization with smooth approximation

Having construct a surrogate model to estimate yield with Bayesian Neural Network (5), its optimization can be expedited if gradient information $\partial p_f / \partial \mathbf{x}$ is known. In other words, one needs to find a smooth and analytic approximation to the discontinuous indicator function without compromise on the accuracy of yield estimation. This is achieved by introducing the logistic function $H(\cdot)$:

$$H(x) = \frac{1}{2} + \frac{1}{2}\tanh qx. \quad (10)$$

Here q is a user-defined value that corresponds to the sharp transition at $x = 0$. Figure 1 illustrates logistic function at different values of q. One can see that as q increases from 1 to 50, the new function approaches the indicator function. When q tends to infinity, the limit of logistic function (10) is:

$$H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}. \quad (11)$$

Now for given values of design parameters and process variations, $(\mathbf{x}^{(i)}, \mathbf{v}^{(j)})$, the indicator function can be approximated with

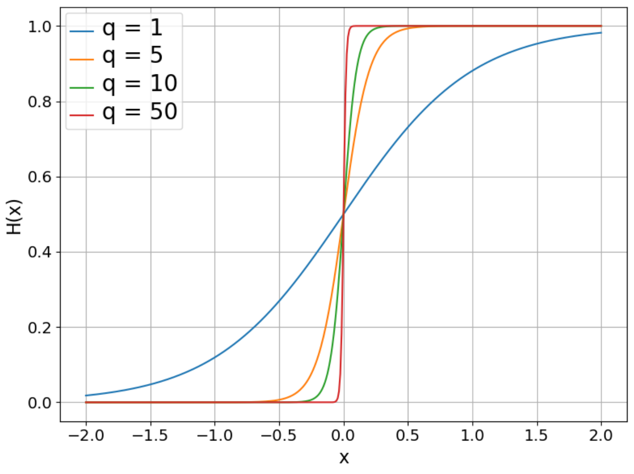

Figure 1: The logistic function (10) **at different values of** q**.**

the logistic function and the surrogate model:

$$I\left(\mathbf{x}^{(i)}, \mathbf{v}^{(j)}\right) \approx \prod_{k=1}^{N_{\mathrm{y}}} H\left[y_k^* - \tilde{y}_k\left(\mathbf{x}^{(i)}, \mathbf{v}^{(j)}\right)\right]. \tag{12}$$

In turn, its estimation of yield becomes:

$$\tilde{P}_{\mathrm{f}}(\mathbf{x}^{(i)}) = \frac{1}{M}\sum_{j=1}^{M}\prod_{k=1}^{N_{\mathrm{y}}} H\left[y_k^* - \tilde{y}_k\left(\mathbf{x}^{(i)}, \mathbf{v}^{(j)}\right)\right]. \tag{13}$$

Now we have a smooth and analytical expression of yield ($\tilde{P}_{\mathrm{f}}$). Its gradient information on design parameters can be easily obtained and utilized for its optimization. In our study, Adam algorithm is employed to solve the following optimization problem:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \ \tilde{P}_{\mathrm{f}}(\mathbf{x}) \quad \text{s.t.} \ \ \mathbf{x} \in \mathcal{X}, \tag{14}$$

It is noted here that our sampling interval of design parameters must be within the domain $\mathbf{x} \in \mathcal{X}$. However, in practice, Adam optimization may lead to values outside such domain and results in optimization failure. To mitigate such risk, it is necessary to check whether the next-step value $\mathbf{x}'$ surpasses its domain after each iteration. If so, it must be pulled back to the closest boundary value. For instance, for a one-dimensional parameter domain, $\mathcal{X} = [-1, 1]$, if the next optimal point x' is calculated as 1.2, it should be restricted to $x' = 1$. For high dimensions N_{x}, the aforementioned examination must cover for each parameter. In summary, our yield optimization can be conducted via Algorithm 2:

4 RESULTS AND DISCUSSION

We examine the performance of our BNN-YEO via 6T-SRAM with comparison to one state-of-the-art method, Bayesian Yield Analysis (BYA) [15]. To be specific, their performances on yield estimation, optimization are studied, respectively. Without notification otherwise, the hyperparameter in the logistic function is taken as $q = 50$. Maximum number of layers and width in the Bayesian Neural Network is set as 7 and 64, respectively. All experiments were conducted on a Linux server with 500GB of memory, one

Algorithm 2: Yield optimization in BNN-YEO

```
Input  : The trained Bayesian neural network $\mathcal{M}$, number
         of optimization iterations $N_{\mathrm{o}}$, and a threshold
         value $\eta$ for accuracy.
Output : Optimal design parameter $\mathbf{x}^*$.
1 for $i = 1$ to $N_{\mathrm{o}}$ do
2     Search for potential values of optimal design parameters,
        $\mathbf{x}^*$, by solving the optimization problem (14) using
        Adam;
3     Check if $\mathbf{x}^*$ is in the defined domain $\mathcal{X}$. If not, round it
        to the closest boundary values;
4     Update and examine the corresponding yield ($\tilde{p}_{\mathrm{f}}$) using
        the surrogate model based on Bayesian neural
        Network (13);
5     If the difference between the new yield $\tilde{p}_{\mathrm{f}}$ and the target
        value $p_{\mathrm{f}}^*$ is smaller than the threshold value $\eta$,
        terminate;
6 end
```

AMD EPYC 7402 24-core processor, and one Nvidia GeForce RTX 3080 Ti graphics card.

4.1 Yield Estimation

The 6T-SRAM is a canonical test case for yield estimation and optimization [14, 15]. Its simplified schematic diagram is shown in Fig. 2. In our study, write delay is taken as the circuit output performance ($\mathbf{y}$), and gate oxide width and length under a 45nm CMOS process for all six transistors are the design parameters of interests ($\mathbf{x}$). In total, there are $N_{\mathrm{x}} = 12$ design parameters for the 6T-SRAM. The circuit is performing within 27 degrees Celsius. The write delay for SRAM is the time taken from the start of the write operation to reaching 90% of the threshold voltage.

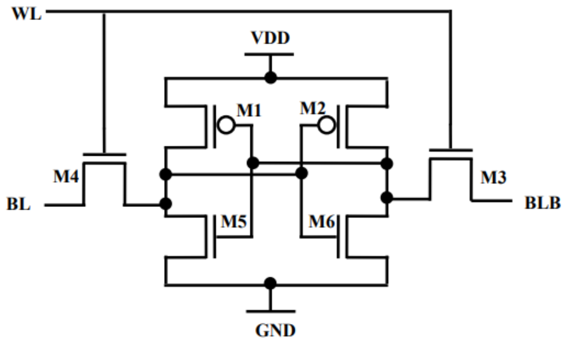

Figure 2: Simplified schematic for 6T-SRAM.

We now examine the accuracy of the BNN-YEO algorithm in yield estimation. The target yield is set as $P_{\mathrm{f}} = 99.9\%$, which is achieved with 50,000 Monte Carlo SPICE simulations. Ten experiments are repeated for BNN-YEO and BYA, respectively, to reduce the impacts of random seeds on yield estimation. The logarithm of their mean absolute errors with respect to target yield are shown in Fig. 3 for various number of SPICE simulations. It is clear that for each method, its estimation accuracy improves as simulations increases. Comparing to BYA, BNN-YEO exhibits a sharper drop in error at the very beginning to almost $O(10^{-4})$ and then follows similar pattern at later stage with more simulations. In general, its mean error for yield estimation is an order of magnitude lower than that of BYA.

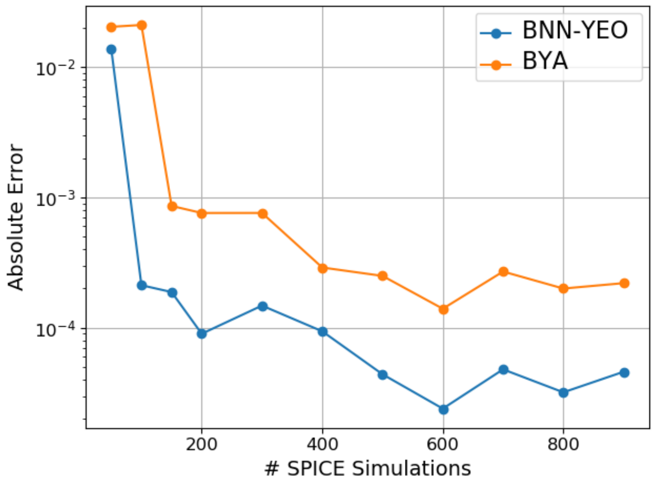

Figure 3: Yield Estimation Absolute Error of 6T-SRAM

For a closer look, we present the absolute errors in yield estimation for both BYA and BNN-YEO in Table 1. Their best, worse and mean results of errors are included for 100, 200, 300 and 500 SPICE simulations, respectively. For the same number of SPICE samples, BNN-YEO generally provides an-order-of-magnitude improvements in yield estimation than BYA. For the same order of accuracy, all three cases of measurements in BNN-YEO reach an absolute error of $O(10^{-4})$ at just 100 simulations and settle to $O(10^{-5})$ with 500 samples. In contrast, it takes BYA 500 SPICE simulations to converge at $O(10^{-4})$. We thus conclude that for yield estimation, BNN-YEO is 5X faster than BYA and 100x more efficient than standard Monte Carlo in terms of SPICE simulations required.

4.2 Yield Optimization

Let us now investigate the effectiveness of BNN-YEO in yield optimization. For comparison, results from BYA method are also included in Table 2. The best, worst and mean results of ten experiments from each method are obtained using 500, 5000 and 10,000 SPICE simulations, respectively. One can see that both approaches improve their optimizations with more simulations involved. For BNN-YEO, all its three cases achieve near-optimal values of 99.98% using just 500 samples. Though they can be further elevated with 5000 and 10,000 simulations, the improvements of 0.01% are somehow limited. This is because that once BNN-YEO completes model training in the yield estimation phase, its optimization is performed on its surrogate model ($\hat{P}_{\mathrm{f}}$) and thus no more simulations are required. For BYA, its optimization performances at small number of SPICE simulations are at best mediocre. With 500 samples, its yield optimization results are below 50%. It increases to < 80% for 5000 simulations and reaches 99% using 10,000 samples. Such performance may be due to its adaptive nature with a continuous inflow of data. It would be discussed and analyzed in details in subsequent subsection. After all, BNN-YEO shows a 20x improvements that state-of-the-art method in SPICE simulations costs.

4.3 Computational Cost Analysis

We tabulate the computational costs for Monte Carlo, BNN-YEO and BYA, respectively, In Table 3. It takes roughly 8.65 seconds to run one SPICE simulations on the current computer environment. For Monte Carlo, 50,000 simulations are needed to converge to the target yield value of 99.9%. For BNN-YEO, its surrogate model requires 500 simulations to train and takes only 0.0032 seconds for one execution. This is 2700x faster than SPICE simulations. Such improvement significantly expedites the yield prediction process in BNN-YEO. For BYA, its Gaussian Process model is also cheaper than SPICE simulation, e.g. 0.126 seconds per run. But its adaptive update on the surrogate model in optimization often entails a continuous inflow of SPICE simulations, which eats up its gain in model speed.

Table 3: Computational Costs of Monte Carlo (MC), BNN-YEO and BYA.

	MC	BYA	BNN-YEO
Simulation Speed	8.65s	0.126s	**0.0032s**
# Simulation	50,000	10,000	**500**

For better understanding, we now conduct detailed analysis on the computational expenses. This is done through a simple white-box model as circuit response:

$$f(\mathbf{x}, \mathbf{v}) = \sum_{i=1}^{N_{\mathrm{x}}} (x_i + v_i)^2, \qquad \mathbf{x} \in [-1, 1]^{N_{\mathrm{x}}}. \tag{15}$$

Here Success is defined if $f(\mathbf{x}, \mathbf{v}) \leq N_{\mathrm{x}}$. In other words, the optimal design parameters values for highest yield are: $\mathbf{x}^* = [0, 0, \cdots, 0]^{\mathrm{T}}$.

Table 4 presents the optimized yield results from BYA and BNN-YEO for three cases, namely, $N_{\mathrm{x}} = 4$, $N_{\mathrm{x}} = 10$ and $N_{\mathrm{x}} = 18$. For both methods, 100, 300 and 500 simulations of (15) are used to construct the surrogate models in each case. Their optimizations all initiate at the same values, $\mathbf{x}_0 = [-1, -1, \cdots, -1]^T$. The target yield values are obtained via 100,000 simulations and are also listed. To reduce the impacts of random seeds on training data, 10 experiments are repeated for each method at each case. Their best, worst and mean performance are also included. It is evident in Table 4, performance of BYA deteriorates as parameter dimension increases. At $N_{\mathrm{x}} = 10$ and $N_{\mathrm{x}} = 18$, BYA cannot find the optimal solution using its surrogate model from 500 samples of (15), but needs 1000 and 3000 samples, respectively. On the other hand, BNN-YEO is capable to capture target yield with only 500 simulations. Such contrast in performance lies in the selection of surrogate models. BYA employs a Gaussian process to construct a surrogate model, which is prone to overfitting and has certain limitations. It can provide a good estimate of optimal value should the iteration value be included in its

Table 1: Yield analysis absolute error for 6T-SRAM

		#Simulation=100		#Simulation=200		#Simulation=300		# Simulation=500	
Test		BYA	BNN-YEO	BYA	BNN-YEO	BYA	BNN-YEO	BYA	BNN-YEO
6T-SRAM	Best	1.95E-03	**1.38E-04**	**3.78E-05**	6.77E-05	4.11E-05	**3.26E-05**	1.75E-04	**2.62E-05**
(Yield=99.9 %)	Mean	2.13E-02	**2.12E-04**	7.59E-04	**9.00E-05**	7.61E-04	**9.42E-05**	2.93E-04	**4.45E-05**
	Worst	4.56E-02	**5.76E-04**	1.52E-03	**1.35E-04**	9.63E-04	**1.20E-04**	5.12E-04	**7.40E-05**

Table 2: Yield optimization for 6T-SRAM

		#Simulation=500		#Simulation=5000		#Simulation=10,000	
Test		BYA	BNN-YEO	BYA	BNN-YEO	BYA	BNN-YEO
	Best	45.45 %	**99.98%**	75.59 %	**99.99%**	99.88 %	**99.99%**
6T-SRAM	Mean	42.34 %	**99.92%**	67.54%	**99.96%**	99.68 %	**99.98%**
	Worst	38.59 %	**99.87%**	61.37%	**99.94%**	98.52 %	**99.98%**

training set. Otherwise, incorrect estimation may ensue and Gaussian process has to be continuously updated through resampling of new SPICE simulations. Our Bayesian neural network is instead sampled from a family of Gaussian distributions for optimal model parameters. Therefore, it can construct a global surrogate model avoiding model distortion and significantly reduce simulation cost.

Table 4: Yield Estimation and Optimization for Case (15).

Test		BYA	BNN-YEO
$N_x = 4$	Best	59.41%	**59.42%**
(Target yield 59.44%)	Worst	59.27%	**59.37%**
	Mean	59.35%	**59.39%**
$N_x = 10$	Best	50.61%	**55.93%**
(Target yield 55.96%)	Worst	46.41%	**55.87%**
	Mean	48.24%	**55.89%**
$N_x = 18$	Best	44.88%	**54.39%**
(Target yield 54.44%)	Worst	35.87%	**54.06%**
	Mean	40.90%	**54.24%**

5 CONCLUSION

In this paper, a novel framework (BNN-YEO) for yield estimation and optimization is proposed. Based on Bayesian Neural Network, it constructs a surrogate model with prior information and replaces SPICE simulations in yield optimization process. Its verification on 6T-SRAM led to the following conclusions:

- The new method provides accurate results in yield estimation and optimization with much fewer SPICE simulations than Monte Carlo and state-of-the-art BYA method.
- Its surrogate model is effective for optimization and at least 2000 times faster than SPICE simulation.
- The globally optimal parameters of the surrogate model are obtained by solving from a family of Gaussian distributions, thereby helping alleviate the common risk of overfitting in many surrogate models.
- Performance of the BNN-YEO approach may be further enhanced if its optimal structure of the Bayesian Neural Network can be found, possibly by another form of machine learning such as reinforced learning.

6 ACKNOWLEDGMENTS

This work is supported by The National Key Research and Development Program of China under Grants 2022YFB4400402.

REFERENCES

[1] Peter J Bickel, Bo Li, Alexandre B Tsybakov, Sara A van de Geer, Bin Yu, Teófilo Valdés, Carlos Rivero, Jianqing Fan, and Aad van der Vaart. 2006. Regularization in statistics. *Test* 15 (2006), 271–344.

[2] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[3] Igor Kononenko. 1989. Bayesian neural networks. *Biol. Cybern.* 61, 5 (1989), 361–370.

[4] Jing Li and Dongbin Xiu. 2010. Evaluation of failure probability via surrogate models. *J. Comput. Phys* 229, 23 (2010), 8966–8980.

[5] Bo Liu, Francisco V Fernández, and Georges GE Gielen. 2011. Efficient and accurate statistical analog yield optimization and variation-aware circuit sizing based on computational intelligence techniques. *IEEE TCAD* 30, 6 (2011), 793–805.

[6] Radford M Neal. 2012. *Bayesian learning for neural networks.* Vol. 118. Springer Science & Business Media.

[7] Masood Qazi, Mehul Tikekar, Lara Dolecek, Devavrat Shah, and Anantha Chandrakasan. 2010. Loop flattening & spherical sampling: Highly efficient model reduction techniques for SRAM yield analysis. In *Proc. of DATE 2010.* IEEE, 801–806.

[8] Xiao Shi, Hao Yan, Qiancun Huang, Jiajia Zhang, Longxing Shi, and Lei He. 2019. Meta-model based high-dimensional yield analysis using low-rank tensor approximation. In *Proc. of the 56th DAC 2019.* 1–6.

[9] Amith Singhee, Sonia Singhal, and Rob A Rutenbar. 2008. Practical, fast Monte Carlo statistical static timing analysis: Why and how. In *Proc. of ICCAD 2008.* IEEE, 190–195.

[10] Michael Stein. 1987. Large sample properties of simulations using Latin hypercube sampling. *Technometrics* 29 (1987), 143–151.

[11] Mengshuo Wang, Wenlong Lv, Fan Yang, Changhao Yan, Wei Cai, Dian Zhou, and Xuan Zeng. 2017. Efficient yield optimization for analog and SRAM circuits via Gaussian process regression and adaptive yield estimation. *IEEE TCAD* 37, 10 (2017), 1929–1942.

[12] Liu Yang, Xuhui Meng, and George Em Karniadakis. 2021. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *J. Comput. Phys* 425 (2021), 109913.

[13] Jian Yao, Zuochang Ye, and Yan Wang. 2012. Statistical analysis of process variations in RF/mm-wave circuits with on-the-fly passive macro-modeling. *IEEE Trans. Microw. Theory Techn.* 61, 2 (2012), 727–735.

[14] Jian Yao, Zuochang Ye, and Yan Wang. 2014. An efficient SRAM yield analysis and optimization method with adaptive online surrogate modeling. *IEEE TVLSI* 23, 7 (2014), 1245–1253.

[15] Shuo Yin, Xiang Jin, Linxu Shi, Kang Wang, and Wei W Xing. 2022. Efficient bayesian yield analysis and optimization with active learning. In *Proc. of the 59th DAC 2022.* 1195–1200.

[16] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *TOMS* 23, 4 (1997), 550–560.