UC Santa Barbara

UC Santa Barbara Electronic Theses and Dissertations

Title

Machine Learning Techniques for Rare Failure Detection in Analog and Mixed-Signal Verification and Test

Permalink

https://escholarship.org/uc/item/34w6s777

Author

Hu, Hanbin

Publication Date

2021

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

University of California
Santa Barbara

Machine Learning Techniques for Rare Failure Detection in Analog and Mixed-Signal Verification and Test

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Hanbin Hu

Committee in charge:

Professor Peng Li, Chair
Professor Li-C. Wang
Professor Upamanyu Madhow
Doctor Rui Wang

September 2021

The Dissertation of Hanbin Hu is approved.

Professor Li-C. Wang

Professor Upamanyu Madhow

Doctor Rui Wang

Professor Peng Li, Committee Chair

August 2021

Machine Learning Techniques for Rare Failure Detection in Analog and Mixed-Signal Verification and Test

Copyright © 2021

by

Hanbin Hu

To my parents.

Acknowledgements

This dissertation is based upon work supported by the National Science Foundation under Grant No. 1956313 and by the Semiconductor Research Corporation (SRC) through Texas Analog Center of Excellence at the University of Texas at Dallas (Task IDs: 2712.032, 2810.031, and 2810.043).

First of all, I would like to express my deep gratitude to my advisor, Professor Peng Li, who has been continuously supporting and guiding me throughout my PhD degree at both Texas A&M Univeristy and University of California, Santa Barbara in the past five years. He not only advises me on particular research projects with his comprehensive expertise and insightful ideas, but also exemplifies himself as a passionate ambitious hardworking researcher, enlighting me how to dream big, aim high and work hard.

Besides, I would also like to thank Professor Li-C. Wang, Professor Upamanyu Madhow, and Rui Wang for serving on my PhD committee and providing invaluable suggestion during my qualifying exam. I am also grateful to Professor Jianhua Z. Huang at Texas A&M University for introducing me to novel research ideas back. I would also like to thank Chen He, Nguyen Nguyen from NXP Semiconductors, Chris M. Cooper, and Lakshmanan Balasubramanian from Texas Instruments, for providing industrial perspective and supporting me in the SRC projects. In addition, I would like to extend my thanks to Joel R. Phillips, Arto Sandroos, Ilya Yusim from Cadence Design Systems, Tuna B. Tarim, Minas Hambardzumyan from Texas Instruments, Bryan Klingner, Tyler Hovanec, Kristen Lurie from Google for their warm host and great mentoring leading to fruitful summers during my internships.

Moreover, I really appreciate all the group members and my personal friends on the path of my 5-year PhD life. I truly value your company and time along the way, encouraging me towards a better life. The list goes to, but no limited to, Wenrui Zhang, Yingyezhe (Jimmy) Jin, Yu Liu, Xin Zhan, Myung Seok Shim, Karthik Somayaji N.S., Jeonjun Lee, Yu Wang, Jianhao Chen, Changqing Xu, Ya (Tony) Wang, Qingran Zheng, Tingrui Zhang, Yukun He, Guanru Wang, Honghuang Lin, Qian Wang, Zhiyu (Albert) Zeng, Zhiyuan Zheng, Hanzi Mao, Yixiao Feng, Zhuotong Chen, Zichang He, Chunfeng Cui, Boyuan Gong, Binghan Li, Kai He, Ziwei Xuan, Daimeng Li, Jicheng Lu, Zhide Wang, Nan Du, Jiangyue Gong, Yongnan Sun, Bicheng Ying, Kun Yuan, Yibo Lin, Wuxi Li, Shuwen Deng, Lihao Zou, Laung-Terng Wang, Xingchen Wan and Xiaoyu Guan.

Last but not the least, I would love to give my endless thanks to my parents for their unreserved love and deep faith in me. Without their support, I would never reach this life point. I sincerely dedicate this dissertation to my beloved parents.

Curriculum Vitæ
Hanbin Hu

Education

09/2019-08/2021	Ph.D. in Electrical and Computer Engineering (Expected), University of California, Santa Barbara.
08/2016-08/2019	Ph.D. in Computer Engineering, Texas A&M University. (Transferred to University of California, Santa Barbara)
09/2013-03/2016	M.S. in Electronic Science and Technology, Shanghai Jiao Tong University.
09/2009-06/2013	B.S. in Microelectronics, Shanghai Jiao Tong University.

Publications

[**ITC'21**] **Hanbin Hu**, Chen He, and Peng Li, "Semi-supervised Wafer Map Pattern Recognition using Domain-Specific Data Augmentation and Contrastive Learning", Proceedings of the IEEE International Test Conference, accepted, 2021.

[**DAC'21**] Myung Seok Shim, **Hanbin Hu**, and Peng Li, "Reversible Gating Architecture for Rare Failure Detection of Analog and Mixed-Signal Circuits", Proceedings of the ACM/IEEE Design Automation Conference, accepted, 2021.

[**DAC'21**] Karthik Somayaji N.S., **Hanbin Hu**, and Peng Li, "Prioritized Reinforcement Learning for Analog Circuit Optimization With Design Knowledge", Proceedings of the ACM/IEEE Design Automation Conference, accepted, 2021.

[**EDTM'21**] Chen He, **Hanbin Hu**, and Peng Li, "Applications for Machine Learning in Semiconductor Manufacturing and Test", Proceedings of the IEEE Electron Devices Technology and Manufacturing Conference, pp. 1-3, 2021. (**Invited Paper**)

[**ITC'20**] **Hanbin Hu**, Nguyen Nguyen, Chen He, and Peng Li, "Advanced Outlier Detection Using Unsupervised Learning for Screening Potential Customer Returns", Proceedings of the IEEE International Test Conference, pp. 1-10, 2020.

[**Preprint**] **Hanbin Hu**, Mit Shah, Jianhua Z. Huang, and Peng Li, "Global Adversarial Attacks for Assessing Deep Learning Robustness," arXiv:1906.07920 [cs.LG], 2019.

[**DAC'19**] **Hanbin Hu**, Peng Li, and Jianhua Z. Huang, "Enabling High-Dimensional Bayesian Optimization for Efficient Failure Detection of Analog and Mixed-Signal Circuits", Proceedings of the ACM/IEEE Design Automation Conference, no. 17, pp. 1-6, 2019.

[**ICCAD'18**] **Hanbin Hu**, Peng Li, and Jianhua Z. Huang, "Parallelizable Bayesian Optimization for Analog and Mixed-Signal Rare Failure Detection in High Coverage", Proceedings of the IEEE/ACM International Conference on Computer-Aided Design, no. 98, pp. 1-8, 2018. (**Best Paper Award Nomination**)

[**DAC'18**] **Hanbin Hu**, Qingran Zheng, Ya Wang, and Peng Li, "HFMV: Hybridizing Formal Methods and Machine Learning for Verfication of Analog and Mixed-Signal Circuits", Proceedings of the ACM/ESDA/IEEE Design Automation Conference, no. 95, pp. 1-6, 2018.

[**TODAES'17**] Guoyong Shi, **Hanbin Hu**, and Shuwen Deng, "Topological Approach to Automatic Symbolic Macromodel Generation for Analog Integrated Circuits," ACM Transactions on Design Automation of Electronic Systems, vol. 22, no. 3, pp. 1-25, 2017.

[**ISCAS'15**] **Hanbin Hu**, Guoyong Shi, Andy Tai, and Frank Lee, "Topological Symbolic Simplification for Analog Design," Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 2644-2647, 2015.

[**ASICON'13**] **Hanbin Hu**, Guoyong Shi, and Yan Zhu, "Incremental Symbolic Construction for Topological Modeling of Analog Circuits," Proceedings of the IEEE International Conference on ASIC, pp. 1-4, 2013.

Workshops

[**TECHCON'20**] **Hanbin Hu**, Nguyen Nguyen, Chen He, and Peng Li, "Unsupervised Prediction of Rare Post-Silicon Defects using Neural Networks", SRC TECHCON Conference, 2020.

[**ICCAD'19**] **Hanbin Hu** and Peng Li, "Efficient Analog/Mixed-Signal Verification Using Bayesian Optimization in High Dimensional Space", International Workshop on Design Automation for Analog and Mixed-Signal Circuits at the IEEE/ACM International Conference on Computer-Aided Design, 2019.

[**TECHCON'19**] **Hanbin Hu**, Yukun He, and Peng Li, "Assessment of Machine Learning Robustness for Analog and Mixed Signal Verification", SRC TECHCON Conference, 2019.

[**ICCAD'18**] **Hanbin Hu**, Peng Li, and Jianhua Z. Huang, "Parallelizable Bayesian Optimization for Analog and Mixed-Signal Rare Failure Detection with High Coverage", International Workshop on Design Automation for Analog and Mixed-Signal Circuit at the IEEE/ACM International Conference on Computer-Aided Design, 2018.

[**TECHCON'18**] **Hanbin Hu** and Peng Li, "A Hybrid Verification Framework for Analog and Mixed-Signal Circuits", SRC TECHCON Conference, 2018.

Professional Experience

06/2020-09/2020	Software Engineering Intern, Google.
05/2019-08/2019	Software Engineer Intern, Texas Instruments.
05/2018-08/2018	Graduate Intern, Cadence Design Systems.
07/2015-09/2015	Software Engineer Intern, Synopsys.
07/2014-09/2014	Quality Assurance Engineer Intern, Synopsys.

09/2019-08/2021	Research Assistant, University of California, Santa Barbara.
08/2016-05/2019	Research Assistant, Texas A&M University.
09/2013-03/2016	Research Assistant, Shanghai Jiao Tong University.
03/2015-05/2015	Teaching Assistant, Shanghai Jiao Tong University.

Service

Reviewer, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2019-2021.

External Reviewer, ACM/IEEE Design Automation Conference (DAC), 2020.

External Reviewer, ACM/IEEE Design Automation Conference (DAC), 2019.

Reviewer, IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2020.

Awards

06/2019	Richard A. Newton Young Student Fellow at DAC 2019
10/2018	IEEE/ACM William J. Mccalla ICCAD Best Paper Award Nomination

Abstract

Machine Learning Techniques for Rare Failure Detection in Analog and Mixed-Signal Verification and Test

by

Hanbin Hu

Due to inherent complex behaviors and stringent requirements in analog and mixed-signal (AMS) systems, verification and testing become key bottlenecks in the product development cycle. Rare failure detection in a high-dimensional parameter space using minimal expensive simulation/measurement data is a major challenge.

For rare failure detection in the verification flow, this dissertation proposes to put machine learning models, that mimic the circuit behavior, under verification, which greatly relaxes the simulation/measurement requirements and improves the verification efficiency.

We first present a hybrid formal/machine-learning verification technique (HFMV) to combine the best of the two worlds. HFMV adds formalism on the top of a probabilistic learning model while providing a sense of coverage for extremely rare failure detection. On the other hand, we also study Bayesian optimization (BO) based approaches to the challenging problem of verifying AMS circuits with stringent low failure requirements. We simultaneously leverage multiple optimized acquisition functions to explore varying degrees of balancing between exploitation and exploration. Furthermore, this dissertation proposes a BO framework under high dimensional space to further improve the verification efficiency. Two techniques are explored here: 1) random embedding to linearly embed input into a low dimensional space and 2) sensor fusion networks to identify important nonlinear features transformed by reversible neural networks. The proposed approaches are very effective in finding very rare failures in high dimensional space which existing statistical techniques can miss.

On the subject of AMS testing, this dissertation proposes to utilize self-supervised learning methods to detect extremely rare customer failure. First, we study a transformation-based self-labeling technique to reliably screen out rare customer return defects. The normality score to an unseen input data is the goodness of the multi-class classification model trained by self-labeled data via a set of reversible transformations. Furthermore, this dissertation suggests a contrastive learning framework for semi-supervised learning and prediction of wafer map patterns. Contrastive learning is applied for the unsupervised encoder representation learning supported by augmented data generated by different transformations (views) of wafer maps. Experimental results demonstrate that the self-supervised learning framework greatly improves test accuracy compared to traditional supervised methods.

Contents

Chapter 1

Introduction

Analog and mixed-signal (AMS) systems are prevailing in many popular applications nowadays. For example, wireless communication or imaging circuits may be used in automobile applications [1]; biomedical circuits may utilize ADCs and DACs for signal collection and stimulation [2]; internet of things application may require analog circuit for energy harvesting or sensing [3]. One important question to be addressed in this dissertation is how the AMS circuit robustness can be verified and tested in an efficient and reliable manner.

1.1 AMS Verification and Test

The verification in the digital world has already been mature for decades. With the popular SystemVerilog [4] and universal verification methodolgy (UVM) [5] in the verification society, and automatic test pattern generation (ATPG) [6, 7] in the testing field, the academia and industry have enabled fast automatic testbench generation for digital systems, greatly boosting the corresponding system robustness and reliability.

However, comparing to the digital counterparts, AMS verification and test are still under-development. In particular, analog signals are usually more complex than digital signals due the continuous nature, which brings in a parameter space under verification or test infinitely large, leading to curse of dimensionality. Moreover, unlike the digital circuits which can be characterized by some simple Boolean logics, the behavior and consideration of AMS circuits and systems are usually highly complex, nonlinear, and customized, which is almost infeasible to design a common description language for different designs. All these factors make AMS verification and test extremely hard.

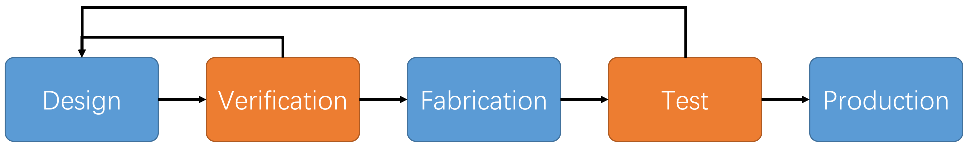


Figure 1.1: Analog & mixed-signal design flow.

In addition, we can notice all the safety-critical applications require high requirements in terms of the AMS circuit/system robustness. Hence, the corresponding AMS verification and test become increasingly critical and demand high accuracy and efficiency. As shown in Fig. 1.1, the verification and test are two major steps during the design flow to check the circuit robustness under various process-voltage-temperature (PVT) variations. Therefore, the efficiency, accuracy and coverage of significant can dramatically impact the turnaround time of the AMS design flow.

1.2 Rare Failure Challenges for AMS Verification and Test

With increasing design complexity and rising robustness requirement, AMS verification and test manifest themselves as the key bottlenecks [8] in the design flow. For instance, automotive electronics may have an extremely low failure rate requirement, e.g.

1 DPPM (defective parts per million) or less, making failure detection and design verification/testing very challenging. Detecting even a single failure for circuits that are designed to be extremely robust with typical simulation/testing budgets during design time is a completely nontrivial problem. Under this case, the rare failure detection problem (e.g. finding the first failure) is a fundamental and challenging problem to be addressed.

1.2.1 AMS Verification Challenges

For the verification side, it is widely known that there are two major trends in AMS verification: formal verification, and simulation-based verification.

Formal verification is appealing as it provides a provable "yes/no" answer w.r.t the specifications under check. However, performing formal verification directly on top of a detailed low-level (nonlinear) SPICE circuit netlist or model (e.g. a DAE or hybrid automation) severely limits scalability. To date, formal techniques are only feasible for small analog blocks described by idealistic models, falling behind the practical industrial needs [9, 10, 11]. In addition, as formal methods require strict mathematical proof, the methods are usually not versatile for different circuits and systems.

On the other hand, the simulation-based verification heavily leverages sampling. If we consider failure rate to be less than 1 DPPM, using simple Monte Carlo sampling, at least 1 million samples should be collected to obtain a single failure, which is infeasible in most cases. In recent years, researchers explore smart statistical sampling techniques (e.g. [12, 13, 14]) to accelerate the sampling process. These methods look promising in term of fast failure rate estimation, however, when it comes to extremely-rare failure detection, they usually neglect one fundamental question whether there exists any failure (which can be extremely rare) at all in a bounded parameter space. These extreme rarity makes most sampling techniques hard to find any failures. If no first failure is detected, it is hard to find a good sampling distribution close to the failure region for importance sampling techniques to use. When failure is extremely rare, failure detection itself becomes a much more fundamental problem than failure rate estimation for design verification. Therefore, some state-of-the-art smart statistical sampling techniques (e.g. [12, 13, 14]) are not specifically targeted for providing a guarantee for rare failure detection and can miss rare failures especially under a limited sampling data budget.

1.2.2 AMS Test Challenges

Screening out all potentially defective parts before shipping to customers is crucial for minimizing the risk of the products failing in the customer line or field [15]. Typically, test process consists of wafer probe test, burn-in test with packaged parts, and final test, as shown in Fig. 1.2. During both the wafer probe and final test phases, a large number of parametric tests are performed to extract the part performance values. Outlier detection is applied using the results from the parametric tests to identify abnormal parts. Such parametric tests and outlier detection are especially important to test analog and mixed-signal circuits, as a defect in those circuits is more likely to cause a parametric shift rather than a hard functional failure.

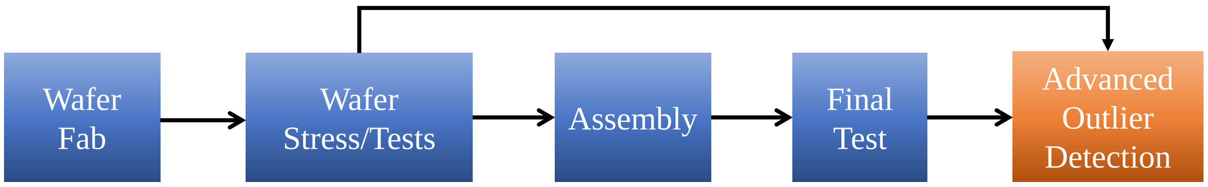
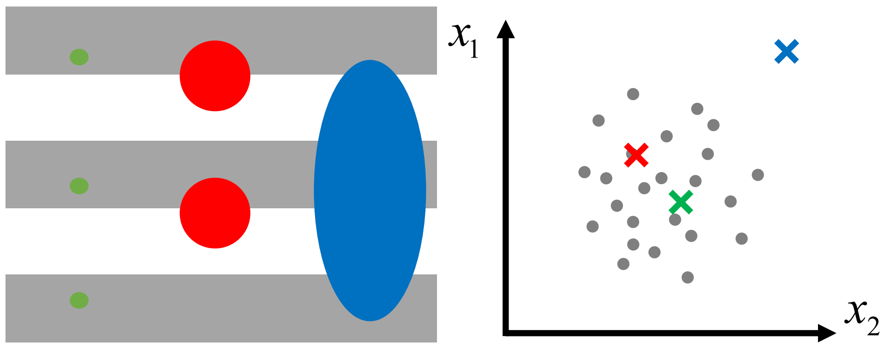


Figure 1.2: Post-silicon production test flow.

The typical measure of design quality is DPPM, i.e., number of Defective Parts which fail after shipping to the customer (also known as customer failures) Per Million parts shipped. The target quality level for chips that are deployed in mission-critical applications, e.g. automotive electrics, can be very stringent. As the complexity of semiconductor products keeps increasing, it is increasingly challenging for post-silicon testing to screen out all defects without any escape to customers. While it is critical to learn from extremely rare (in DPPM level) customer failures to improve outlier defect detection to reach the Zero Defect (or zero customer failure) quality, there remain major challenges in this learning process. **1)** Customer failures are rare, as they are escapes from a rather comprehensive test process. As nearly-all of the defective parts have been screened out, data on customer failures are extremely scarce, typically at several parts per million (PPM) level at most. **2)** It can be difficult to identify a subset of parametric tests that expose potential failures. Alternatively, outlier detection may be performed over a high-dimensional input feature space formed by a large number of parametric test measurements. **3)** It is difficult to catch latent reliability faults by comparing with the normal chip data distribution, leading to defect escapes.



(a) Defect illustration (b) Defect distribution

Figure 1.3: Illustration of potential defects types.

As one illustrative example, Fig. 1.3(a) shows different types of defects around parallel wires (in gray) on a particular metal layer, and Fig. 1.3(b) gives the corresponding distribution of two parametric test results. The small green defects have no impact on the circuit performance, which also locate in the center of the distribution of the parametric test data. The large blue defect can cause catastrophic short-circuit fault and are typically far away from the center of the distribution, making it easy to screen it out. The red defects, the so-called latent reliability defects, might not be caught by post-silicon testing, however, can evolve into early life failures in the customer field due to aging. Such latent defects are extremely hard to be detected unless a large combination of different parametric tests is analyzed in detail to distinguish them from normal circuit behaviors or inconsequential defects during post-silicon testing.

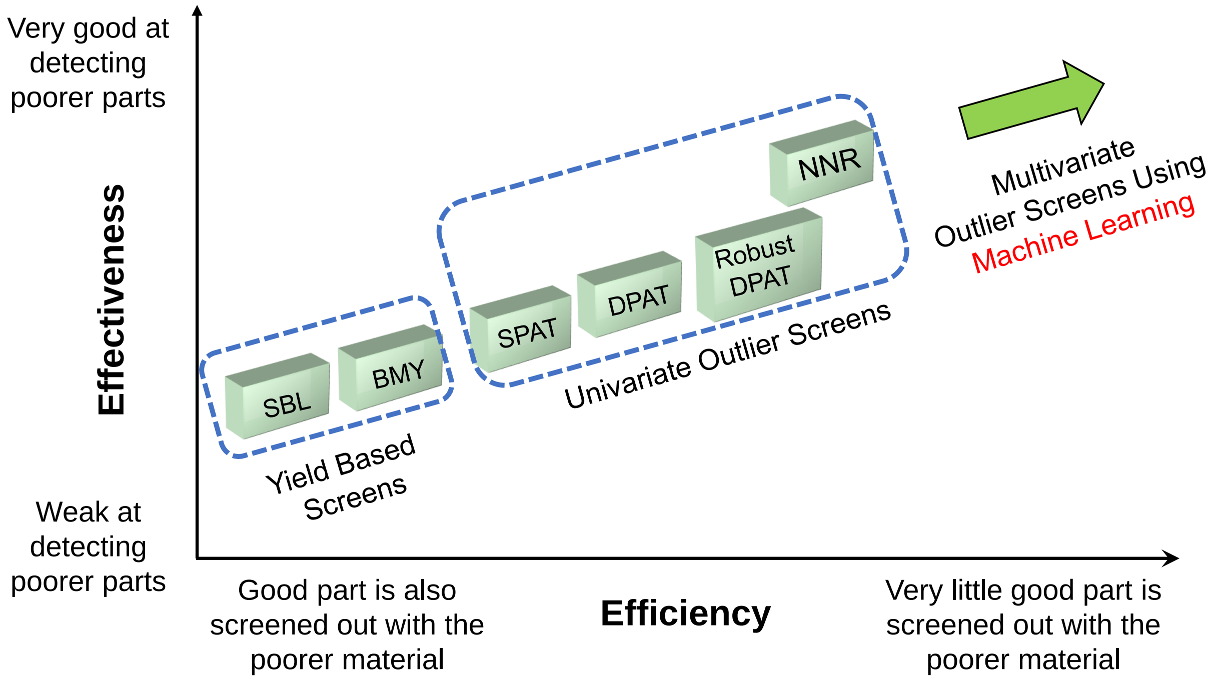

Very good at detecting poorer parts
Effectiveness
Weak at detecting poorer parts
SBL
BMY
Yield Based Screens
SPAT
DPAT
Robust DPAT
NNR
Univariate Outlier Screens
Multivariate Outlier Screens Using Machine Learning
Good part is also screened out with the poorer material
Efficiency
Very little good part is screened out with the poorer material

Figure 1.4: Advanced outlier detection evolution.

Due to the scarcity of failure data, typically there are insufficient defective samples to validate *unsupervised* models for outlier detection [16]. As shown in Fig. 1.4, most early works in this field applied yield based screening strategies like statistical bin limits (SBL) and below minimum yield (BMY) [17, 18]. In order to more efficiently and effectively screen out outlier parts and reduce the corresponding yield loss, several univariate outlier detection methods such as static/dynamic part average test (S/DPAT) [19, 20, 21, 22], and nearest neighbor residual (NNR) [23, 24, 25], and location average [23, 26] were proposed and are commonly employed in the industry. Multivariate outlier detection methods were also proposed for screening rare defects and customer returns [27, 28]. However, with recent advances in the machine learning field, it remains interesting to study how advanced machine learning techniques can be helpful for outlier detection in test.

1.3 Dissertation Contributions

This dissertation mainly proposes two directions for efficient and reliable failure detection for AMS verification and test using machine learning techniques: machine-learning-based AMS verification and self-supervised test framework. On the verification side, we put a machine learning model, mimicking the AMS circuits and systems behavior, under verification, instead of the original system, which greatly improves verification efficiency. On the test side, the self-supervised learning is studied and embedded into the test framework for rare failure detection.

1.3.1 Machine Learning Based AMS Verification

Given a variational parameter space Ω, the analog verification attempts finds out the existence of certain points $\mathbf{x} \in \Omega$, whose performance value $y(\mathbf{x})$ given by simulation or measurements fails the specification T. Without loss of generality, the problem can be formulated as follows.

$$\exists \mathbf{x} \in \Omega,\ s.t.\ y(\mathbf{x}) \leq T \tag{1.1}$$

One of the major challenges of this problem is the huge amount of the simulation for verification purpose, especially in the rare failure detection. As shown in Fig. 1.5, instead of directly verifying the AMS circuit itself, machine-learning-based analog verification proposes to train a machine learning model $f(\mathbf{x})$ with a small number of simulation

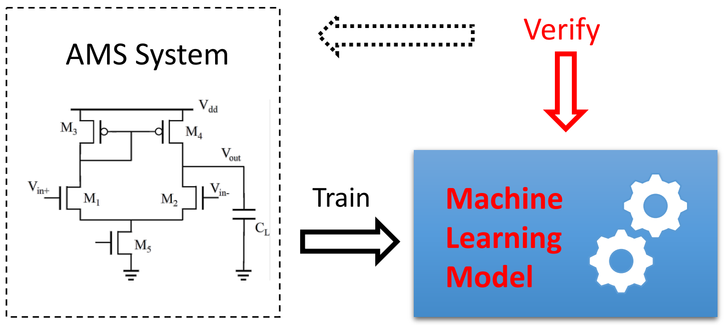


Figure 1.5: Machine learning based AMS verification illustration.

samples, and then verifies the surrogate machine learning model. Assuming the machine learning model is robust enough to mimic the AMS system behavior, we can see that the costly simulation is replaced with the efficient model evaluation. Meanwhile, unlike the black-box simulation $y(\mathbf{x})$ without a closed form, machine learning model $f(\mathbf{x})$ is usually a white-box function, all kinds of operations like derivative computation can be executed more transparently and efficiently, which facilitate the verification algorithm development a lot.

Despite the benefits of machine learning based verification, one question to be answered is how much we can trust the model to be reliable enough to mimic the behavior model, which can be formulated as follows.

$$\exists \mathbf{x} \in \Omega, \ \ s.t. \ |y(\mathbf{x}) - f(\mathbf{x})| \geq \delta \tag{1.2}$$

If the difference between the simulation and the machine learning model is too large, it is unlikely to solve (1.1) robustly. The lack of simulation samples and highly-nonlinear behavior of the certain AMS circuit may even deteriorate the situation more, making the verification results untrustworthy. Actually, (1.2) can be regarded as a verification problem as well. Without know the simulation function $y(\mathbf{x})$, the reliability of the machine learning model is hard to be directly measured. A fully study about the robustness of machine learning model is required to guarantee the reliability of the verification results.

1.3.2 Self-supervised Testing Framework

Unlike the verification task, when using machine learning techniques for testing, one critical issue encountered in reality is lack of the labeled data. It usually requires huge manual effort to correctly label the sample as pass/fail or put them into the right test bins. In addition to the manual effort, due to the failure scarcity, there may exist no failures at all during the model training phase, which results in an highly imbalanced dataset.

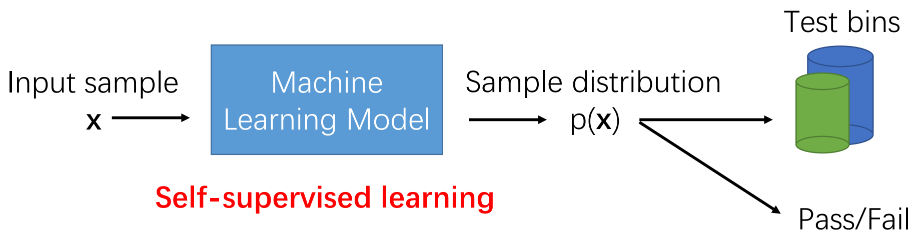


Figure 1.6: Self-supervised testing framework illustration.

Based on the recent development in self-supervised learning [29, 30], we relax the need of sample labels, and only requires unlabeled test samples to learn a distribution $p(\mathbf{x})$ of the die under test after fabrication, as shown in Fig. 1.6. Using self-supervised techniques, the distribution of the fabrication data or the compressed representation can well captured without providing any data labels. After the distribution is learnt, based on the probability distribution (or score), we can then assign the input sample as a pass/fail signal or put it into certain test bins.

1.4 Dissertation Organizations

The dissertation is organized as follows.

Chapter 2, Chapter 3, and Chapter 4 mainly focus on the machine learning based AMS verification. In particular, Chapter 2 hybridize the formal methods and machine learning techniques for efficient verification with reliable guarantees. To further improve the scalability of the proposed methods, Chapter 3 applies Bayesian optimization techniques by elegantly considering the balance between failure detection and model improvement, resulting in more effective and efficient rare failure detection. Furthermore, the efficiency of Bayesian optimization under high dimensional space is discussed and improved in Chapter 4. Moreover, in Chapter 5, we explore the model robustness issue, which is critical for machine learning based AMS verification, by proposing a global adversarial attack method to expose the model vulnerability.

For the self-supervised learning framework, Chapter 6 utilizes a transform-based classification to acquire the abnormal score of test samples in parametric test, capturing extremely-rare customer return failures. In addition, contrastive learning concept is explored in Chapter 7 for wafer pattern recognition, where data labels are hard to be obtained. With a small subset of labeled data, we demonstrate a great accuracy improvement of wafer pattern recognition under a semi-supervised learning configuration.

The dissertation is concluded in Chapter 8.

Chapter 2

Hybridizing Formal Methods and Machine Learning for AMS Verification

This chapter presents a new perspective in AMS verification by proposing a hybrid formal/machine-learning verification (HFMV) framework that simultaneously exploits formal and machine learning techniques as shown in Fig. 2.1. In its most abstract form, HFMV comprises two key elements: a *probabilistic machine learning model* and *formal verification* that acts on top of the machine learning model. It is the interactions between the two elements that form the promise of HFMV. The probabilistic model is trained from limited simulation/measurement data and comes with a measure of uncertainty for each prediction of the circuit performance under verification. Given a bounded verification space of design or uncertainty parameters, e.g. process variations or operating conditions,

©2018 IEEE. Reprinted, with permission, from Hanbin Hu, Qingran Zheng, Ya Wang, and Peng Li, "HFMV: Hybridizing Formal Methods and Machine Learning for Verification of Analog and Mixed-Signal Circuits", ACM/ESDA/IEEE Design Automation Conference (DAC), 2018.

formal verification is applied with respect to a symbolic formula derived from the posterior prediction of the probabilistic learning model to check if the targeted specification is met across the *entire* verification space with a sufficiently high confidence.

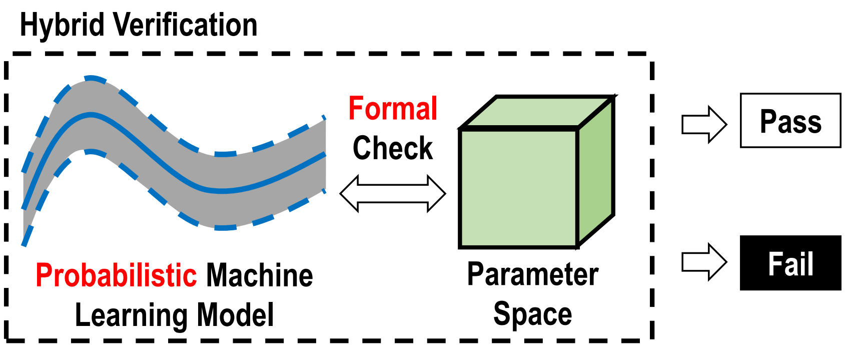


Figure 2.1: Hybrid verification framework HFMV.

HFMV has the best of the two worlds: it adds a degree of formalism on top of learning-based models by utilizing satisfiability modulo theories (SMT) [31, 32] formal techniques; and it is much more scalable than pure formal techniques at the same time. Furthermore, to circumvent the inherent uncertainty of machine learning, the proposed framework "formally" bounds learning model uncertainty and practically verifies design properties over a high-dimensional space of design uncertainty. HFMV presents several key contributions to AMS verification:

- Bridges the gap between design complexity and scalability of verification by integrating formal and machine-learning techniques into a general hybrid verification framework;
- Builds a degree of formalism into machine-learning based verification to safeguard detection of extremely rare failure under a limited data budget;
- Explores novel formally-guided active learning to iteratively reduce learning model uncertainty towards rare failure detection;

- Significantly accelerates formal solutions by efficient one-time preprocessing of SMT formulas to be checked.

Experimental studies have demonstrated that HFMV can reliably verify AMS design specifications and identify extremely rare failures under complex high-dimensional parametric uncertainties for which state-of-the-art smart statistical sampling techniques fail.

2.1 Probabilistic Model-Based Failure Prediction

We propose a notion of probabilistic model-based failure prediction before presenting HFMV. Given a bounded D-dimensional parameter space $\Omega \subseteq \mathbb{R}^D$, the true performance $y(\mathbf{x})$ at a particular point $\mathbf{x} \in \Omega$ of the design under verification (DUV) can be determined either by simulation or measurement. Without loss of generality, a point $\mathbf{x} \in \Omega$ is considered as a (true) failure if $y(\mathbf{x}) \geq T$, where T is the targeted specification (assuming greater the value, worse the performance). Verifying a highly robust design for which failures are extremely rare, finding a failure point in a high-dimensional space can be extremely challenging and expensive in terms of numbers of measurements and simulation samples needed. We propose to leverage an efficient probabilistic machine learning model to replace direct measurements and simulations.

2.1.1 Probabilistic Machine Learning Model

HFMV exploits a large body of popular probabilistic machine learning models where each inference is probabilistic such as Bayesian additive regression trees [33], kriging [34], relevance vector machine (RVM) [35], and sparse relevance kernel machine (SRKM) [36, 37]. The last three fall under the broad family of Gaussian processes.

Generally, each prediction from a probabilistic model with model parameters θ is based on a posterior predictive distribution, whose cumulative distribution function (CDF) $F_Y(y;\theta)$ specifies the probability for true performance $y(\mathbf{x})$ to fall in the range of $[a,b]$:

$$Prob\{a \le y(\mathbf{x}) \le b\} = F_Y(b;\theta) - F_Y(a;\theta). \tag{2.1}$$

We define the **P-Prediction** $\hat{y}(\mathbf{x},P;\theta)$ associated with a probability value P for a certain point $\mathbf{x}$ as:

Definition 1 $\hat{y}(\mathbf{x},P;\theta) = F_Y^{-1}(1-P;\theta)$.

According to the definition above, it is straightforward to show that the probability for true performance $y(\mathbf{x})$ to be no less than $\hat{y}(\mathbf{x},P;\theta)$ is P:

$$Prob\{y(\mathbf{x}) \ge \hat{y}(\mathbf{x},P;\theta)\} = P. \tag{2.2}$$

Consider SRKM as an example, which is an extension to the relevance vector machine (RVM) [35] and offers improved accuracy and the appealing probabilistic feature weighting capability [36, 37]. A trained SRKM model has a posterior Gaussian predictive prediction with mean $\hat{y}_{est}(\mathbf{x})$ and variance $\hat{\sigma}_{est}(\mathbf{x})$ at a point $\mathbf{x} \in \Omega$ as:

$$y \sim \mathcal{N}\left(\hat{y}_{est}(\mathbf{x}), \hat{\sigma}_{est}^2(\mathbf{x})\right) \tag{2.3}$$

$$\hat{y}_{est}(\mathbf{x}) = \bar{\mathbf{v}}^T\mathbf{K}(\mathbf{x}) \tag{2.4}$$

$$\hat{\sigma}_{est}(\mathbf{x}) = \sqrt{\sigma^2 + \mathbf{K}(\mathbf{x})^T\mathbf{\Sigma}_{\mathbf{v}}\mathbf{K}(\mathbf{x})}, \tag{2.5}$$

where σ^2 is the estimated intrinsic noise, $\bar{\mathbf{v}}$ and $\mathbf{\Sigma}_{\mathbf{v}}$ are the posterior $D \times 1$ expectation and $D \times D$ covariance matrix of the feature weights, respectively, and $\mathbf{K}(\mathbf{x})$ is the $D \times 1$ design vector based on a chosen kernel function, which will be further discussed in Section

2.4. The detailed expressions for the above prediction can be found from [36, 37]. The P-Prediction for SRKM is given by:

$$\hat{y}_{SRKM}\left(\mathbf{x}, P; \theta\right) = \hat{y}_{est}\left(\mathbf{x}\right) - \Phi^{-1}\left(P\right) \cdot \hat{\sigma}_{est}\left(\mathbf{x}\right). \tag{2.6}$$

where $\Phi^{-1}\left(\cdot\right)$ is the inverse function of CDF of a standard normal distribution.

2.1.2 Probabilistic Failure Detection

We leverage a probabilistic model for failure detection. However, instead of using the optimal posterior performance estimator, which is the mean $\hat{y}_{est}\left(\mathbf{x}\right)$ of the posterior predictive distribution, we make use of the P-Prediction to cope with uncertainty of machine learning and check if $\mathbf{x}$ is a failure at a given confidence level by:

$$\hat{y}\left(\mathbf{x}, P; \theta\right) \geq T. \tag{2.7}$$

If (2.7) holds true, it is easy to see the following based on the monotonicity of the CDF:

$$Prob\left\{y\left(\mathbf{x}\right) \geq T\right\} \geq Prob\left\{y\left(\mathbf{x}\right) \geq \hat{y}\left(\mathbf{x}, P; \theta\right)\right\} = P. \tag{2.8}$$

Therefore, if a point $\mathbf{x}$ satisfies $\hat{y}\left(\mathbf{x}, P; \theta\right) \geq T$, $\mathbf{x}$ is a (true) failure with a probability at least P. If P is large enough, $\mathbf{x}$ can be identified as a true failure (red cross) with a high confidence as shown in Fig. 2.2. Conversely, if the P-Prediction $\hat{y}\left(\mathbf{x}, P; \theta\right) < T$, then

$$Prob\left\{y\left(\mathbf{x}\right) < T\right\} \geq 1 - Prob\left\{y\left(\mathbf{x}\right) \geq \hat{y}\left(\mathbf{x}, P; \theta\right)\right\} = 1 - P. \tag{2.9}$$

Therefore, $\mathbf{x}$ is a good design point with a probability at least $1-P$. When P is small enough, $\mathbf{x}$ may be identified as a good point (blue dot) with high confidence as shown in

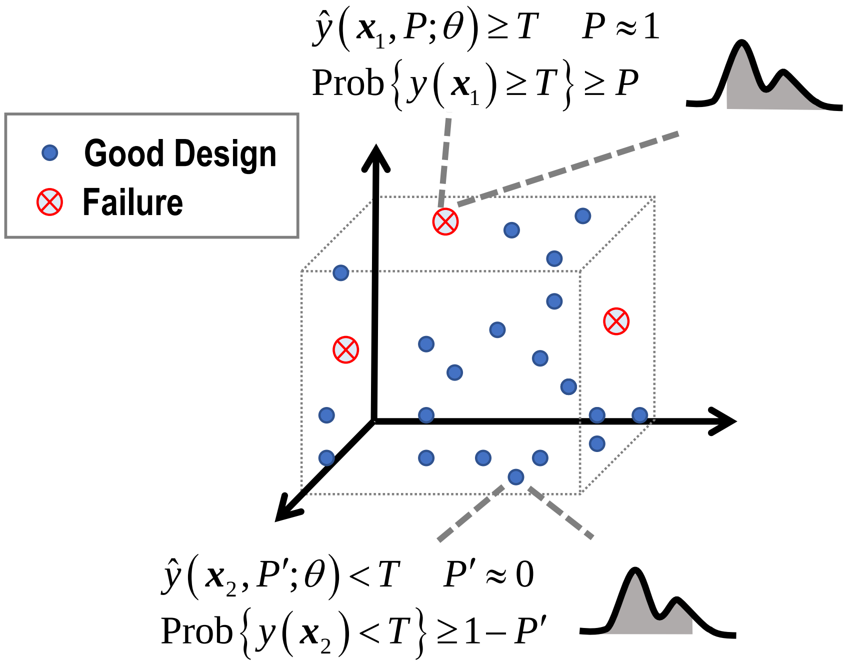

$\hat{y}(\boldsymbol{x}_1, P; \theta) \geq T \quad P \approx 1$
$\text{Prob}\{y(\boldsymbol{x}_1) \geq T\} \geq P$
Good Design
Failure
$\hat{y}(\boldsymbol{x}_2, P'; \theta) < T \quad P' \approx 0$
$\text{Prob}\{y(\boldsymbol{x}_2) < T\} \geq 1 - P'$

Figure 2.2: Probability to be a failure/good point by a probabilistic machine learning model.

Fig. 2.2. Based on the satisfiability of (2.7), $\mathbf{x}$ can be classified as a failure/good point with certain model belief determined by P as summarized in Table 2.1.

Table 2.1: Model beliefs based on statisfiability of (2.7).

	$P \approx 1.0$	$P \approx 0.0$
SAT	Failure (strong belief)	Failure (weak belief)
Non-SAT	Good (weak belief)	Good (strong belief)

2.2 Formal Problem Formulation

We apply formal verification on top of a trained probabilistic machine learning model to provide a degree of **coverage** for failure detection, which is accomplished by exhaustively proving or disproving a given specification T at an adaptively chosen confidence level P in the entirety of the parameter space Ω. To do so, we make use of the recent advances in satisfiability modulo theories (SMT) solvers.

SMT solvers are extensions to Boolean satisfiability (SAT) counterparts which check the satisfiability of formulas defined on Boolean variables and operations. SMT solvers come with added expressiveness of uninterpreted function symbols, equality, quantifiers, and various operations such as arithmetic, datatype and array operations [38]. While originally developed in 1970s, SMT technology has undergone significant improvements lately. A number of efficient SMT solvers have emerged, for example Z3[32] and iSAT3 [31].

2.2.1 Two Hybrid Verification Problems

In HFMV, *two hybrid verification problems are defined.* To exhaustively check the existence of any failure point according to the machine learning model belief in the entire parameter space, we define an SMT-based problem called **Failure Detection Problem** using (2.7):

$$\exists \mathbf{x} \in \Omega \text{ s.t. } \hat{y}\left(\mathbf{x}, P; \theta\right) \geq T,\ P \approx 1.0. \tag{2.10}$$

A SAT solution with P close to one returned by the SMT solver is very likely to be a true failure. If this is verified to be a true failure by a single simulation/measurement, a "Fail" conclusion is immediately drawn for the verification task, and additional failures may be obtained by finding more SAT solutions if desired.

We define the **Design Certification Problem** which is checked when one attempts to verify that the targeted specification is met across the entire parameter space:

$$\exists \mathbf{x} \in \Omega \text{ s.t. } \hat{y}\left(\mathbf{x}, P; \theta\right) \geq T,\ P \approx 0. \tag{2.11}$$

A Non-SAT solution from the SMT solver indicates that all points in Ω are believed to be good by the model at a high confidence level. In practice, we draw a "Pass" conclusion of verification only when both P and the model uncertainty (measured by the prediction variance) are sufficiently low, the latter of which is achieved during the active-learning guided iterative model re-training process described in Section 2.3. By tuning the confidence level P and monitoring the model uncertainty while operating on the two problems, we direct the SMT solver towards solving either the *Failure Detection Problem* or the *Design Certification Problem*.

Note again the HFMV framework can be built upon any probabilistic machine learning model as long as a posterior predictive distribution $F_Y(y;\theta)$ is provided. Using SRKM as the underlying machine learning model as an example, assume Ω is a D-dimensional bounding box with $\underline{x(i)} \le x(i) \le \overline{x(i)}$ along each parameter dimension i, the SMT form of the *Failure Detection Problem* or *Design Certification Problem* at a properly chosen P is:

$$\begin{aligned} &\exists \mathbf{x} \in \mathbb{R}^D \\ \text{s.t.} \;& \left\{\hat{y}_{est}(\mathbf{x}) - \Phi^{-1}(P) \cdot \hat{\sigma}_{est}(\mathbf{x}) \ge T\right\} \\ &\wedge \left\{\underline{x(i)} \le x(i) \le \overline{x(i)}\right\}, i = [1, D]. \end{aligned} \tag{2.12}$$

2.3 Proposed Active Learning

Based on the fact that failures are extremely rare and hard to detect, a small initial training dataset may not contain any failure, which results in an initial "lousy" probabilistic model as illustrated in Fig. 2.3. To address this challenge, we propose to iteratively improve the model accuracy through active learning with the goal of approaching rare failure regions under a limited data budget. Active learning selects optimal sampling locations on-the-fly and directs re-training of the machine learning model across multiple iterations. We explore two active learning approaches: **1)** max variance learning to reduce model uncertainty based on max variance values of model prediction, and **2)** a novel formally-guided approach aiming at discovery of rare failure regions.

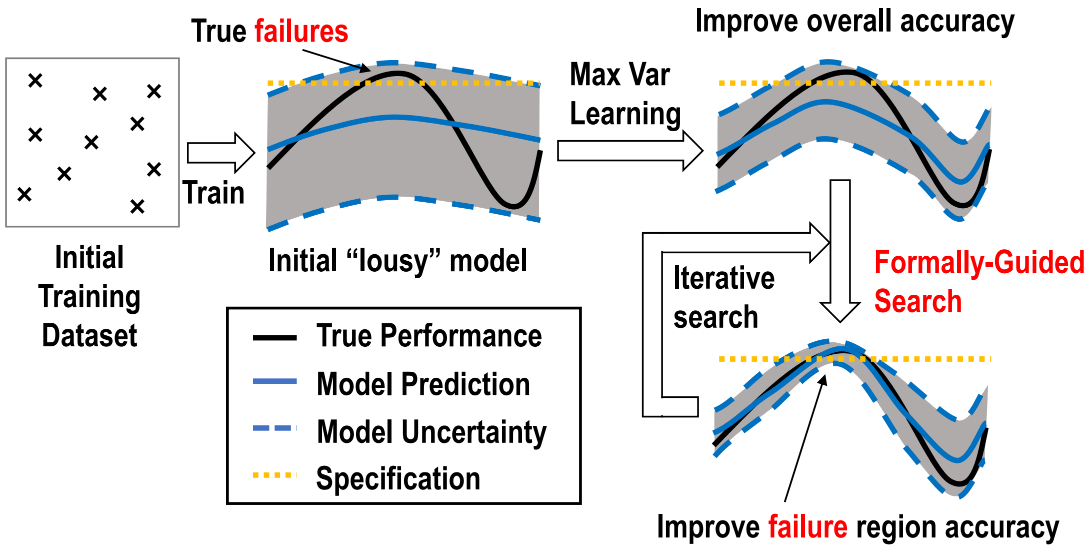


Figure 2.3: Proposed active learning.

2.3.1 Max Variance Learning

The posterior predictive distribution $F_Y(y;\theta)$ reveals the essential information of model uncertainty. In particular, regions with large prediction variance $Var(y;\theta)$ correspond to locations where model uncertainty is high. Additional sampling can be performed at points with the largest variance to improve the overall model accuracy:

$$\underset{\mathbf{x}}{\operatorname{argmax}}\, Var(y;\theta), \text{ s.t. } \mathbf{x} \in \Omega. \tag{2.13}$$

Since the above *Max Variance Learning* phase takes place early on in the active learning process as shown in Fig. 2.3, the optimization needs not to be done exactly. Instead, we efficiently evaluate the model variance at a large number of randomly chosen points in Ω, and pick the top N_{var} locations for additional simulation or measurement. Then, the model is retained using the larger training dataset. Experimentally, performing one or two such iterations is sufficient.

2.3.2 Formally-Guided Active Learning

Finding extremely-rare failures can be very challenging for designs with stringent failure-rate requirements. Improving just the overall machine learning model accuracy as typically done in a standard active learning strategy is far from addressing the rare-failure detection challenge. Our key idea is to propose a novel formally-guided active learning approach, where the main objective is to search for the most-probable failure locations in the entire high-dimensional parameter space as shown in Fig. 2.4.

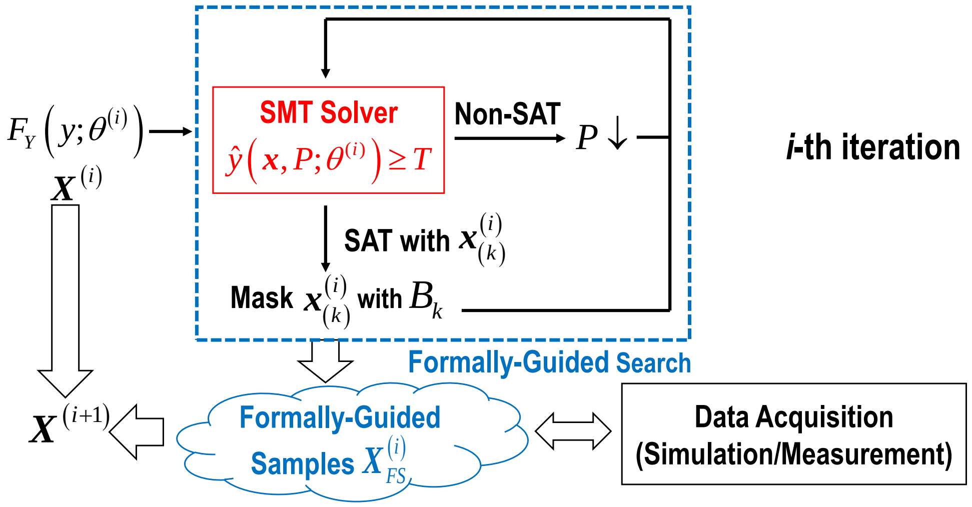

$F_Y\left(y;\theta^{(i)}\right)$
$\boldsymbol{X}^{(i)}$
SMT Solver
$\hat{y}\left(\boldsymbol{x},P;\theta^{(i)}\right)\geq T$
Non-SAT
$P\downarrow$
i-th iteration
SAT with $\boldsymbol{x}_{(k)}^{(i)}$
Mask $\boldsymbol{x}_{(k)}^{(i)}$ with B_k
Formally-Guided Search
$\boldsymbol{X}^{(i+1)}$
Formally-Guided Samples $X_{FS}^{(i)}$
Data Acquisition (Simulation/Measurement)

Figure 2.4: Formally-guided active learning.

For the i-th iteration of the proposed formally-guided approach, denote the parameters and the posterior predictive distribution of the present model by $\theta^{(i)}$ and $F_Y\left(y;\theta^{(i)}\right)$, respectively, which are trained on the current dataset $\mathbf{X}^{(i)}$. The corresponding P-Prediction is denoted by $\hat{y}\left(\mathbf{x},P;\theta^{(i)}\right)$. The *Failure Detection Problem* of (2.10) is solved to find the most-probable failure locations with $P\approx 1$. It is entirely possible that a Non-SAT solution is returned, indicating no points can be identified as a failure with a strong model belief. In this case, P is reduced with a small step gradually to allow for finding candidate failure points with reduced model confidence. Otherwise, a returned

SAT solution $x_{(k)}^{(i)}$ satisfying (2.10) will be included as the kth sampling point in the i-th iteration of active learning. We repeatedly solve the following SMT instance to get the $(k+1)$-th point while avoiding getting the same solutions returned before:

$$\begin{aligned} &\exists \mathbf{x}_{(k+1)}^{(i)} \in \Omega \backslash \cup \{B_1, B_2, \ldots, B_k\} \\ &\text{s.t. } \hat{y}\left(\mathbf{x}_{(k+1)}^{(i)}, P; \theta^{(i)}\right) \geq T, \end{aligned} \tag{2.14}$$

where each B_j $(j = [1,k])$ is a D-dimensional bounding box, i.e. we can define a hypercube as $B_j = \left\{\mathbf{x} \in \Omega \,\middle|\, \left\|x(p) - x_{(j)}^{(i)}(p)\right\| \leq d_0, p = [1, D]\right\}$ which encloses $x_{(j)}^{(i)}$ at its center with a length of $2d_0$ along each dimension. Assume at each i-th active learning iteration, a user-defined N_i number of formally guided samples are selected: $\mathbf{X}_{FS}^{(i)} = \left\{x_{(1)}^{(i)}, x_{(2)}^{(i)}, \ldots, x_{(N_i)}^{(i)}\right\}$. All points in $\mathbf{X}_{FS}^{(i)}$ are queried using either circuit simulation or measurement to obtain the corresponding true performance values. Adding these training samples to the dataset used in the i-th iteration gives a larger dataset: $\mathbf{X}^{(i+1)} = \mathbf{X}^{(i)} \cup \mathbf{X}_{FS}^{(i)}$, which is used to re-train the model and update the predictive distribution $F_Y\left(y; \theta^{(i+1)}\right)$.

"Actively" finding out the most-probable failure locations in the high-dimensional parameter space is instrumental for extremely-rare failure detection under limited data budgets. The proposed active learning process terminates when reaching the set data limit, when a large percentage of formally determined points are verified to be true failures, or when a targeted number of true failures have been found. In the event of no true failure detection during the active learning process, we then attempt to solve the *Design Certification Problem* of (2.11) by which we certify the circuit to be good if both P and model uncertainty are sufficiently low.

2.4 Acceleration of SMT Solutions

A key computational component of the proposed HFMV framework to solve variants of (2.10). To significantly boost runtime efficiency, a promising solution is to simplify the nonlinear SMT formula through novel equivalent transformations or approximations that can be much more efficiently solved. We propose two numerical preprocessing schemes: *input space re-mapping* and *linear approximation* under the context of SRKM based probabilistic model. These two techniques only present negligible one-time pre-processing overhead for each SMT instance but have been shown to speed up SMT solving by a few orders of magnitude.

2.4.1 Input Space Re-Mapping

SRKM employs a vector kernel function $\mathbf{K}\left(\mathbf{x}_*\right) \in \mathbb{R}^{D\times 1}$ of the following form to compute the similarity between M training samples and the input vector $\mathbf{x}_*$ at which a prediction shall be made over D parameter dimensions:

$$\mathbf{K}\left(\mathbf{x}_*\right)(k) = \sum_{j=1}^{M} \omega\left(j\right) \cdot K_k\left(\mathbf{x}_*\left(k\right), \mathbf{X}\left(j\right)\left(k\right)\right), k = [1, D], \tag{2.15}$$

where $\omega\left(j\right)$ is the posterior mean estimation for the j-th sample weight, $\mathbf{X}\left(j\right)\left(k\right)$ is the k-th feature of the j-th training sample from the training dataset $\mathbf{X}$, and $K_k\left(\cdot,\cdot\right)$ is a scalar kernel function measuring the similarity of two input vectors over the k-th dimension, which can be chosen arbitrarily by the user to be, for example, a radial basis function (RBF) kernel or polynomial kernel.

Important to note that in a trained model, $\mathbf{x}_*$ is the only symbolic vector variable in (2.15) and other terms are known constants. Furthermore, $\mathbf{K}\left(\mathbf{x}_*\right)(k)$ is only symbolically dependent on the k-th dimension (parameter) $\mathbf{x}_*\left(k\right)$ of the input vector $\mathbf{x}_*$, allowing defining new symbolic variables $\mathbf{a} = [a(1), a(2), \cdots, a(D)]^T$:

$$a(k) = g(\mathbf{x}_*(k)) = K(\mathbf{x}_*)(k), \tag{2.16}$$

where function $g(\cdot)$ is introduced to signify the fact that $a(k)$ only depends on $\mathbf{x}_*(k)$. This allows to re-map the input vector from the original X-space to the new A-space. The minimum $\underline{a(k)}$ and maximum $\overline{a(k)}$ of $a(k)$ for each dimension can be obtained through a trivial one-dimensional optimization:

$$\underline{a(k)} = \min_{\underline{x_*(k)} \le x_*(k) \le \overline{x_*(k)}} K(\mathbf{x}_*)(k), \tag{2.17}$$

$$\overline{a(k)} = \max_{\underline{x_*(k)} \le x_*(k) \le \overline{x_*(k)}} K(\mathbf{x}_*)(k), \tag{2.18}$$

where $\underline{x_*(k)}$ and $\overline{x_*(k)}$ specify the bounds of the k-th component of the input vector in the original parameter space Ω.

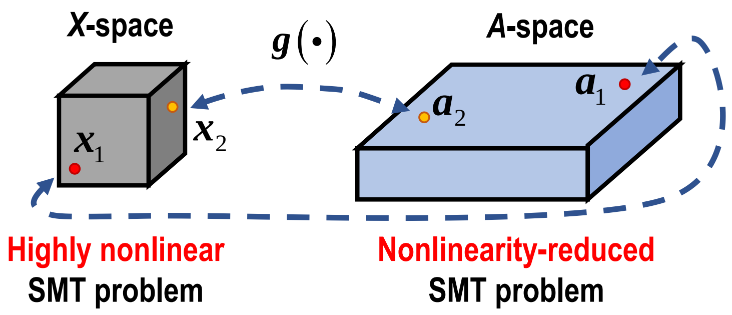


Figure 2.5: Input space re-mapping.

Instead of operating in the the original X-space, the SMT problem can be reformulated in the A-space as illustrated in Fig. 2.5:

$$\begin{aligned}
&\exists \mathbf{a} \in \mathbb{R}^D \\
\text{s.t. } &\left\{ \bar{\mathbf{v}}^T \mathbf{a} - \Phi^{-1}(P) \cdot \sqrt{\sigma^2 + \mathbf{a}^T \boldsymbol{\Sigma}_{\mathbf{v}} \mathbf{a}} \geq T \right\} \\
&\wedge \left\{ \underline{a(k)} \leq a(k) \leq \overline{a(k)} \right\}, k = [1, D].
\end{aligned} \tag{2.19}$$

The new SMT instance of (2.19) is equivalent to the original problem while having no strong nonlinearity introduced by the nonlinear kernel function, and hence can be more efficiently solved. A solution obtained in the A-space is easily mapped back to the X-space numerically.

2.4.2 Linear Approximation

Note that the formula of (2.19) is nonlinear due to the square root computation for the model variance and the quadratic term $\mathbf{a}^T \boldsymbol{\Sigma}_{\mathbf{v}} \mathbf{a}$. We propose to find a close linear approximation of (2.19) such that a state-of-the-art fast linear SMT solver such as Z3 [32] can be applied.

Since Σ_v is a positive semidefinite matrix, the lower bound of $\mathbf{a}^T \boldsymbol{\Sigma}_{\mathbf{v}} \mathbf{a}$ term can be efficiently found by a one-time convex quadratic minimization within the bounded hypercube:

$$l_a = \min_{\mathbf{a}} \mathbf{a}^T \Sigma_v \mathbf{a}, \text{ with } \underline{a(k)} \leq a(k) \leq \overline{a(k)}, k = [1, D]. \tag{2.20}$$

Inserting the this lower bound into (2.19) leads a safe linear approximation to (2.21):

$$\begin{aligned}
&\exists \mathbf{a} \in \mathbb{R}^D \\
\text{s.t. } &\left\{ \bar{\mathbf{v}}^T \mathbf{a} - \Phi^{-1}(P) \cdot \sqrt{\sigma^2 + l_a} \geq T \right\} \\
&\wedge \left\{ \underline{a(k)} \leq a(k) \leq \overline{a(k)} \right\}, k = [1, D].
\end{aligned} \tag{2.21}$$

Our results show that this linear formula is reasonably accurate and can be very efficiently solved. The identified candidate failure points are further checked by the exact formula to filter out false solutions. Z3 can solve 1,000 SMT instances of (2.21) in 2 CPU minutes while solving one instance of (2.19) may take around 9 minutes.

2.5 Experimental Results

We test the proposed HFMV on three analog circuits and compare its performance with the Monte Carlo (MC) method and Scaled-Sigma Sampling algorithm (SSS) [13, 12] , a state-of-the-art smart statistical sampling technique that has demonstrated excellent performance for analog yield estimation. To maximize the possibility of hitting rare failures based on MC, uniform sampling is adopted in each bounded parameter space. The prototyped HFMV tool was developed using SRKM as the underlying probabilistic machine learning model in C++, and compiled by g++ 4.8.5, and runs on a server with 2.8GHz Intel(R) Xeon(R) CPU E5-2680 v2 Processors.

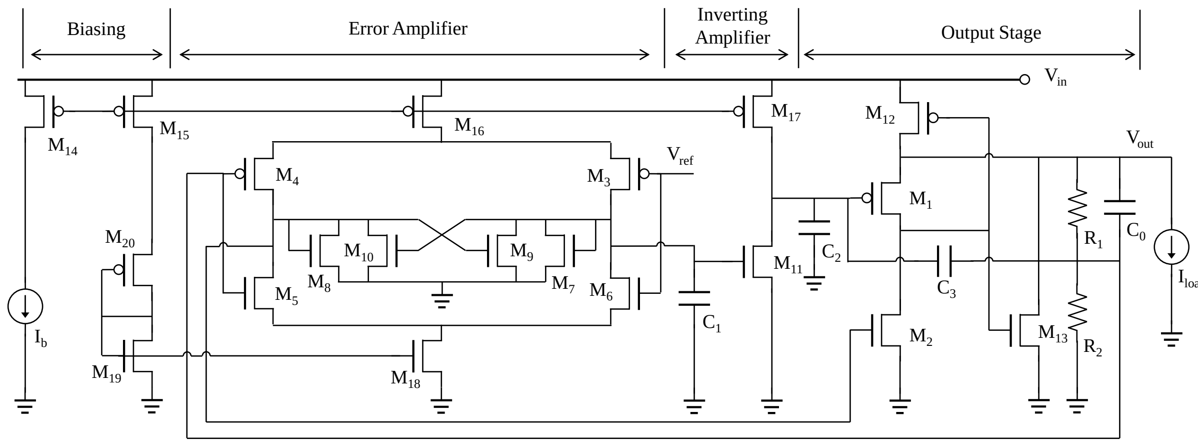


Figure 2.6: A LDO with 60 transistor-level variations.

The three test circuits are: a differential amplifier (Amp), a low-dropout voltage regulator (LDO) (Fig. 2.6) [39] , and a DC-DC converter (DCDC) (Fig. 2.7) [40] , all designed

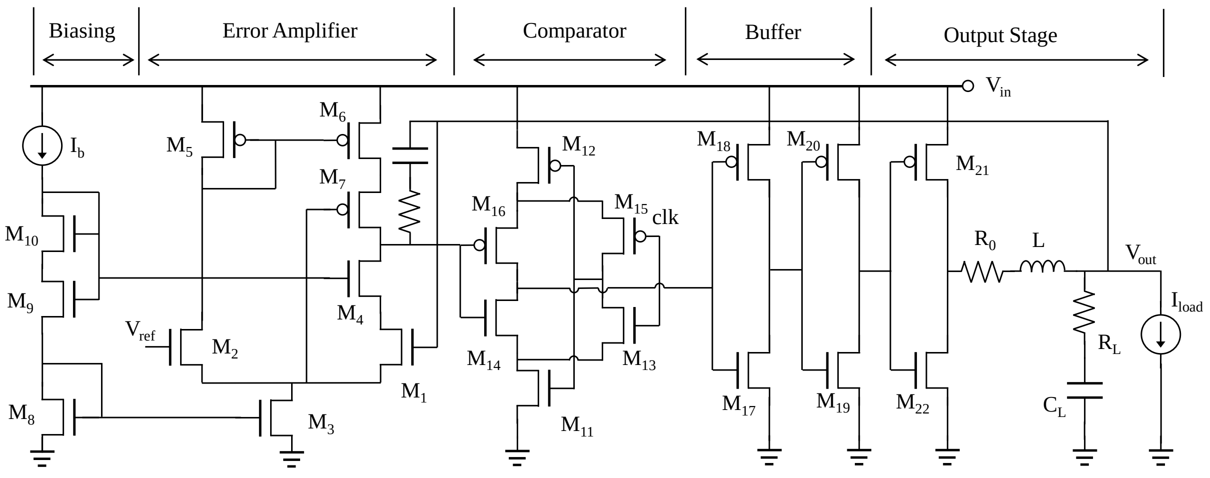


Figure 2.7: A DC-DC converter with 44 transistor variations.

using a commercial 90nm CMOS technology design kit. Simulation data is collected using Synopsys HSPICE (for Amp and LDO) and Cadence Spectre (for DCDC). A few specifications for each of the following 10 performances are chosen as verification targets: GBW, gain and CMRR for the amplifier, OA (output accuracy), OS (overshoot), RS (ripple size) and PE (power efficiency) for the DC-DC converter, and QC (quiescent current), US (undershoot) and LR (load regulation) for the LDO. Three types of transistor-level variations are considered for each transistor in the amplifier and LDO: channel length, threshold voltage, and gate oxide thickness, resulting in a 15-dimensional and 60-dimensional verification problem, respectively. Channel length and width variations are considered for each transistor in the DC-DC converter, resulting in a 44-dimensional verification problem.

Two 15-D hyper-cubes covering $\pm 4\sigma$ and $\pm 8\sigma$ variation of each device parameter around the mean, respectively, are set up as the bounded parameter space for verification of the amplifier. Similarly, 44-D and 60-D hyper-cubes are set up for $\pm 4\sigma$ and $\pm 8\sigma$ verification of the DC-DC converter and LDO, respectively. Based upon the Gaussian distribution used to model all process variations (not required by HFMV though), the probability masses outside each hyper-cube are 0.1%, 0.3% and 0.4% respectively for the amplifier, DC-DC converter and LDO in the $\pm 4\sigma$ case, and $2 \times 10^{-12}\%$, $6 \times 10^{-12}\%$, and $8 \times 10^{-12}\%$ for the three designs in the $\pm 8\sigma$ case.

2.5.1 Active-Learning Guided Failure Discovery

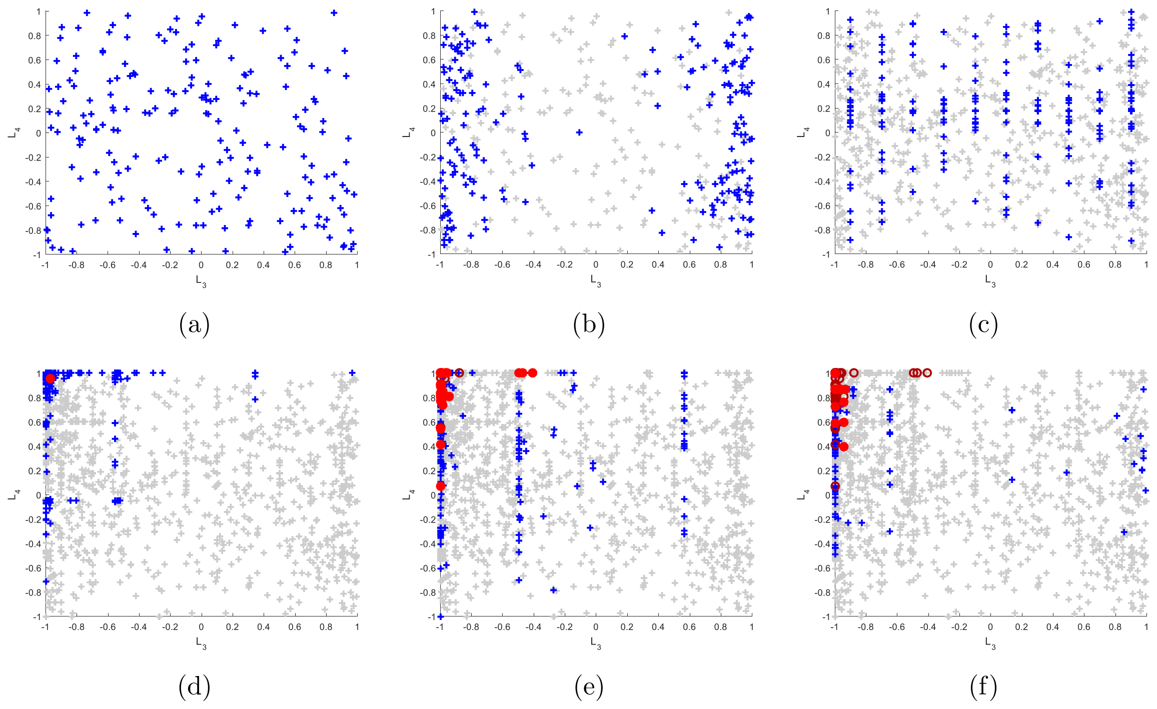


Figure 2.8: Selected 6 iterations of proposed active learning projected onto a 2D space spanned by two device variables for $\pm 8\sigma$ failure detection of amplifier CMRR. Gray crosses: samples from the previous iteration; blue crosses: samples selected in current iteration; red circle: sampled true failures. (a) Initial dataset; (b) max variance learning; (c) first iteration of formally-guided active learning; (d) first true failure found; (e)(f) large numbers of failures found.

By interfacing with HSPICE or Spectre, HFMV starts with 200 initial simulation samples uniformly sampled in the bounded parameter space. After that, the proposed active learning strategy performs two rounds of sampling to collect 400 data points with max variance of model prediction, and then directs failure discovery by collecting around $N_i \approx 350$ samples for each formally-guided learning iteration. During this process, we record the number of samples taken for finding the first true design failure. The process moves on to find additional failures until reaching a user-defined target or sample size limit. The verification process ended when the 75% of samples in one round of X_{FS} are true failure points or the training sample size exceeds 4000. Much larger numbers of random simulation samples are applied to MC and SSS. In case of finding no failure, a specification is considered satisfied across the entire bounded parameter space if a non-SAT solution is returned for the SMT formula that checks the nonexistence of any point at which the performance meets the specification with a probability lower than some user-specified probability approximating to 100%.

We illustrate the active-learning guided failure discovery process for the challenging task of $\pm 8\sigma$ CMRR verification of the amplifier in Fig. 2.8. The process starts off with samples having max SRKM model variance to improve the overall model accuracy. The proposed active learning then directs the sampling process towards rare failure points in the 15-D bounded parameter space. The effectiveness of the active learning can be observed by the discovery of a true failure point early on in the process and then many other failure points later on, which are very rare.

Table 2.2: Comparison on $\pm 4\sigma$ failure detection. # Samp: # of training (simulation) samples used by each method; # 1st Fail: # of samples used for finding the first true failure by HFMV; # Fail: # of failures found in the bounded parameter space.

Spec.		Target	HFMV			SSS	
			# Samp	# 1st Fail	# Fail	# Samp	# Fail
Amp	GBW	22MHz	1,307	600	227	6,000	0
	Gain	2.5dB	2,307	1,507	155	6,000	0
	CMRR	10dB	1,400	1,000	437	6,000	0
DCDC	OA	5.50%	1,200	600	334	4,000	0
	OS	0.94%	1,000	600	319	4,000	0
	RS	0.598mV	1,000	600	162	4,000	0
	PE	83.20%	900	600	198	4,000	0
LDO	QC	16mA	2,486	600	287	6,000	0
	US	60%	1,800	1,000	319	6,000	0
	LR	55%	1,897	898	435	6,000	0

Table 2.3: MC results on $\pm 4\sigma$ failure detection. Variables defined as in Table 2.2.

Spec.		Target	MC		
			# Samp	# Fail	Time
Amp	GBW	22MHz	600,000	0	228:10:48
	Gain	2.5dB	600,000	0	
	CMRR	10dB	600,000	0	
DCDC	OA	5.50%	45,000	0	699:16:48
	OS	0.94%	45,000	0	
	RS	0.598mV	45,000	0	
	PE	83.20%	45,000	0	
LDO	QC	16mA	649,000	0	160:25:12
	US	60%	649,000	0	
	LR	55%	649,000	0	

Table 2.4: Comparison on $\pm 8\sigma$ failure detection. Variables defined as in Table 2.2.

Spec.		Target	HFMV			SSS	
			# Samp	# 1st Fail	# Fail	# Samp	# Fail
Amp	GBW	5MHz	1,000	600	396	9,000	0
DCDC	OA	10.0%	1,300	600	312	9,000	0
	OS	1.00%	600	600	85	9,000	0
	RS	0.6mV	600	600	87	9,000	0
	PE	80.00%	1,000	600	275	9,000	0
LDO	QC	20mA	896	600	140	9,000	0
	US	100%	1,897	897	381	9,000	0
	LR	80%	1,618	599	382	9,000	0

2.5.2 Rare Failure Detection

All three methods are applied to $\pm 4\sigma$ verification of 10 specifications of the three designs as in Table 2.2 and Table 2.3. Only HFMV and SSS are applied to $\pm 8\sigma$ verification as shown in Table 2.4 as it is almost completely meaningless to even try MC within such wide-ranges of parameter variations for finding any extremely rare failure. HFMV mainly targets at extremely rare failure detection, where the samples are very expensive to collect. The listed runtimes for MC in Table 2.3 attempt to demonstrate that even for relatively small circuits, the simulation cost is huge when requiring a large number of samples. When facing a fairly large circuit, the simulation time can easily dominate the overall HFMV runtime. For example, the overall HFMV runtime for output accuracy of the DCDC converter cost around 14 hours, and around 10 hours were consumed by simulation for the pre-layout schematic. Hence, the number of simulation runs shall be minimized as much as possible. As seen from Table 2.2 and Table 2.4, the numbers of simulation samples used by HFMV are significantly lower than SSS and MC. HFMV can hit the first true failure point using 600 to about 1,500 samples, which are about **10x and up to 1,000x lower than used by SSS and MC**, respectively. Yet, both MC and SSS cannot find any true failure in the bounded parameter space. While SSS is one of the state-of-the-art statistical sampling technique and has been shown to produce excellent results for yield estimation of analog circuits [13, 12] , it lacks mechanisms specifically targeting for finding extremely rare failure locations in high-dimensional parameter spaces.

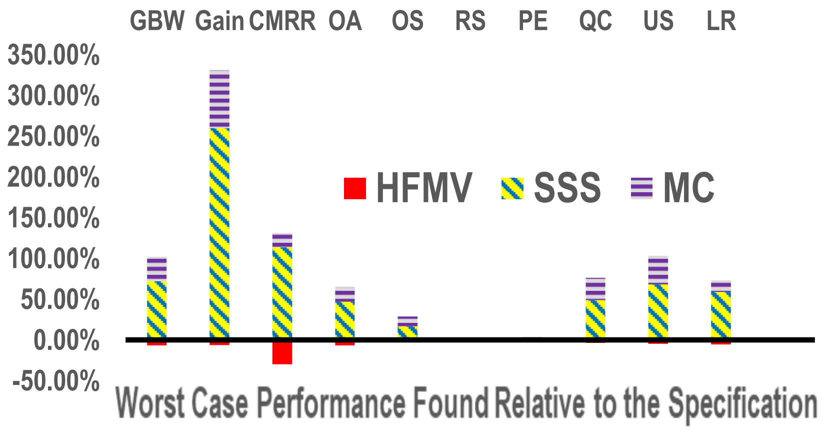


Figure 2.9: Worst-case performances relative to the specifications found by each method in the $\pm 4\sigma$ region.

Fig. 2.9 and Fig. 2.10 report the worst-case performances normalized with respect to the corresponding specifications found by each method. It can be observed that

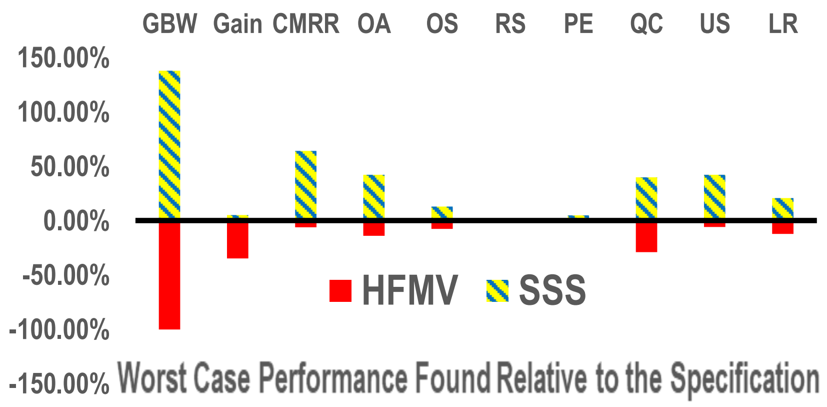


Figure 2.10: Worst-case performances relative to the specifications found by HFMV and SSS in the $\pm 8\sigma$ region.

since both MC and SSS fail to find any true failure for all targeted performances, they produce misleading outcomes for verification. In contrast, HFMV is able to find many specification violations (true failures). The identified worst-case performance values can be significantly *worse* than the corresponding specifications.

2.6 Summary

A novel hybrid approach, namely HFMV, has been presented to address rare failure detection challenges associated with AMS verification. HFMV combines the key benefits of formal verification and machine-learning based approaches while circumventing their key limitations in terms of scalability and model uncertainty. It has been demonstrated that HFMV can provide reliable verification of AMS performance specifications in high-dimensional parameter spaces for which both Monte Carlo and a state-of-the-art sampling technique lead to misleading results.

Chapter 3

Parallelizable Bayesian Optimization for AMS Verification with High Coverage

This chapter intends to address the rare failure detection problem by introducing a sequential experiment design technique called Bayesian Optimization (BO) into AMS verification field. Popularized by its applications in automatic hyper-parameter tuning for machine learning [41, 42] and reinforcement learning [43] in recent years, BO is a powerful tool to find the optimum value for a black-box function [44]. In integrated circuit design area, BO also gains interest recently in DNN hardware design [45] and analog circuit optimization [46]. Instead of dealing with the original hard-to-optimize function, BO optimizes over an easy-to-optimize statistical model learned by a small number of function samples (i.e., function evaluations) sequentially collected based on

©2018 IEEE. Reprinted, with permission, from Hanbin Hu, Peng Li, and Jianhua Z. Huang, "Parallelizable Bayesian Optimization for Analog and Mixed-Signal Rare Failure Detection with High Coverage", IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2018.

the iteratively updated learning model.

By balancing between detecting potential failures of the current model (exploitation) and exploring the highly-uncertain space (exploration), our proposed BO method attempts to optimize the worst performance value using as small number of of samples as possible. This economical use of samples is attractive for AMS design verification since data collection is usually very costly. Moreover, unlike standard uses of BO, we go beyond a pure optimization task (finding the worst failures) by taking into account the coverage issue (finding multiple failure mechanisms). Since good coverage helps identifying different failure regions (or failure mechanisms) and is beneficial for circuit designers to refine and improve the circuits, the coverage issue is of particular interest to AMS design verification, though not a concern in typical BO applications. Our BO method achieves good coverage by incorporating a novel term in the acquisition function to penalize sample clustering. The main contributions of this work can be summarized as follows:

- This is the first work adapting Bayesian optimization into AMS design verification and failure detection (to the best knowledge of the authors);
- Our method efficiently detects the first failure region with a novel criteria function balancing between exploration and exploitation with a limited simulation budget;
- We propose a novel Bayesian optimization acquisition function which yields high coverage for detecting multiple failure regions due to different mechanisms;
- We significantly accelerate the Bayesian optimization procedure with parallelizable simulation.

Our experimental results demonstrate that Bayesian optimization is more suitable for failure detection compared to traditional statistical sampling methods, and the proposed methods can expedite the failure detection process while ensure certain coverage in detecting multiple failure regions.

3.1 Bayesian Optimization for Failure Detection

3.1.1 Failure Detection Problem Formulation

Given a D-dimension parameter space $\Omega \subseteq \mathbb{R}^D$, the performance value $y(\mathbf{x})$ at a particular variational point $\mathbf{x} \in \Omega$ of the design under verification (DUV) can be determined by circuit simulation or measurement. Without loss of generality, a point $\mathbf{x}$ can be regarded as a failure if the following condition is satisfied,

$$y(\mathbf{x}) < T, \mathbf{x} \in \Omega, \tag{3.1}$$

where T is the targeted specification (assuming the smaller the value, the worse the performance). Directly verifying (3.1) is usually infeasible since $y(\mathbf{x})$ is highly nonlinear in a high dimensional space and expensive in terms of simulation or measurement. Instead, we formulate the original problem as an optimization problem as follows,

$$\min_{\mathbf{x} \in \Omega} y(\mathbf{x}) < T. \tag{3.2}$$

Obviously, if the worst performance value can be obtained through optimization, the verification conclusion ("Pass" or "Fail") can be simply drawn by comparing the worst value with the specification target T. Bayesian Optimization is adopted here to solve the optimization problem in (3.2) considering the objective function $y(\mathbf{x})$ as a black-box objective function. Treating y(x) as a black box is a sensible choice as the mapping from $\mathbf{x}$ to y is highly complex and typically no analytical expression exists.

Note that (3.2) can just tell if the circuit fails the specification or not, but ignores coverage issue when multiple failure regions appear, which will be addressed and discussed in Section 3.3.

3.1.2 Overview of Bayesian Optimization

The Bayesian optimization approach is composed of two essential components. The first component is a statistical surrogate model of the objective function with embedded model uncertainty prediction. This surrogate model, which serves as an approximation to the black-box objective function, is gradually improved via re-training with sequentially collected additional samples. The sequential sampling procedure is guided by optimizing an acquisition function $\alpha(\mathbf{x}; \mathcal{D})$ with the objective of finding the best samples that can improve the surrogate model and guide the optimization process in the most cost-efficient manner, as shown in Fig. 3.1. The design and optimization of the acquisition function is the second key component of Bayesian optimization. The acquisition function balances between finding the worst performance value (exploitation) and exploring the undiscovered parameter space (exploration), when it comes to failure detection.

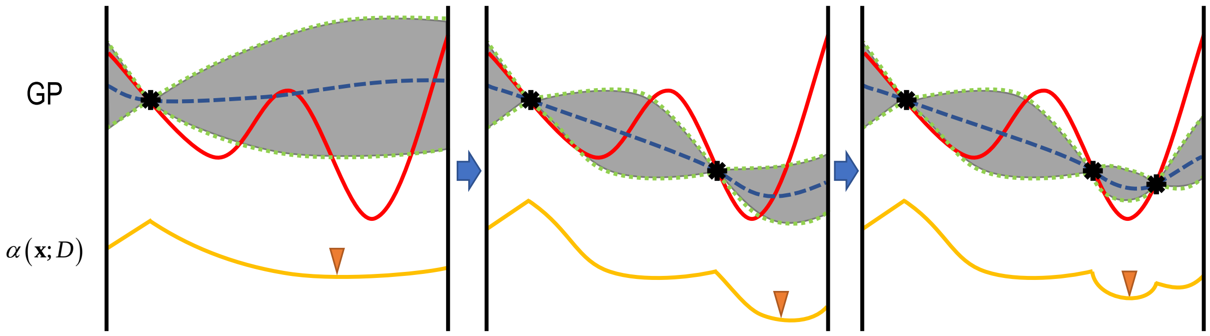


Figure 3.1: Bayesian optimization procedure example. Red line represents the objective function.

Algorithm 1 provides the detailed flow for failure detection using Bayesian optimization with an initial dataset $\mathcal{D}_0$ and a limited simulation budget of n. The initial statistical

Algorithm 1: Bayesian optimization for failure detection

```
Input  : Initial sample dataset $\mathcal{D}_0$; Simulation budget $n$;
         Objective function $y(\mathbf{x})$; Target specification $T$.
Output: Detected failure set $\mathcal{F}$.
1  Construct initial statistical model $p\left(y^* \left| \mathbf{x}^*, \ \mathcal{D}_0\right.\right)$;
2  $\mathcal{F} \leftarrow \{\}$ ;
3  for $i \leftarrow 1$ to $n$ do
4      $\mathbf{x}_i \leftarrow \arg\min_{x\in\Omega} \alpha\left(\mathbf{x}; \mathcal{D}_{i-1}\right)$;
5      $y_i \leftarrow y\left(\mathbf{x}_i\right)$;
6      if $y_i < T$ then
7          $\mathcal{F} \leftarrow \{\mathcal{F}, (\mathbf{x}_i, y_i)\}$;
8      end
9      $\mathcal{D}_i \leftarrow \{\mathcal{D}_{i-1}, (\mathbf{x}_i, y_i)\}$;
10     update statistical model $p\left(y^* \left| \mathbf{x}^*, \ \mathcal{D}_i\right.\right)$;
11 end
12 return $\mathcal{F}$
```

model is built using $\mathcal{D}_0$ and updated by augmented data $(\mathbf{x}_i, y_i)$ in each iteration, where the query point $\mathbf{x}_i$ is found in Step 4, and performance value y_i is simulated or measured in Step 5.

Gaussian Process Model

A typical choice of the statistical surrogate model is Gaussian Process (GP) model, which provides both mean and uncertainty prediction of the black-box objective function in a closed form. Given any finite collection of N samples $\mathbf{X} = [\mathbf{x}_1, \cdots, \mathbf{x}_N]$, we define variables $\mathbf{f} = \left[f\left(\mathbf{x_1}\right), \cdots, f\left(\mathbf{x}_N\right)\right]^{\mathrm{T}}$ and $\mathbf{y} = [y_1, \cdots, y_N]$ to represent the function values and noisy observations, respectively. Particularly, for design verification, $\mathbf{X}$ can represent the parameter variations like transistor channel length, $\mathbf{f}$ stands for our belief about performance value like gain or bandwidth, and $\mathbf{y}$ is the actual performance value obtained from simulation or measurement which may contain noise. In GP regression, we assume that $\mathbf{f}$ is jointly Gaussian distributed with prior mean $\mu_0\left(\mathbf{x}\right)$ and covariance $k\left(\mathbf{x}, \mathbf{x}'\right)$

functions, giving the corresponding Bayesian generative model [34][1],

$$\mathbf{f} \mid \mathbf{X} \sim \mathcal{N}(\mathbf{m}, \mathbf{K}), \tag{3.3}$$

$$\mathbf{y} \mid \mathbf{f}, \sigma^2 \sim \mathcal{N}\left(\mathbf{f}, \sigma^2 \mathbf{I}\right), \tag{3.4}$$

where the element of mean vector and covariance matrix are defined as $m_i = \mu_0(\mathbf{x})$ and $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$.

Provided the dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, and a new test point $\mathbf{x}^*$, the mean and variance prediction of the objective function at $\mathbf{x}^*$ given by the GP model can be represented as follows:

$$y^* \mid \mathbf{x}^*, \mathcal{D} \sim \mathcal{N}\left(\mu(\mathbf{x}^*; \mathcal{D}), \sigma^2(\mathbf{x}^*; \mathcal{D})\right) \tag{3.5}$$

$$\mu(\mathbf{x}^*; \mathcal{D}) = \mu_0(\mathbf{x}^*) + \mathbf{k}(\mathbf{x}^*)^{\mathrm{T}} \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} (\mathbf{y} - \mathbf{m}) \tag{3.6}$$

$$\sigma^2(\mathbf{x}^*; \mathcal{D}) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^{\mathrm{T}} \left(\mathbf{K} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{k}(\mathbf{x}^*) \tag{3.7}$$

where the element of $\mathbf{k}(\mathbf{x}^*)$ is defined as $k_i(\mathbf{x}^*) = k(\mathbf{x}^*, \mathbf{x}_i)$.

Acquisition Function

The acquisition function should be carefully designed to consider both exploitation and exploration. If the acquisition function concentrates on exploitation too much, the sequential GP model refinement process may be stuck at the local optimum of the current model, which may miss worse performance value in the space; if the acquisition function is biased towards exploration, the information in the current GP model is not well utilized in selecting the next set of sample points towards optimization of the targeted objective function, leading to slow convergence. There are a variety of acquisition

[1] $\mathbf{A} \sim \mathcal{N}(\mu, \mathbf{\Sigma})$ means random vector $\mathbf{A}$ follows multivariate normal distribution with mean vector μ and covariance matrix $\mathbf{\Sigma}$.

functions discussed in rich literatures [44, 43]. Two typical categories of the acquisition functions are improvement-based policy and optimistic policy, such as: probability of improvement (PI) as shown in (3.8) and lower confidence bound (LCB) as shown in (3.9) below.

$$\alpha_{PI}\left(\mathbf{x};\mathcal{D}_i\right) = -\Pr\left[y\left(\mathbf{x}\right) < \tau\right] = \Phi\left(\tfrac{\mu(\mathbf{x};\mathcal{D}_i)-\tau}{\sigma(\mathbf{x};\mathcal{D}_i)}\right) \tag{3.8}$$

$$\alpha_{LCB}\left(\mathbf{x};\mathcal{D}_i\right) = \mu\left(\mathbf{x};\mathcal{D}_i\right) - \beta\sigma\left(\mathbf{x};\mathcal{D}_i\right) \tag{3.9}$$

where $\Phi\left(\cdot\right)$ is the cumulative distribution function (CDF) for standard normal distribution. Although the exploitation and exploration are both touched in these two policies using the mean and uncertainty prediction of the surrogate model, the underneath philosophy is distinct. PI attempts to find the failure points with largest posterior probability of improvement inside the current surrogate model w.r.t τ, which can be defined as the target T or the worst performance value y^{-} found so far, while LCB optimistically experiments on the points w.r.t the one-sided lower confidence bound, as long as they have certain chance to appear.

3.2 Proposed Parallelizable Bayesian Optimization

3.2.1 Limitation of Traditional Bayesian Optimization

Given a limited simulation budget, the design verification demands fast and efficient failure detection with the smallest amount of required simulation/measurement data possible. The traditional Bayesian optimization approach only utilizes a single acquisition function during the entire process, which is potentially shortsighted and can jeopardize the failure detection efficiency.

A simple illustration of this limitation is shown in Fig. 3.2. Consider two different ob-

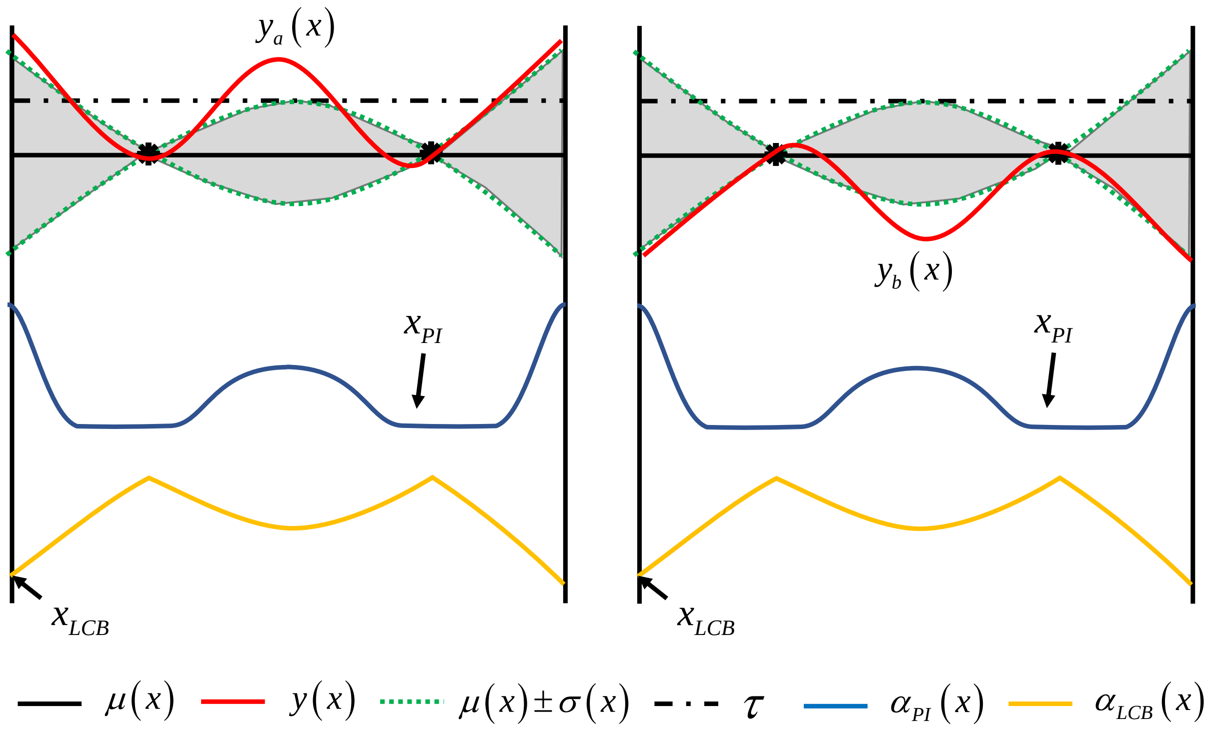


Figure 3.2: Bayesian optimization limitation with single acquisition function considered.

jective functions $y_a(x)$ and $y_b(x)$ in a bounded one-dimensional space. The two collected samples for both situations are the same, yielding the same acquisition function value for both PI and LCB methods. However, we can see that the locations to minimize PI and LCB are different, denoted as x_{PI} and x_{LCB}, respectively. If we choose LCB for $y_a(x)$ or PI for $y_b(x)$, the resulting search direction would be completely misleading. However, there is no extra information facilitating the decision of which acquisition function to use before measuring the actual points. This kind of behavior would be much harder and even infeasible to predict with more complex objective functions in a high-dimensional space, which can slow down the failure detection process significantly.

3.2.2 Parallelizable Bayesian Optimization

In order to reconcile the previous discussed limitation, we propose to optimize multiple acquisition functions simultaneously to explore the entire parameter space in a batch mode. If we consider PI and LCB at the same time to collect two samples per step for the example shown in Fig. 3.2, both optimizations for $y_a(x)$ and $y_b(x)$ can be handled more properly. Here the key idea is to explore the diversity arising from multiple acquisition functions at each sampling step of Bayesian optimization. As discussed earlier, selecting a single optimal acquisition function that works for any given optimization problem *a priori* is a very challenging task. Instead, simultaneously collecting multiple sample points based on multiple acquisition functions practically provides a more robust solution. Furthermore, this strategy can naturally explore different degrees of exploration vs. exploitation represented by the multiple acquisition functions, leading to an overall robust and efficient failure detection process.

Suppose n_b acquisition functions $\alpha_1, \ldots, \alpha_{n_b}$ are preset to be optimized inside a single batch. The statistical model is updated with n_b augmented data in each iteration. The proposed parallelizable Bayesian optimization flow is presented in Algorithm 2. Besides the efficient failure detection, the acquisition function optimization and the circuit simulation/measurement (the for-loop from Step 4 to 10) can be executed in a parallel way to expedite the overall runtime compared to the serialized simulation in Algorithm 1.

Note that there exist lots of flexibility w.r.t the acquisition function family to be chosen in this framework, which can be further studied in future work. The final acquisition function can be chosen from different policies or combine multiple acquisition functions with different weighting parameters as what we will show in the following section.

Algorithm 2: Parallelizable Bayesian optimization for failure detection

```
Input  : Initial sample dataset $\mathcal{D}_0$;
         Simulation budget $n$; Batch size $n_b$;
         Preset $n_b$ acquisition functions $\alpha_1, \ldots, \alpha_{n_b}$;
         Objective function $y(\mathbf{x})$; Target specification $T$.
Output: Detected failure set $\mathcal{F}$.
1  Construct initial statistical model $p\left(y^* \mid \mathbf{x}^*,\ \mathcal{D}_0\right)$;
2  $\mathcal{F} \leftarrow \{\}$ ;
3  for $b \leftarrow 1$ to $n/n_b$ do
4      for $i \leftarrow 1$ to $n_b$ do
5          $\mathbf{x}_{b,i} \leftarrow \arg\min_{x\in\Omega} \alpha_i\left(\mathbf{x}; \mathcal{D}_{b-1}\right)$;
6          $y_{b,i} \leftarrow y\left(\mathbf{x}_{b,i}\right)$;
7          if $y_{b,i} < T$ then
8              $\mathcal{F} \leftarrow \{\mathcal{F}, (\mathbf{x}_{b,i}, y_{b,i})\}$;
9          end
10     end
11     $\mathcal{D}_b \leftarrow \{\mathcal{D}_{b-1}, \left(\mathbf{x}_{b,1}, y_{b,1}\right), \ldots, \left(\mathbf{x}_{b,n_b}, y_{b,n_b}\right)\}$;
12     update statistical model $p\left(y^* \mid \mathbf{x}^*,\ \mathcal{D}_b\right)$;
13 end
14 return $\mathcal{F}$
```

3.2.3 Proposed Acquisition Function

A new acquisition function is obtained combining PI aiming at finding most probable failures and LCB exploring potential failures with small probabilities. Since $\Phi(\cdot)$ is monotonically increasing, minimizing $\alpha_{PI}(\mathbf{x})$ is equivalent to minimizing $\frac{\mu(\mathbf{x};\mathcal{D}_i)-T}{\sigma(\mathbf{x};\mathcal{D}_i)}$ where τ is set to be the target T. Without loss generality, T can always be regarded as 0 by shifting the entire response $y(\mathbf{x})$. Therefore, we can have the following mixed acquisition function combining LCB and PI:

$$\begin{aligned}\alpha(\mathbf{x};\mathcal{D}) &= \kappa\left[\mu(\mathbf{x};\mathcal{D}_i) - \beta\sigma(\mathbf{x};\mathcal{D}_i)\right] + (1-\kappa)\,\bar{\sigma}\frac{\mu(\mathbf{x};\mathcal{D}_i)}{\sigma(\mathbf{x};\mathcal{D}_i)} \\ &= \left[\kappa + (1-\kappa)\frac{\bar{\sigma}}{\sigma(\mathbf{x};\mathcal{D}_i)}\right]\mu(\mathbf{x};\mathcal{D}_i) - \kappa\beta\sigma(\mathbf{x};\mathcal{D}_i)\end{aligned} \tag{3.10}$$

where $\bar{\sigma}$ is the mean of the posterior uncertain prediction to normalize between two parts and κ is the weighting parameter. The previous equation can be regarded as the balance between mean and variance prediction of the current surrogate model. Assume that $\bar{\sigma}$ is comparable to most of $\sigma\left(\mathbf{x};\mathcal{D}_i\right)$, i.e, $\bar{\sigma}/\sigma\left(\mathbf{x};\mathcal{D}_i\right) \approx C$, which gives the following equation,

$$\alpha\left(\mathbf{x};\mathcal{D}\right) = \left[\kappa + (1-\kappa)\,C\right]\mu\left(\mathbf{x};\mathcal{D}_i\right) - \kappa\beta\sigma\left(\mathbf{x};\mathcal{D}_i\right) \tag{3.11}$$

Note that C depends on the variance of the model, and κ and β are hard to be determined beforehand. Parallelizable Bayesian optimization utilizes multiple acquisition functions which can cover different configurations of hyperparameters. Therefore, instead of tuning three hyperparameters, a more concise form capable of incorporating into parallelizable Bayesian optimization can be formulated as follows.

$$\begin{aligned}\alpha_i\left(\mathbf{x};\mathcal{D}_b\right) &= \alpha_{pBO}\left(\mathbf{x};\mathcal{D}_b, w_i\right)\\ &= \left(1-w_i\right)\mu\left(\mathbf{x};\mathcal{D}_b\right) - w_i\sigma\left(\mathbf{x};\mathcal{D}_b\right)\end{aligned} \tag{3.12}$$

Here $\alpha_{pBO}\left(\mathbf{x};\mathcal{D}_b, w_i\right)$ is the i-th acquisition function in b-th batch, weighted by w_i. Given n_b preset w_i, we can have n_b different acquisition functions for parallelizable Bayesian optimization, exploring the diversity of acquisition functions. The final form is very similar to LCB acquisition function, however, this new acquisition function can be optimized solely by the uncertainty of the surrogate model with $w_i = 1$, which is unavailable in LCB.

3.3 Bayesian Optimization with High Coverage Consideration

In addition to efficient failure detection, coverage is also a key perspective in design verification. Not only shall we identify the existence of (single) failure, we shall also identify different types of failure mechanisms or failure regions to provide a good coverage. A failure detection method with a good coverage property can provide a more complete picture of different types of failures involved in a given design and assist the iterative process of verification, debug, and re-design. However, utilizing the standard Bayesian optimization method to optimize for the worst performance value doesn't take coverage into account, usually resulting in clustered detected failures, as demonstrated in Section 3.4.3. To maximize the failure coverage, one approach is to consider coverage as another optimization target after the first failure is found in the process, by adding an additional term (referred to below as the high-coverage term) into the acquisition function to penalize clustered samples as follows.

$$\alpha_{pHCBO}\left(\mathbf{x};\mathcal{D}_b,w_i\right)=\alpha_{pBO}\left(\mathbf{x};\mathcal{D}_b,w_i\right)+\alpha_{HC}\left(\mathbf{x};\mathcal{D}_b,w_i\right) \tag{3.13}$$

Specifically, to avoid redundant data collection in the same failure region, the new collected point should be far away from the samples optimized by the same acquisition function in previous batches as shown in Fig. 3.3. A high-coverage term can be formulated with a normalization term N_{HC} and the geometry mean of some distance measure regarding p previous samples as follows.

$$\alpha_{HC}\left(\mathbf{x};\mathcal{D}_b,w_i\right)=N_{HC}\sqrt[p]{\prod_{j=1}^{p}f_{HC}\left(\left\|\mathbf{x}-\mathbf{x}_{b-j,i}\right\|,d\right)} \tag{3.14}$$

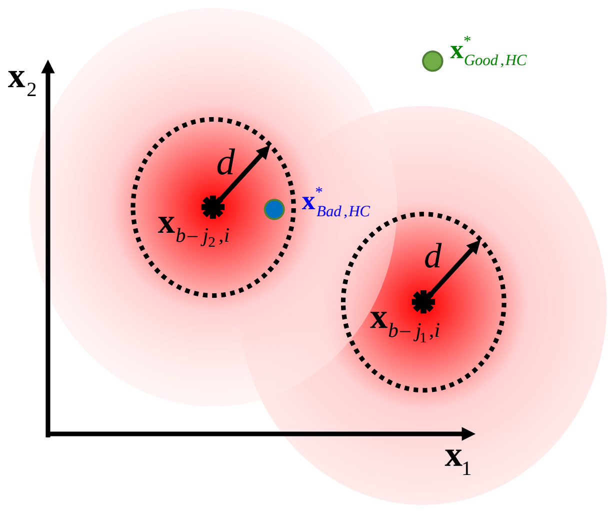


Figure 3.3: High coverage acquisition function illustration in 2-D space.

Here the normalization term N_{HC} mainly considers the weighting parameter w_i and the scale of collected samples. With $d_{\mathbf{x}} = \|\mathbf{x} - \mathbf{x}_{b-j,i}\|$ as the distance between new sample and previous sample, $f_{HC}\left(d_{\mathbf{x}}, d\right)$ intends to greatly penalize the new sample which is too close to the previous collected sample with large value (close to infinity), and encourages the new sample to be located in an unexplored area with small value (close to one). The parameter d, named as failure distance resolution, is a user-defined parameter specifying the failure region size in user's belief. If the distance between two failure samples is larger than d, the two samples are induced by different failure mechanisms in designers' perspective, which makes d an expected failure region size to distinguish different failure mechanisms. Therefore the resulting function should own the following property.

$$f_{HC}\left(d_x, d\right) \approx \begin{cases} +\infty & d_x \ll d \\ 1 & d_x \gg d \end{cases} \tag{3.15}$$

A sharp transition around d is usually desirable to clearly distinguish the failure regions w.r.t d. In particular, the following equation is used in our experiment.

$$f_{HC}\left(d_{\mathbf{x}}, d\right) = \exp\left[\left(\frac{d}{d_{\mathbf{x}}}\right)^{10}\right] \tag{3.16}$$

Note that Algorithm 1 and 2 can be used without modification when the high-coverage term is added to any acquisition function.

3.4 Experimental Results

To demonstrate the effectiveness of the proposed Bayesian optimization method, we applied it to a synthetic mathematical model, a one-stage single-ended differential amplifier, and an under-voltage lockout circuit, and compared with both statistical sampling techniques and Bayesian optimization methods with several commonly-used acquisition functions. Specifically, Monte Carlo (MC) method and Scaled-Sigma Sampling (SSS) algorithm [13, 12], a state-of-art statistical sampling technique, are chosen as compared sampling techniques. All the parameter variations follow Gaussian distribution, required by statistical sampling techniques. To maximize the possibility of hitting rare failures based on MC, uniform distribution is assumed in the bounded hyper-cube space, which encloses a $\pm 4\sigma$ parameter variation space. Expectation of Improvement (EI) [41], Probability of Improvement (PI) [47] and Lowest Confidence Bound (LCB) [41] are selected as the acquisition functions for comparison in Bayesian optimization, which are implemented in a popular toolbox called BayesOpt [48].

We considered two alternatives derived from our approach: *pBO* uses only the proposed parallelizable Bayesian optimization acquisition function balancing between exploration and exploitation as described in Section 3.2; *pHCBO* further includes the additional modification of the acquisition function for coverage of multiple failure regions as discussed in Section 3.3 after we found first failure using *pBO*. The batch size for our parallel Bayesian optimization is set to be 5. The 5 weighting parameters for the 5 corresponding α_{pBO} acquisition functions are $\mathbf{w} = (0, 0.25, 0.5, 0.75, 1)^{\mathrm{T}}$. For *pHCBO*, we set $p = 5$ in (3.14) to select 5 previous samples to avoid clustering of sequential sampling points. The distance resolution d is chosen to be 0.1 for the synthetic mathematical model, and 0.2 for the two circuit examples. The proposed methods are implemented in C++ under the BayesOpt [48] framework using DIRECT_L [49] and BOBYQA [50] in NLopt library [51] for optimization. All the experiments are conducted on a workstation with a 3.50GHz Intel(R) Xeon(R) E5-1620 v4 CPU.

3.4.1 Test Case Configuration

Synthetic Mathematical Model

A simple synthetic mathematical model having K unconnected failure regions in D dimensional space is designed with convenient configuration flexibility for failure detection as follows:

$$Y(\mathbf{x}) = \prod_{i=1}^{K} \left(\|\mathbf{x} - \mathbf{c}_i\| - r_i \right), \tag{3.17}$$

where the i-th failure region $F_i(\mathbf{c}_i, r_i)$ is defined as a hyper-ball centering at $\mathbf{c}_i$ with a range of r_i. Each failure region $F_i(\mathbf{c}_i, r_i)$ is carefully chosen to avoid overlapping with other regions. We define $\mathbf{x}$ as a failure point when $Y(\mathbf{x}) < 0$, i.e., $\mathbf{x}$ is inside one of the failure regions F_i as shown in Fig. 3.4. Here we set the model to be a 6-dimensional function with 2 failure regions whose centers are located at $\mathbf{c}_1 = (0.3, 0.4, 0.5, 0.6, 0.7, 0.8)^{\mathrm{T}}$ and $\mathbf{c}_2 = (0.5, 0.6, 0.3, 0.4,\ \ 0.3, 0.2)^{\mathrm{T}}$, respectively, each with radius $r_{1,2} = 0.1$. The two failure regions only occupy 10^{-5} volume of the entire bounded space enclosing $[0, 1]^6$, making it a hard test case for comparing different failure detection methods. For each of the considered Bayesian optimization approach, 10 samples are provided to train an initial GP model. After that, another 240 samples are sequentially collected for this verification task.

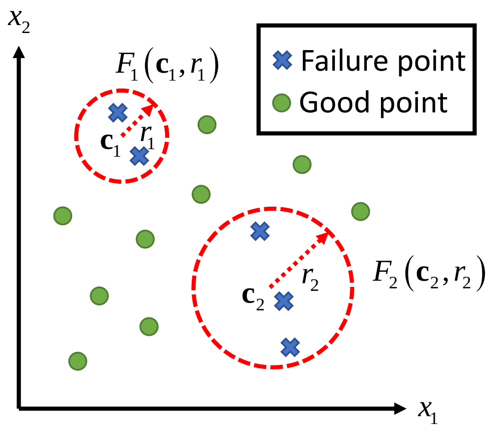


Figure 3.4: A 2D illustration for the 6-dimensional synthetic mathematical test case.

One-Stage Single-Ended Differential Amplifier

The amplifier circuit is designed using a commercial 90nm CMOS technology design kit, and simulated remotely with Synopsys HSPICE on a server with a 2.80GHz Intel(R) Xeon(R) E5-2680 v2 CPU. Three specifications, gain-bandwidth product (GBW), gain and common-mode rejection ratio (CMRR), are chosen as the verification targets for the amplifier. Three types of transistor-level variations are considered for 5 transistors in the circuit: channel length, threshold voltage and gate oxide thickness, resulting in a 15-dimensional verification problem. All experiments related to Bayesian optimization use the same 50 initial samples for the first GP model training and a simulation budget of 350 samples for sequential experiment designs later on.

CMOS Under-voltage Lockout Circuit

Another circuit example considered is the Under-voltage Lockout (UVLO) circuit [52] as shown in Fig. 3.5 which monitors the battery voltage and disconnects the battery from the load when battery voltage is detected to be lower than a certain threshold. The circuit is also designed using the commercial 90nm CMOS technology design kit, and simulated remotely with Cadence Spectre on the server with the 2.80GHz Intel(R) Xeon(R) E5-2680 v2 CPU. The variation of the turn-off threshold voltage $|\Delta V_{THL}|$ is chosen as the interested verification target for the UVLO circuits, which may undergo dramatic fluctuations even with small parametric variations, making this circuit an interesting test case. The most relevant variations that have a large impact on the threshold voltage are the resistances of three resistors in the "Hysteresis Control" part and the sizings of the 5 transistors in "Hysteresis Control" and "Inverter Pair" of the circuit, resulting in a 8-dimensional verification problem. 5 initial samples are collected as the starting points for all the Bayesian optimization experiments, and another 95 samples are simulated subsequently for failure detection.

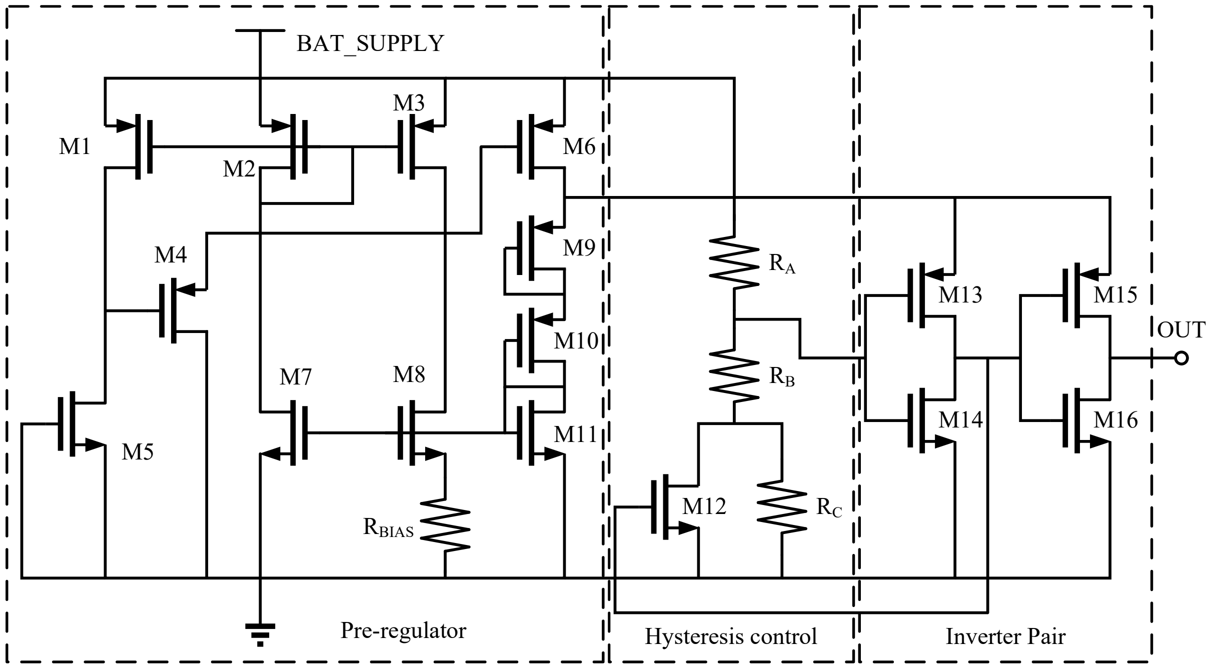


Figure 3.5: A CMOS under-voltage lockout circuit.

3.4.2 Failure Detection Effectiveness and Efficiency

Table 3.1: Failure detection result comparison for the amplifier verification with a specification of 22MHz for GBW.

Method	# Sim	Worst Case	1st Fail	# Fail.	# Reg.	Runtime
MC	600,000	28.7MHz	-	0	0	228h10m48s
SSS	6,000	37.8MHz	-	0	0	1h59m33s
EI	$50_{init} + 350_{seq}$	24.6MHz	-	0	0	15m44s
PI	$50_{init} + 350_{seq}$	25.2MHz	-	0	0	15m58s
LCB	$50_{init} + 350_{seq}$	21.9MHz	161	240	1	15m23s
pBO	$50_{init} + 5 \times 70_{batch}$	21.9MHz	**66**	228	1	**11m49s**
pHCBO	$50_{init} + 5 \times 70_{batch}$	**20.5MHz**	**66**	181	**4**	**11m40s**

Table 3.2: Failure detection result comparison for the amplifier verification with a specification of 2.5dB for Gain.

Method	# Sim	Worst Case	1st Fail	# Fail.	# Reg.	Runtime
MC	600,000	7.16dB	-	0	0	228h10m48s
SSS	6,000	13.61dB	-	0	0	1h59m33s
EI	$50_{init} + 350_{seq}$	3.75dB	-	0	0	16m01s
PI	$50_{init} + 350_{seq}$	2.17dB	282	74	5	15m53s
LCB	$50_{init} + 350_{seq}$	2.17dB	264	137	1	16m47s
pBO	$50_{init} + 5 \times 70_{batch}$	2.17dB	**109**	232	3	**11m18s**
pHCBO	$50_{init} + 5 \times 70_{batch}$	**1.80dB**	**109**	209	**10**	**11m56s**

Table 3.3: Failure detection result comparison for the amplifier verification with a specification of 10dB for CMRR.

Method	# Sim	Worst Case	1st Fail	# Fail.	# Reg.	Runtime
MC	600,000	11.42dB	-	0	0	228h10m48s
SSS	6,000	16.57dB	-	0	0	1h59m33s
EI	$50_{init} + 350_{seq}$	11.79dB	-	0	0	16m03s
PI	$50_{init} + 350_{seq}$	11.52dB	-	0	0	16m08s
LCB	$50_{init} + 350_{seq}$	11.52dB	-	0	0	16m29s
pBO	$50_{init} + 5 \times 70_{batch}$	**7.48dB**	**376**	**20**	**4**	**12m01s**
pHCBO	$50_{init} + 5 \times 70_{batch}$	**7.48dB**	**376**	**20**	**4**	**11m39s**

To compare different failure detection methods, we consider two key performance properties of these methods:*failure detection effectiveness* and *failure detection efficiency.*

Table 3.4: Failure detection result comparison for the UVLO circuit verification with a specification of 0.9V for $|\Delta V_{THL}|$.

Method	# Sim	Worst Case	1st Fail	# Fail.	# Reg.	Runtime
MC	10,000	0.77V	-	0	0	2h31m36s
SSS	2,000	0.17V	-	0	0	29m04s
EI	$5_{init} + 95_{seq}$	**0.95V**	63	26	5	4m04s
PI	$5_{init} + 95_{seq}$	0.07V	-	0	0	4m07s
LCB	$5_{init} + 95_{seq}$	0.17V	-	0	0	4m08s
pBO	$5_{init} + 5 \times 19_{batch}$	**0.95V**	**26**	49	6	**1m09s**
pHCBO	$5_{init} + 5 \times 19_{batch}$	**0.95V**	**26**	45	**8**	**56s**

The former concerns the fundamental question of whether a method can detect existence of any (rare) failure given the circuit to be verified is faulty. Furthermore, if a method is *effective*, i.e. capable of failure detection, we further consider its *efficiency* which measures the number of simulation samples required to find the first failure or a given number of failures.

Rare Failure Detection Effectiveness

Targeting at rare failure detection scenarios, we can observe from Table 3.1 to Table 3.5 that statistical sampling methods like MC and SSS provision huge simulation budgets and large amounts of CPU time (10x to 1,000x compared to most Bayesian optimization methods) while failing to find any failure for all three test cases. As a result, they provide completely misleading "optimistic" verification results. This suggests that without specifically targeting the challenges associated with rare failure detection, standard and advanced statistical sampling techniques like MC and SSS may not be adequate if only the choice of the sampling distribution is considered or optimized. On the other hand, most Bayesian optimization methods can find some failures with only hundreds of simulation budget, demonstrating the effectiveness of Bayesian optimization for rare failure detection.

Bayesian Optimization Effectiveness Comparison

Although Bayesian optimization is demonstrated to be effective in general, the performance, however, significantly depends on the design of the acquisition function. From Table 3.1 to Table 3.4, traditional acquisition functions like PI, EI and LCB fail to detect failure effectively for several specifications. No such method can identify failures for all considered specifications, showing the limitations of Bayesian optimization claimed in Section 3.2.1. However, the proposed *pBO* and *pHCBO* can detect failures in all scenarios within a very limited simulation budget, thanks to the design of multiple acquisition functions.

Bayesian Optimization Failure Detection Efficiency

For the design specifications where all Bayesian optimization methods can detect failure, the proposed *pBO* and *pHCBO* are the most efficient in terms of required simulation data for the first failure detection and almost always the most efficient measured by the total number of failures found, as shown in "1st Fail" and "# Fail." columns in Table 3.1 to Table 3.4. "1st Fail" is the number of samples collected before the first failure point is found during the process. Fig. 3.6 shows the worst amplifier gain performance reached by each Bayesian optimization method as the search (optimization) process proceeds. These results demonstrate that by simultaneously considering multiple acquisition functions weighting exploitation and exploration differently the proposed methods can efficiently identify increasingly worse performance levels as the optimization process continues, leading to the overall improved efficiency of failure detection.

In terms of runtimes shown in Table 3.1 to Table 3.4, the proposed methods show advantages over other Bayesian optimization methods due to their amenability to parallel data acquisition via circuit simulation parallelization. Clearly, as the simulation

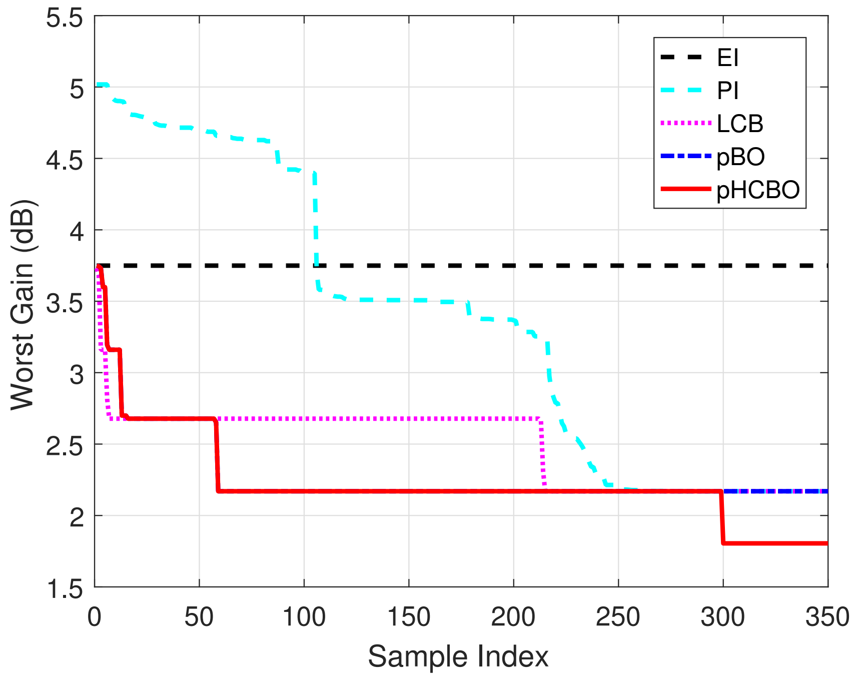


Figure 3.6: The worst amplifier gain performance reached during the Bayesian optimization process.

data collection gets increasingly expensive for larger circuits, the speedup benefit of the proposed parallelizable Bayesian optimization will be more significant, for instance in the case of large industrial AMS designs, for which a single simulation run can easily take up hours or days.

3.4.3 High Coverage Failure Detection with Multiple Failure Regions

In order to clearly demonstrate the superiority of the proposed high coverage acquisition function, first, all methods are compared based on the synthetic mathematical model whose failure locations are known *a priori*. Table 3.5 reports that *pHCBO* is the only method detecting failures in both $F_1(\mathbf{c}_1, r_1)$ and $F_2(\mathbf{c}_2, r_2)$ regions each of which contains a very small relative volume, whereas other Bayesian optimization methods can

Table 3.5: Failure region coverage for the synthetic mathematic model.

Method	# Sim	$F_1(\mathbf{c}_1, r_1)$	$F_2(\mathbf{c}_2, r_2)$
MC	50,000	0	0
SSS	10,000	0	0
EI	$10_{init} + 240_{seq}$	31	0
PI	$10_{init} + 240_{seq}$	201	0
LCB	$10_{init} + 240_{seq}$	39	0
pBO	$10_{init} + 5 \times 48_{batch}$	0	31
pHCBO	$10_{init} + 5 \times 48_{batch}$	**20**	**23**

only find one of the two failure regions.

Now, we move onto the failure region coverage results for the two circuit test cases shown in Table 3.1 to Table 3.4. Purely for the ease of counting and comparison, we bisect the entire parametric variation space into two parts based on the nominal value of every parameter: one where the parametric value is above the nominal and the other where the parametric value is below the nominal. As such, for example, the 15-dimensional parametric space of the amplifier verification problem is divided into $2^{15} = 32768$ different parametric regions. Whenever a failure sample is detected inside a region, the region is considered as a failure region. Clearly seen in Table 3.1 to Table 3.4, *pHCBO* provides more failure region coverage as it detects the largest numbers of failure regions compared to other methods. Furthermore, *pHCBO* covers all failure regions which are detected by other methods w.r.t all four different specifications in the amplifier and UVLO circuit.

We briefly comment on the types of failures detected by *pHCBO*. Without any prior knowledge of the circuit, *pHCBO* recognizes that the mismatch between differential pairs and current mirror would cause large performance degradation, which forms certain failure regions. *pHCBO* also identifies particular combinations of resistor variations to fail the $|\Delta V_{THL}|$ spec for the UVLO circuit.

3.4.4 Additional Comparison between *pBO* and *pHCBO*

A more detailed comparison between *pBO* and *pHCBO* is presented in Fig. 3.7 and Fig. 3.8 to illustrate the sequential batch query process for the verification of the differential amplifier gain. With larger weight w, the new query point is more likely to jump out of the current local optima to explore the part of the undiscovered parametric space. In comparison to *pBO*, *pHCBO* attempts to avoid large penalty resulted from the high coverage acquisition function when the query point is close to previous collected samples. Hence, it has a tendency to minimize redundant data sampling that discovers the worst failure case in the same failure region, spreading out failure detection over the entire parametric space and driving the search process towards distinct failure regions. The two spikes in Fig. 3.7 are caused by a trivial strategy introducing random samples when detecting too many previous samples trapped in the same local, however, this phenomenon doesn't get alleviated without high coverage consideration in acquisition functions. Moreover, *pHCBO* is even capable of detecting worser cases compared to *pBO* since it covers more failure regions, as shown in Table 3.2.

3.5 Summary

In this chapter, we develop a novel Bayesian optimization procedure for rare failure detection of analog/mixed-signal circuits. In particular, we design a new acquisition function for Bayesian optimization to make the procedure particularly effective for multiple failure region detection. Our experiments show that the proposed Bayesian optimization procedure outperforms existing statistical sampling techniques in terms of both detection effectiveness and efficiency. The proposed procedure can find rare design failures much more efficiently and can discover a large number of failure mechanisms with much less required sampling data.

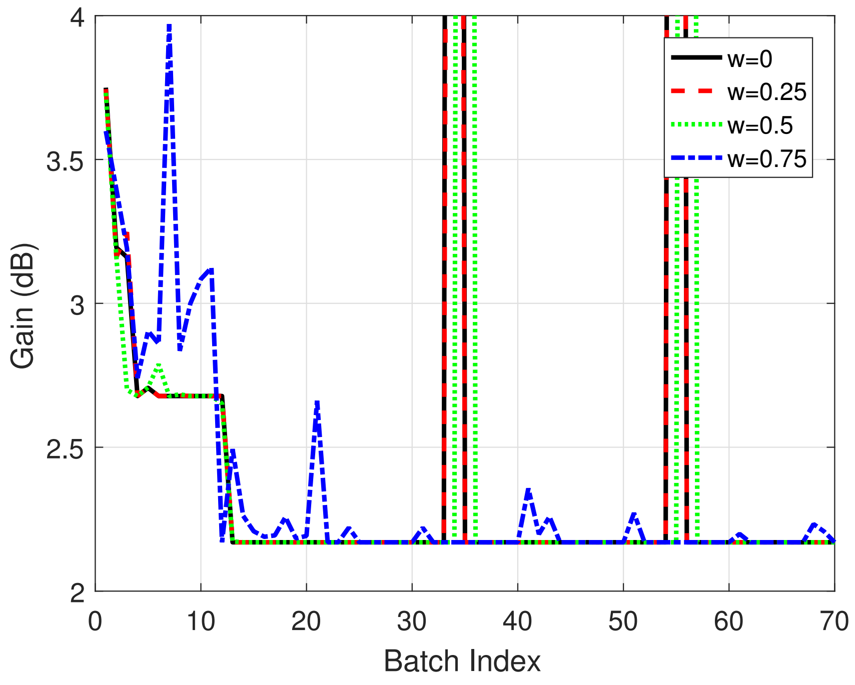


Figure 3.7: Amplifier gain query points for each batch using *pBO*. The fifth point ($w = 1$) inside each batch (based solely on uncertainty measure) is excluded due to its large response scale for clarity of visualization.

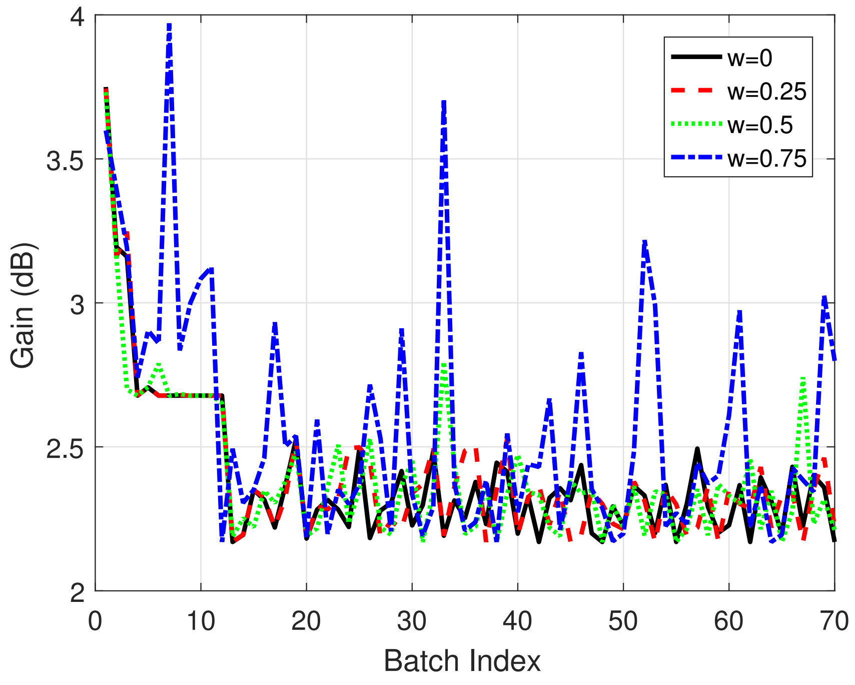


Figure 3.8: Amplifier gain query points for each batch using *pHCBO*. The fifth point ($w = 1$) inside each batch (based solely on uncertainty measure) is excluded due to its large response scale for clarity of visualization.

Chapter 4

Enabling High-Dimensional Bayesian Optimization

While providing an attractive black-box solution applicable to AMS verification, a well-known limitation of Bayesian optimization is its limitation in dealing with high-dimensional problems [44, 53]. When the dimensionality of the black-box optimization problem increases, so does the dimensionality of the optimization of the acquisition function, which is typically non-convex, at each sequential sampling step. Solving high-dimensional optimization problems can be both computationally expensive and hard. The high run-time cost and degradation of optimization solution quality for high-dimensional problems severely limit the scalability of BO.

This chapter aims to extend the applicability of BO to the challenging problem of rare

©2019 IEEE. Reprinted, with permission, from Hanbin Hu, Peng Li, and Jianhua Z. Huang, "Enabling High-Dimensional Bayesian optimization for Efficient Failure Detection of Analog and Mixed-Signal Circuits", ACM/IEEE Design Automation Conference (DAC), 2019.

©2021 IEEE. Reprinted, with permission, from Myung Seok Shim, Hanbin Hu, and Peng Li, "Reversible Gating Architecture for Rare Failure Detection of Analog and Mixed-Signal Circuits", ACM/IEEE Design Automation Conference (DAC), 2021.

failure detection of AMS circuits with large numbers of design uncertainties. First, we propose to employ *random embedding* [54] to effectively reduce the effective dimensionality of the verification problem. Dimensionality reduction is possible for AMS circuits since under many practical situations variational parameters of a circuit do not have equal significance to a given design performance to be verified [36, 37]. Specific circuit topologies employed in practical circuits build constrained structures into the way different circuit/process parameters interact with each other and influence the given design performance. This gives rise to parameters that are statistically insignificant to the targeted performance. It shall be noted, however, such parametric redundancy in practice may be only identified in a transformed parameter space. Towards this end, random embedding provides a systematic way to explore hidden parametric redundancy. As such, parameter redundancy needs not to be specified by the designer *a prior*, which is very hard in general. Instead, it can be streamlined in the sequential statistical learning/black-box optimization framework of Bayesian optimization.

Moreover, to also handle nonlinear manifold, we also propose a RevNet based gating neural network with the improved performance for the rare failure detection problem using the BO framework. Our main contributions are: **1)** propose a new RevNet based auxiliary-model regulated gating architecture, called *Rev-Gate*, to utilize gating fusion weights for efficient dimension reduction; **2)** propose a novel dimension embedding method using RevNet and Bayesian neural network (BNN) to embed low-dimensional nonlinear internal representation back into the high-dimensional original variation parameters; and **3)** investigate the proposed dimension embedding in a BO framework for efficient rare failure detection via extensive experimental studies. We demonstrate in the experimental study that our proposed Rev-Gate architecture efficiently detects rare AMS failures with significantly less runtime while other methods don't.

4.1 Bayesian Optimization Challenges under High Dimensional Space

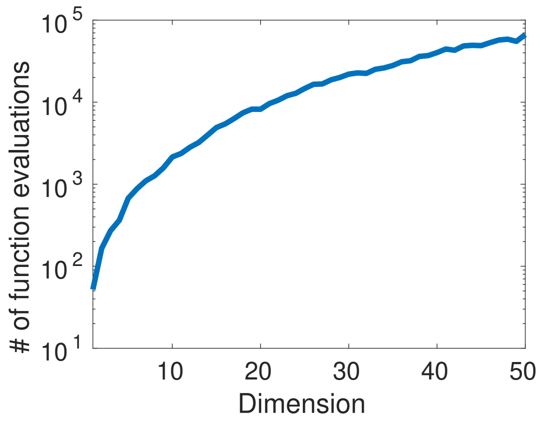
A global optimization method assisted with a local gradient-free optimizer is usually to optimize the D-dimensional acquisition function. Such methods often suffer severely from the curse of dimensionality. We tested the optimization efficiency of DIRECT_L [49] and COBYLA [55] from the NLopt library [51] on a simple objective function:

$$y_{syn}\left(\mathbf{x}\right) = \frac{\left\|\mathbf{x} - \mathbf{c}\right\|_2}{\left\|\mathbf{c}\right\|_2}, \tag{4.1}$$

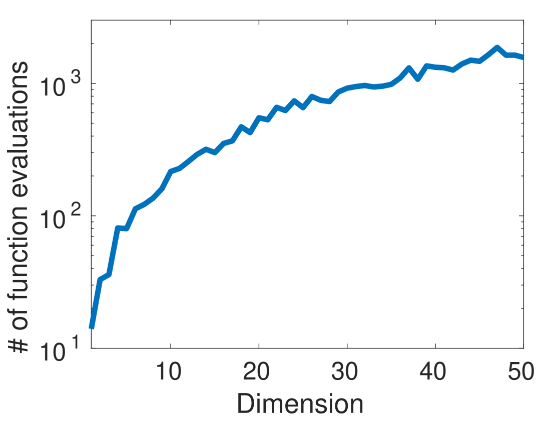
where c is a D-dimensional vector. Fig. 4.1 shows that the required number of function evaluations for both methods is super-linear in D. This suggests that in general when BO is applied to a black-box function the number of acquisition function evaluations can be much larger than D. The time complexity for evaluating simple acquisition functions like PI, EI and LCB is $O\left(N^2 + ND\right)$, where N is the number of training examples. Therefore, the time complexity for optimizing the acquisition once is greater than $O\left(N^2D + ND^2\right)$ which is quadratic in D at minimum. Optimizing general non-convex acquisition functions in high-dimensions can be challenging. To force the completion, the number of acquisition function evaluations is upper bounded, leading to poor optimization quality. In addition, hyper-parameter tuning for GP models also suffers from high dimensionality.

4.2 High Dimensional Bayesian Optimization

As stated in the previous section, the traditional BO suffers from costly GP training and poor optimization quality of the acquisition function over high-dimensional space, which leads to inefficient black-box optimization. One way to mitigate this effect is to



(a) DIRECT_L



(b) COBYLA

Figure 4.1: Number of function evaluations per optimization for two optimization methods.

embed the original high dimensional space $\mathcal{X} \subseteq \mathbb{R}^D$ into a low dimension space $\mathcal{Z} \subseteq \mathbb{R}^d$, where $d < D$, so that both surrogate model and acquisition function can be performed in a low dimensional space $\mathcal{Z}$ for fast training convergence and better acquisition function optimization. After optimized $\mathbf{z}^*$ is extracted from the acquisition function, it can be embedded back to the original space $\mathcal{X}$ for the actual circuit simulation as shown in Fig. 4.2 via a dimension embedding process $\mathbf{x}^* = E_{\mathcal{Z}\to\mathcal{X}}(\mathbf{z}^*)$.

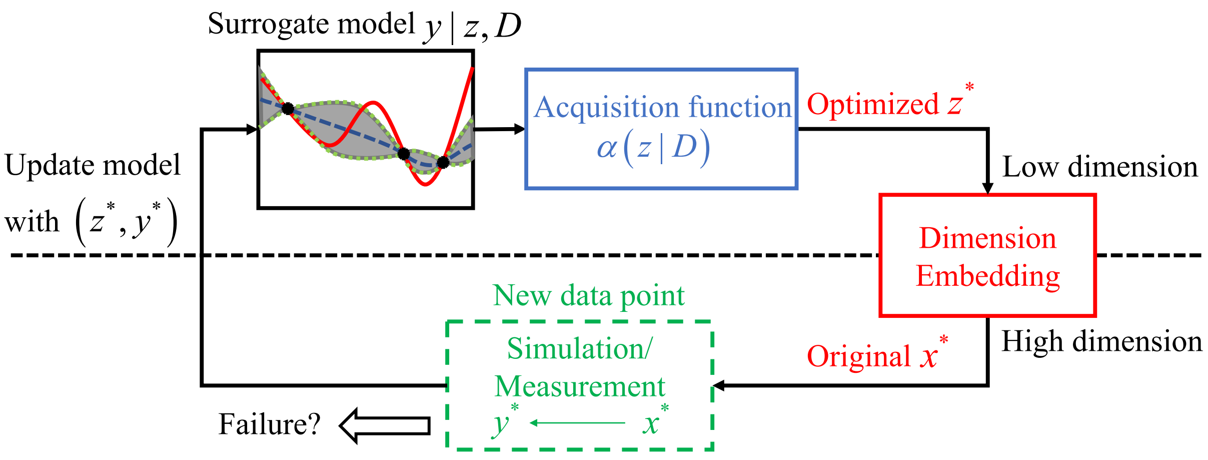


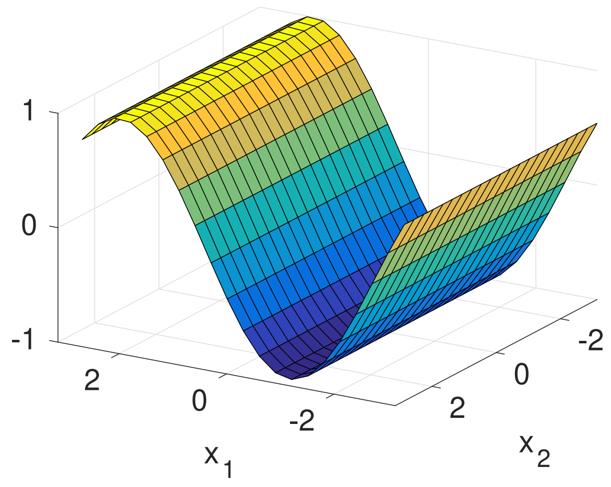
Figure 4.2: Bayesian optimization for high-dimensional problems.

Here we refer to this scheme as high dimensional Bayesian optimization (HDBO).

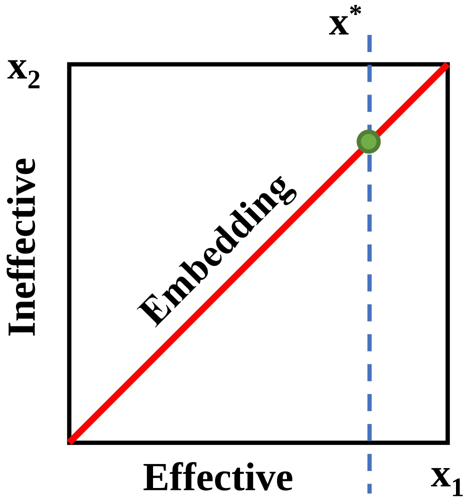
4.3 Random Embedding for HDBO

Our experimental studies in Section 4.1 have shown that the degradation of optimization solution quality and high time complexity of high-dimensional AMS circuits can make BO fail to detect rare design failures. We address this challenge by exploring random embedding to effectively reduce the dimensionality motivated by the fact typically only a subset of circuit parameters and parameter combinations have a significant impact on a target design performance.

4.3.1 Dimension Reduction: Random Embedding



(a) 2D function surface



(b) 1D embedding

Figure 4.3: Random embedding illustration.

Consider that the original D-dimensional parameter space has a d_e-dimensional effective linear subspace $\mathcal{V}$ such that for all $\mathbf{x}_e \in \mathcal{V}$ and $\mathbf{x}_u \in \mathcal{V}^{\perp}$, we have $y\left(\mathbf{x}_e + \mathbf{x}_u\right) = y\left(\mathbf{x}_e\right)$, where d_e is the minimum integer number satisfying this property. Intuitively, parametric variations in the subspace orthogonal to $\mathcal{V}$ with the lowest possible dimensionality d_e does not alter the performance value. As proven in [54], with a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$

with entries independently sampled according to $\mathcal{N}(0,1)$, where embedding dimension $d \geq d_e$, for $\forall \mathbf{x} \in \mathbb{R}^D$, there exists a $\mathbf{z} \in \mathbb{R}^d$ such that $y(\mathbf{x}) = y(\mathbf{Az})$ with probability 1. Therefore, the original high dimensional space can be embedded into a low dimensional space via a random matrix, resulting in a low dimension search space in Bayesian optimization. The optimum solution $\mathbf{x}^* \in \mathbb{R}^D$ can be found at some point $\mathbf{z}^* \in \mathbb{R}^d$, where $\mathbf{x}^* = \mathbf{Az}^*$.

For example, the 2D objective function in Fig. 4.3 only depends on x_1. The 2D parameter space can be embedded into a 1D space (red solid line) along which the optimum solution can be found.

4.3.2 Proposed BO with Random Embedding

We define the failure search region for $\mathbf{z}$ as $\mathcal{Z} \subseteq \mathbb{R}^d$. Typically, the normalized failure search space Ω for $\mathbf{x}$ can be set as a bounded hyper-cube $[-1,1]^D$. The exact mapping of Ω in the embedding subspace may be complex, but can be well approximated by another bounded hyper-cube $[-\sqrt{d}, \sqrt{d}]^d$ [54]. Now BO can operate in the low d-dimensional space defined by random embedding: both GP modeling and optimization of the acquisition function take place in terms of $\mathbf{z}$. The sampling of training data for the GP model is confined in $\mathcal{Z}$. Each sampled $\mathbf{z} \in \mathcal{Z}$ is mapped to a $\mathbf{x} \in \Omega$ via the random matrix $\mathbf{A}$ by:

$$\mathbf{x} = p_\Omega(\mathbf{Az}). \tag{4.2}$$

In case that $\mathbf{Az}$ locates outside Ω, the projection operation $p_\Omega(\cdot)$ is performed to constrain the mapped $\mathbf{x}$ within Ω. Then the circuit performance at $\mathbf{x}$ is obtained using circuit simulation.

We summarize our Bayesian optimization algorithm using both random embedding technique and parallelizable acquisition function (3.12) as shown in Algorithm 3. With

Algorithm 3: Proposed Bayesian optimization for failure detection in high dimension space

```
Input  : Original function dimensionality $D$;
         Initial sample dataset $\mathcal{D}_0$;
         Simulation budget $n$; Batch size $n_b$;
         Preset $n_b$ weighting parameters $w_1, \ldots, w_{n_b}$;
         Objective function $y(\mathbf{x})$; Target specification $T$.
Output: Detected failure set $\mathcal{F}$.
1  Select an embedding dimension d from $\mathcal{D}_0$;
2  Sample a random matrix $\mathbf{A} \in \mathbb{R}^{D \times d}$;
3  Build the initial statistical model $p\left(y^* \mid \mathbf{z}^*,\ \mathcal{D}_0\right)$;
4  $\mathcal{F} \leftarrow \{\}$ ;
5  for $b \leftarrow 1$ to $n/n_b$ do
6      for $i \leftarrow 1$ to $n_b$ do
7          $\mathbf{z}_{b,i} \leftarrow \arg\min_{z \in \mathcal{Z}} \alpha_{pBO}\left(\mathbf{z}; \mathcal{D}_{b-1}, w_i\right)$;
8          $y_{b,i} \leftarrow y\left(p_\Omega\left(\mathbf{A}\mathbf{z}_{b,i}\right)\right)$;
9          if $y_{b,i} < T$ then
10             $\mathcal{F} \leftarrow \{\mathcal{F}, \left(p_\Omega\left(\mathbf{A}\mathbf{z}_{b,i}\right), y_{b,i}\right)\}$;
11         end
12     end
13     $\mathcal{D}_b \leftarrow \{\mathcal{D}_{b-1}, \left(\mathbf{z}_{b,1}, y_{b,1}\right), \ldots, \left(\mathbf{z}_{b,n_b}, y_{b,n_b}\right)\}$;
14     Update statistical model $p\left(y^* \mid \mathbf{z}^*,\ \mathcal{D}_b\right)$;
15 end
16 return $\mathcal{F}$.
```

the proposed algorithm, the acquisition function optimization is executed in a low dimension space $\mathcal{Z} \subseteq \mathbb{R}^d$, which can be expected to have better optimization quality and efficiency. In addition, the GP model is trained under the low-dimensional space as well, resulting in more efficient GP training and evaluation.

4.3.3 Embedding Dimensionality Selection

Algorithm 4: Proposed embedding dimension selection.

```
   Input  : Initial sample dataset $\mathcal{D}_0 = \{\mathbf{X}, \mathbf{y}\}$;
            Original function dimensionality $D$;
            Random matrix maximum trial count $T$.
   Output: Embedded dimension $\tilde{d}$.
1  for $d \leftarrow 1$ to $D$ do
2      for $i \leftarrow 1$ to $T$ do
3          Sample a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$;
4          $\mathbf{A}^{\dagger} \leftarrow \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}$;
5          Build statistical model $p\left(y^* \left| \mathbf{z}^*, \left\{\mathbf{A}^{\dagger}\mathbf{X}, \mathbf{y}\right\}\right.\right)$;
6          Compute $mse_i$ of the model given $\left\{\mathbf{A}^{\dagger}\mathbf{X}, \mathbf{y}\right\}$;
7      end
8      $MSE_d \leftarrow \frac{1}{T}\sum_{i=1}^{T} mse_i$;
9  end
10 Pick the smallest $\tilde{d}$ where MSE stops decreasing from the plot using
     $\{MSE_1 \cdots MSE_D\}$;
11 return $\tilde{d}$.
```

While [54] provides the general theoretical principle of random embedding, it does not offer guidance for finding the effective dimensionality d_e. Selection of the embedding dimension d must balance two conflicting needs. An overly small d can lead to over-compression of the original parameters $\mathbf{x}$ and hence poor accuracy of the surrogate GP model, jeopardizing the robustness of failure detection. On the other hand, if d is too large, we can barely benefit from the dimension reduction brought by random embedding.

We propose the following data-efficient approach to select the embedding dimensionality prior to the BO based failure detection. For this, we collect a small training dataset to train multiple GP models with varying embedding dimensionalities. To share the same training dataset for all such GP models, the sampling of the training dataset takes place in the original D-dimensional parameter space, and the labels (circuit performance values) are queried using circuit simulation. Then, each sampled vector $\mathbf{x} \in \mathbb{R}^D$ is mapped to the corresponding vector $\mathbf{z} \in \mathbb{R}^d$ with embedding dimension d via pseudo inverse of the random embedding:

$$\mathbf{z} = \mathbf{A}^{\dagger}\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{x}. \tag{4.3}$$

The above procedure maps one common training dataset in $\mathbf{x}$ to a training set for each embedding dimension d such that a GP model with dimension d can be trained using the mapped data. We then use the mean-square error (MSE) to evaluate each GP model. If the dimensionality d is smaller than the unknown effective dimension d_e, we expect the MSE of the corresponding GP model would be large. We track the the variation of MSE as d increases. If the MSE stops decreasing at some dimension $\tilde{d}$, $\tilde{d}$ is likely to be just somewhat greater than d_e, and hence a good choice as the embedding dimension used for the sequential BO process. Since we only want to use a small amount of data to determine $\tilde{d}$, multiple, say T, GP models with different random matrices are trained for each d and their MSEs are averaged to minimize the variance of random embedding with small data. Embedding dimension selection is summarized in Algorithm 4.

4.4 Experimental Results for Random Embedding

4.4.1 Experimental Setups

Two of the same circuits: an under-voltage lockout circuit [52] (19 dimensions) and a low-dropout regulator [39] (60 dimensions), are used here, as shown in Fig. 3.5 and

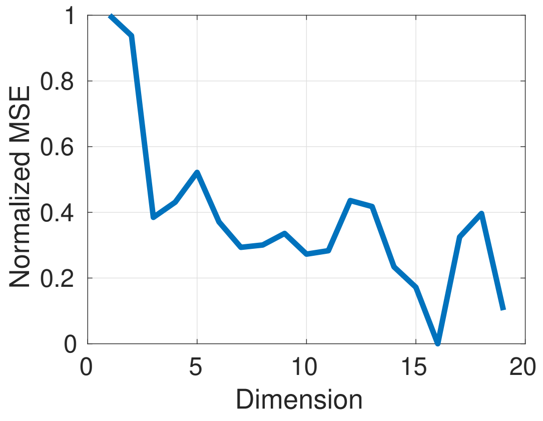
2.6, respectively. Both circuits are designed using a commercial 90nm CMOS technology design kit, and simulated remotely on the server with the 2.80GHz Intel(R) Xeon(R) E5-2680 v2 CPU.

The performances of two categories of techniques, i.e. sampling methods and Bayesian optimization, are studied. Under the first category, we employ Monte Carlo (MC) method and Scaled-Sigma Sampling (SSS) algorithm [13, 12], a state-of-the-art statistical sampling technique. The parameter variations of interest are bounded inside a large hyper-cube, which encloses a wide $\pm 4\sigma$ range for each parameter. To maximize the possibility of hitting rare failures within the large hyper-cube, uniform sampling distribution is adopted for MC. In the second category, Bayesian optimization approaches using different acquisition functions EI, PI, LCB [44] and the parallelizable multi-acquisition functions (pBO) are selected to compare the proposed BO approach with random embedding. The BO methods were implemented in C++ under the BayesOpt [48] framework using DIRECT_L [49] for global optimization and COBYLA [55] for local optimization in NLopt library [51]. All the experiments were conducted on a workstation with a 3.50GHz Intel(R) Xeon(R) E5-1620 v4 CPU.

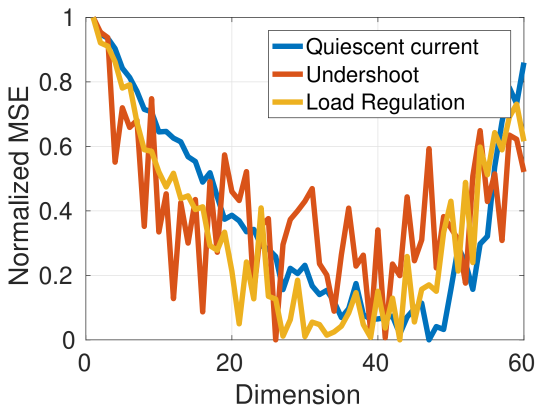
For the UVLO circuit, 5 initial samples are collected as the starting points for all the Bayesian optimization experiments, and another 95 samples are subsequently collected for failure detection. For the amplifier, all experiments related to Bayesian optimization use the same 50 samples for the first GP model training and a simulation budget of 350 examples for the sequential experiment design later on.

4.4.2 Random Embedding Dimension Selection

To pick the embedding dimension $\tilde{d}$, Algorithm 4 is performed for both circuits. For this, 5 initial examples are used for the UVLO circuit, and 50 for the LDO. The GP



(a) UVLO



(b) LDO

Figure 4.4: Random embedding dimension selection results.

model accuracy corresponding to various dimensions is presented in Fig.4.4, where the MSE results are normalized into the range of $[0, 1]$ for demonstration convenience. For the UVLO circuit, the minimum MSE is achieved at dimension 16, which however does not bring in much benefit from dimension reduction. Instead, we pick $\tilde{d}_{UVLO} = 8$, a good tradeoff between model accuracy and dimension reduction. For all the three specifications of the LDO, the MSE reaches the minimal level around dimension 30, therefore we set $\tilde{d}_{LDO} = 30$.

4.4.3 Failure Detection Effectiveness and Efficiency

Table 4.1: Failure detection result comparison for the UVLO circuit verification (19 dimension).

Spec	Target	Method	# Sim	Worst Case	1st Fail	Runtime
		MC	20,000	0.86V	-	4h22m07s
		SSS	1,000	0.15V	-	13m24s
		EI	$5_{init} + 95_{seq}$	0.16V	-	8m30s
$\lvert\Delta V_{THL}\rvert$	0.9V	PI	$5_{init} + 95_{seq}$	0.04V	-	7m56s
		LCB	$5_{init} + 95_{seq}$	0.17V	-	7m40s
		pBO	$5_{init} + 5 \times 19_{batch}$	0.14V	-	9m01s
		This work	$5_{init} + 5 \times 19_{batch}$	**0.95V**	**26**	**5m32s**

As shown in Tables 4.1 and 4.2, the MC and SSS methods collect thousands to hundreds of thousands of simulation examples without detecting a single failure. This also indicates that the failures in these two circuits are extremely rare. Meanwhile, traditional acquisition functions like EI, PI and LCB or parallelizable Bayesian optimization (pBO) method cannot detect a single failure as well due to the inherent difficulty in applying BO in high-dimensional spaces. The proposed failure detection approach is the only method detecting failures for all specifications. The worst-case performance levels found by our method are much worse than the given target, while the statistical sampling methods and the more conventional Bayesian optimization methods are overly optimistic.

Moreover, the number of simulation runs required by the proposed methods is much less than others. As presented in Tables 4.1 and 4.2, we only need 26 simulation data points to discover the first failure inside the 19-dimensional space for the UVLO circuit and hundreds of samples to detect the first failures in the 60-dimensional space for the LDO. The large reduction of simulation data brought by the proposed technique can be even more significant for rare failure detection of larger and more complex AMS circuits for which transistor-level simulation can be prohibitively expensive.

The runtime reported in Table 4.1 and 4.2 is the total runtime for Algorithm 3 including circuit simulation in a single thread configuration, i.e., no parallel mechanism is activated, which offers a clearer view of the runtime reduction provided by random embedding technique. As described earlier, since the original high-dimensional parameter space is embedded into a space of a lower dimensionality, the Gaussian process model can be trained at a much reduced cost, which speeds up both its posterior distribution evaluation and hyper-parameter tuning. Meanwhile, since the optimization of the acquisition function is also executed in the lower-dimensional space, the quality of optimization is improved and the number of function evaluations is greatly reduced, resulting in significantly less runtime compared to other Bayesian optimization methods. The runtime

Table 4.2: Failure detection result comparison for the LDO regulator verification (60 dimension).

Spec	Target	Method	# Sim	Worst Case	1st Fail	Runtime
		MC	649,000	11.6mA	-	160h25m12s
		SSS	6,000	8.2mA	-	1h38m41s
		EI	$50_{init} + 350_{seq}$	7.0mA	-	6h46m13s
Quiescent current	12mA	PI	$50_{init} + 350_{seq}$	7.2mA	-	6h00m23s
		LCB	$50_{init} + 350_{seq}$	8.0mA	-	6h33m42s
		pBO	$50_{init} + 5 \times 70_{batch}$	7.0mA	-	8h03m19s
		This work	$50_{init} + 5 \times 70_{batch}$	**12.7mA**	**231**	**1h57m05s**
		MC	649,000	0.39V	-	160h25m12s
		SSS	6,000	0.19V	-	1h38m41s
		EI	$50_{init} + 350_{seq}$	0.20V	-	6h45m11s
Undershoot	0.40V	PI	$50_{init} + 350_{seq}$	0.20V	-	6h51m05s
		LCB	$50_{init} + 350_{seq}$	0.18V	-	6h23m55s
		pBO	$50_{init} + 5 \times 70_{batch}$	0.14V	-	7h51m14s
		This work	$50_{init} + 5 \times 70_{batch}$	**0.51V**	**87**	**2h01m52s**
		MC	649,000	47.2%	-	160h25m12s
		SSS	6,000	22.6%	-	1h38m41s
		EI	$50_{init} + 350_{seq}$	25.1%	-	6h49m51s
Load regulation	50.0%	PI	$50_{init} + 350_{seq}$	9.2%	-	6h35m03s
		LCB	$50_{init} + 350_{seq}$	28.4%	-	6h38m14s
		pBO	$50_{init} + 5 \times 70_{batch}$	17.3%	-	7h50m02s
		This work	$50_{init} + 5 \times 70_{batch}$	**55.0%**	**302**	**1h55m14s**

for Algorithm 4 is typically less than one minute with small sample size, which can be ignored compared to expensive simulation cost.

4.5 Rev-Gate based Bayesian Optimization

Even though the random embedding method reduces the BO dimension effectively, there still exist several major concerns for this method. Firstly, since random embedding is agnostic to the black-box function under optimization, it is hard to decide the low dimension d for random matrix generation beforehand. In addition, the dimension embedding quality is unknown before the actual BO process. Secondly, the random embedding only performs the dimension embedding in a linear manner, which cannot be utilized when a non-linear low dimensional manifold is desired. Finally, it provides no information about the actual important variational parameters, which is essential in aspect of failure detection field for circuit designers to gain more insights about the circuit behavior.

To tackle the challenges introduced by random embedding in BO for high-dimensional failure detection as mentioned in the previous section, we propose a Rev-Gate architecture for the dimension embedding in the BO framework, which incorporates the RevNet and the ARGate to effectively identify important variational parameters and reduce the dimension through the reversibility. In order to learn the low-dimensional manifold property from the black-box function under optimization, we pre-train the Rev-Gate architecture before the BO process by using a small amount of data, which extracts the important feature information and helps choose effective low dimension d for surrogate model construction and acquisition function optimization. During the BO process, the trained RevNet performs the dimension embedding in Fig. 4.2, recovering the low dimensional point z^* to the original input space x^* for actual circuit simulation via a restoration scheme, which will be further discussed in Section 4.6. The rest of this section mainly talks about how to identify important features for BO dimension embedding via Rev-Gate pre-training.

4.5.1 ARGate for important feature extraction

For AMS failure detection under high-dimensional space, typically there exists certain redundancy for the variational parameters under consideration, and only a small number of them are critical to the final circuit performance. With only important features utilized and inessential ones removed, the circuit performance can still be predicted nicely via the surrogate model even with small amount of training data. To efficiently identify the important variational parameters, we adopt the ARGate [56] using gating architecture to switch off inessential feature via fusion weights.

In terms of network structure, the ARGate is composed of two networks as shown in Fig. 4.5: a main model and an auxiliary (aux) model. The fusion weights are extracted from the grey box (denoted as "Fusion Weight Extraction" in Fig. 4.5) in the main model, where the fusion happens with pre-processed features after a fully connected (FC) layer in each feature path.

The key idea of the ARGate is that the importance of features is represented via the fusion weights. In the main model, the output of each FC layer on each feature path are multiplied with the corresponding fusion weights to obtain a weighted internal representation, which is then passed to later network layers to get final classification/regression output. These weights are normalized between $[0, 1]$ for the feature importance interpretability. For example, assume that there are only four features under consideration. If the first feature is the only important feature in the datasets, the corresponding first fusion weight $FW1$ is the largest fusion weight out of four, which is close to 1. Then, then

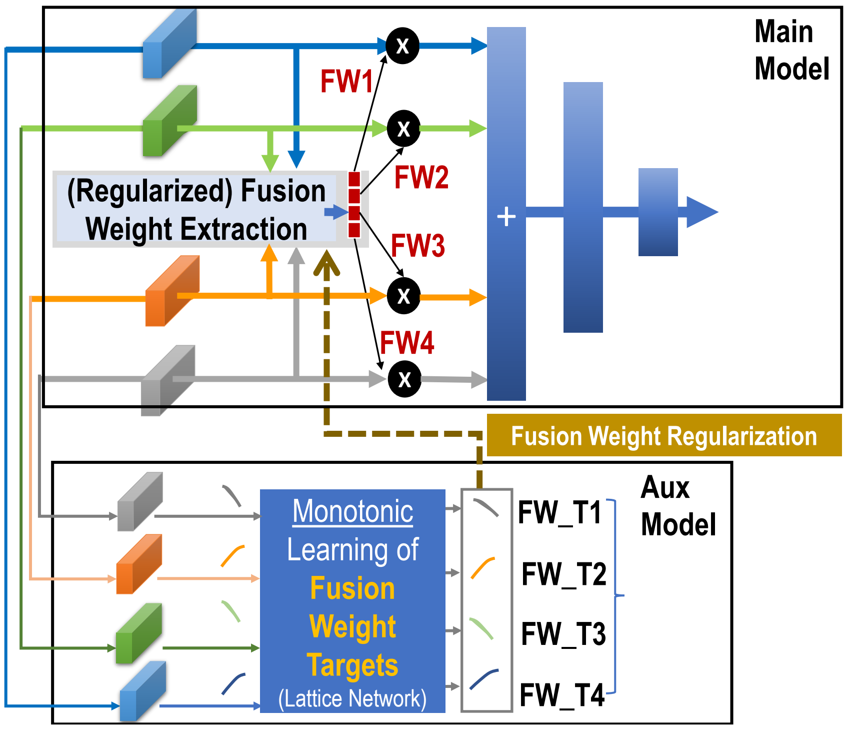

Main Model
FW1
FW2
FW3
FW4
(Regularized) Fusion Weight Extraction
+
Fusion Weight Regularization
Aux Model
Monotonic Learning of Fusion Weight Targets
(Lattice Network)
FW_T1
FW_T2
FW_T3
FW_T4

Figure 4.5: The ARGate architecture overview.

the fusion weights of other features $FW2$, $FW3$, and $FW4$ are relatively close to 0. As the multiplication mechanism of the fusion weights which switches off the unimportant feature path, the first feature makes a larger impact on the target prediction than the other features.

The auxiliary model is added here to facilitate the reliable training for the fusion weights, regularizing the fusion weights with auxiliary losses reflecting the relevance between the target value and each individual feature.

4.5.2 Bijective RevNet for Non-Linear Representation Learning

The ARGate identifies the important input features through fusion weights, which serves as a great tool for dimensionality reduction. However, if we directly apply AR-

Gate on the raw variation parameters, the reduced dimension is only a subset of original variation parameters, which completely ignores the correlation between different variational parameters. Therefore, we applied reversible residual network (RevNet) [57, 58, 59, 60, 61] to learn the correlation between multiple features, and forms a nonlinear internal representation for original features. In addition, one of major advantage of using RevNet is that it avoids information loss between its input and output, giving a bijective function. Fig. 4.6 gives a typical RevNet block with the feature mixing process given follows.

$$\begin{aligned} u_{n+1} &= u_n + hK_{n,1}^T\sigma(K_{n,1}v_n + b_{n,1}), \\ v_{n+1} &= v_n - hK_{n,2}^T\sigma(K_{n,2}u_{n+1} + b_{n,2}), \end{aligned} \tag{4.4}$$

where n ranges from $[0, N-1]$ for a RevNet with N RevNet blocks, u_n and v_n are two partitions for the nth state with same dimensionality, and h is a scaling factor.

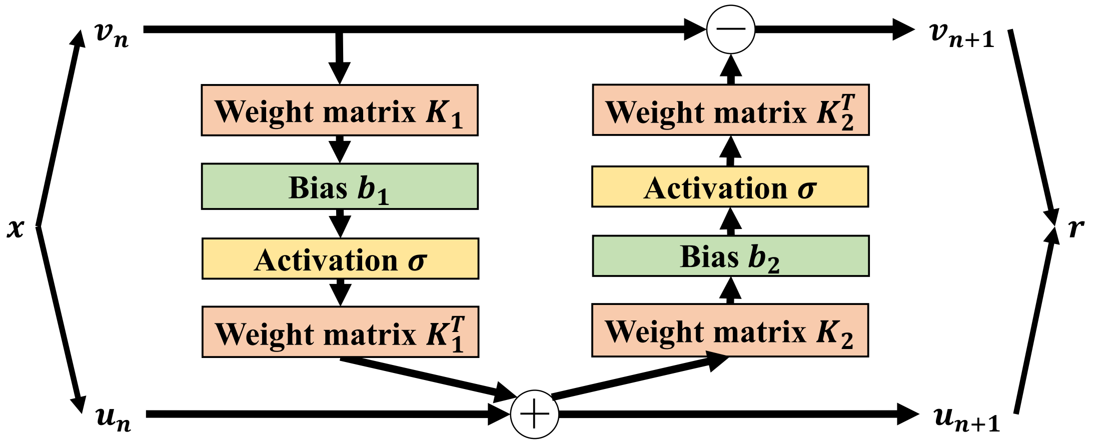


Figure 4.6: The reversible block.

Here we denote the RevNet nonlinear representation learning with $g : \mathbf{x} \mapsto \mathbf{r}$, which goes through N blocks of (4.4) as follows.

$$\mathbf{x} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \underset{g^{-1}}{\overset{g}{\underset{\leftarrow}{\rightarrow}}} \begin{bmatrix} u_N \\ v_N \end{bmatrix} = \mathbf{r}, \tag{4.5}$$

where $u_0 := (x_1, ..., x_{[D/2]})^{\mathrm{T}}$ and $v_0 := (x_{[D/2]+1}, ..., x_D)^{\mathrm{T}}$ are the two partitions of the input vector $\mathbf{x}$, and $u_N := (r_1, ..., r_{[D/2]})^{\mathrm{T}}$ and $v_N := (r_{[D/2]+1}, ..., r_D)^{\mathrm{T}}$ are the two partitions of the RevNet output $\mathbf{r}$.

4.5.3 Proposed Rev-Gate architecture

The proposed Rev-Gate architecture connects a RevNet and an ARGate in serial for efficient dimension reduction. In order to utilize such architecture for high-dimensional Bayesian optimization, we first pre-train the proposed architecture using a small amount of data to identify good dimension size d for dimension reduction, and then we utilize the trained RevNet in the reverse direction to embed the low dimension optimized $\mathbf{z}^*$ from acquisition function into $\mathbf{x}^*$ in the original high-dimension space.

Dimension Reduction via the Proposed Architecture

Thanks to the nonlinear bijective characteristic of the RevNet, we generate an internal representation $\mathbf{r} = g(\mathbf{x})$ mapped from the original variation parameter $\mathbf{x}$, which share the same dimensionality D as $\mathbf{x}$. With the feature importance interpretability from the ARGate, the importance of each internal representation dimension r_i can be estimated via the corresponding trained fusion weight FW_i. Given user-defined importance threshold FW_{TH}, the dimensionality d of low dimension space $\mathcal{Z}$ in BO can be determined by the number of fusion weights larger than FW_{TH}. The corresponding d internal representation elements can be reassembled into the low dimension feature $\mathbf{z} = (r_{i_1}, \cdots, r_{i_d})^{\mathrm{T}}$ used in BO, with each element r_{i_j} having $FW_{i_j} \geq FW_{TH}$, achieving the dimension reduction for surrogate model and acquisition function. The rest $D - d$ elements in $\mathbf{r}$ are considered as noncritical, and marked as $\mathbf{r}_n$ here.

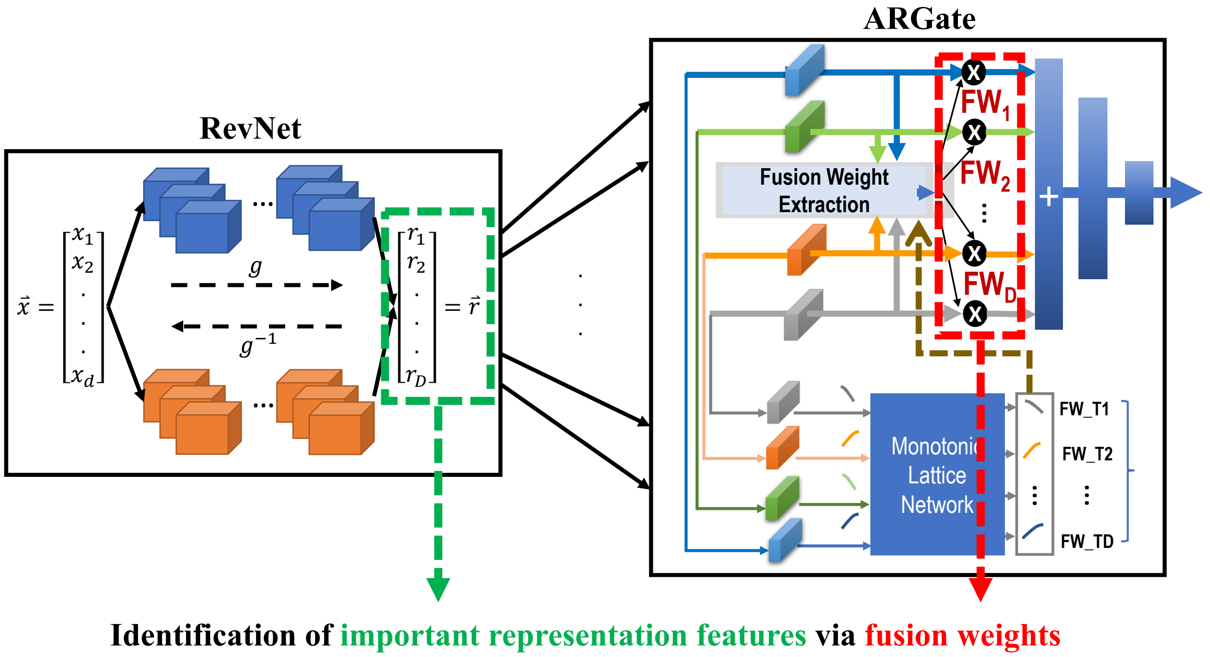

ARGate
RevNet
$\vec{x} = [x_1, x_2, \ldots, x_d]$
g
g^{-1}
$[r_1, r_2, \ldots, r_D] = \vec{r}$
Fusion Weight Extraction
FW_1
FW_2
FW_D
Monotonic Lattice Network
FW_T1
FW_T2
FW_TD
Identification of important representation features via fusion weights

Figure 4.7: The proposed Rev-Gate architecture. Important internal representation features are identified via fusion weight values.

Dimension Embedding using Reverse RevNet

As shown in Fig. 4.2, the BO framework provides a $\mathbf{z}^*$ with a dimensionality of d during each iteration, which needs to be embedded into the original space $\mathcal{X}$. Here we use the trained RevNet in a reverse direction to map low-dimensional $\mathbf{z}^*$ back to $\mathbf{x}^*$ in the high dimensional space. However, to use RevNet for the restoration of the original variational parameters $\mathbf{x}^*$, it requires the same dimensionality for the input and the output of RevNet. Therefore, $D - d$ new elements should be generated and combined with $\mathbf{z}^*$ to obtain the restored internal representation $\mathbf{r}^*$. More details about this conversion from $\mathbf{z}^*$ to $\mathbf{r}^*$ is discussed in Section 4.6. Given the reversibility of RevNet without information loss, we can easily recover the original variation parameters using $\mathbf{x}^* = g^{-1}(\mathbf{r}^*)$ to perform the required dimension embedding in BO as shown in Fig. 4.8.

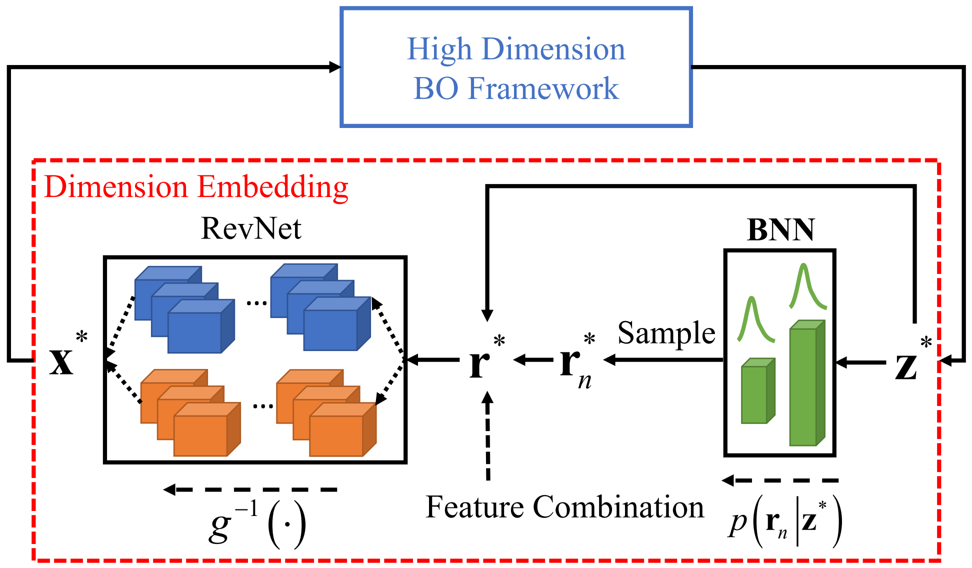


Figure 4.8: The proposed BO dimension embedding using RevNet and BNN.

4.6 Enhanced Dimension Embedding via Bayesian Neural Network

As mentioned in Section 4.5.3, additional elements should be appended to $\mathbf{z}^*$ to ensure the resulting $\mathbf{r}^*$ sharing the same dimensionality as $\mathbf{x}^*$ for traversing the RevNet in the reverse direction. As we know from the fusion weights, the additional elements are less critical for the final circuit performance. Hence, the simplest approach here is to append zeros to $\mathbf{z}^*$ to the high dimensionality D.

However, the zero appending approach neglects the correlation between $\mathbf{z}^*$ and $\mathbf{r}_n$ which both depend on the original input vector $\mathbf{x}$. Instead, we propose to learn a conditional probability distribution $p\left(\mathbf{r}_n \mid \mathbf{z}^*\right)$ to recover $\mathbf{r}^*$ from $\mathbf{z}^*$. Here, the particular probabilistic model we used for this conditional distribution is a Bayesian neural network (BNN).

After the Rev-Gate is trained, with the fixed RevNet, we can generate the internal representation $\mathbf{r}$, and seperate them into important features $\mathbf{z}$ and non-important ones

$\mathbf{r}_n$ for all the training data. Then the conditional distribution $p(\mathbf{r}_n \mid \mathbf{z}^*)$ represented by the BNN is estimated with maximum likelihood estimation using these pre-processed data. During dimension embedding in the BO process, a new $\mathbf{r}_n^*$ is randomly sampled from the learnt $p(\mathbf{r}_n \mid \mathbf{z}^*)$ using a trained BNN, and then combined with $\mathbf{z}^*$ to obtain recovered internal representation $\mathbf{r}^*$ for the RevNet conversion. The complete dimension embedding illustration is presented in Fig. 4.8.

4.7 Experimental Results for Rev-Gate BO

4.7.1 Experimental Setups

We demonstrated our proposed Rev-Gate architecture with BO approach with the two same circuits as before: a low-dropout (LDO) regulator [39] (60 dimensions) and a DC-DC converter [40] (44 dimensions).

For the rare failure detection performance comparison, we compared our proposed architectures with Monte Carlo (MC), expected improvement (EI), probability of improvement (PI) in [44], parallelizable Bayesian optimization (pBO) and parallelizable Bayesian optimization with random embedding (HDBO). The BayesOpt [48] was utilized for implementing BO methods. All the simulations were run on a workstation with a 3.50GHz Intel(R) Xeon(R) E5-1620 v4 CPU.

The proposed Rev-Gate is implemented with Pytorch 1.2 [62]. To be specific with the training process, the Rev-Gate is pre-trained with a small amount of the circuit simulation samples which are uniformly distributed in a pre-defined hyper-cube space. For a fair comparison with different BO based methods, we matched total simulation budget for all the BO methods including the number of training samples for the Rev-Gate. After the training phase, fusion weight values were examined for top-d indexes of important features extraction and screening out some non-important features. With the indexes of the important features, a simple BNN is trained for the non-essential component conditional distribution estimation. Finally, with the trained RevNet and the BNN, the BO framework is operated so that the z^* vector is computed with BNN and RevNet in reverse direction to generate x^*. The x^* is used as input sample for the circuit simulation and the simulation output y^* is passed onto the BO surrogate model.

Low-dropout Regulator

The number of essential features extracted from the proposed architecture is 26 out of 60 for quiescent current and load regulation, 30 for undershoot. The same dimension reduction is used for HDBO. Furthermore, in the light of circuit aspects, the Rev-Gate identify the actual important features from the inputs while HDBO cannot. We observe that most important parameters are located on the output stage in the LDO regulator, which is close to the circuit designers' insights.

DC-DC converter

Through our proposed Rev-Gate, we could reduce the number of dimension from 44 to 14 for output accuracy, and 16 for overshoot, which is far less than half of the total number of input features. Detailed simulation budget setup is included in Table 4.4.

4.7.2 Failure Detection Results

From Table 4.3 and 4.4, MC, EI, PI, pBO and HDBO methods cannot detect a failure case due to the challenging rare failure detection in the high-dimensional parameter space. On the other hand, our proposed Rev-Gate based BO framework successfully find the worst case for all specifications with in the simulation budget thanks to proposed Rev-

Table 4.3: Failure detection result comparison for the LDO regulator (60 dimension).

Spec	Target	Method	# Sim	Worst Case	1st Failure Hit	Runtime
Quiescent Current	11.0mA	MC	$330,000$	10.8mA	-	47h45m
		EI	$50_{init} + 750_{seq}$	7.1mA	-	24h06m
		PI	$50_{init} + 750_{seq}$	8.3mA	-	23h43m
		pBO	$50_{init} + 5 \times 150_{batch}$	10.5mA	-	23h53m
		HDBO	$50_{init} + 5 \times 150_{batch}$	10.1mA	-	19h36m
		Rev-Gate	$500_{training} + 50_{init} + 5 \times 50_{batch}$	**11.6mA**	**621**	**3h32m**
Undershoot	0.52V	MC	$330,000$	0.32V	-	47h45m
		EI	$50_{init} + 1250_{seq}$	0.27V	-	80h25m
		PI	$50_{init} + 1050_{seq}$	0.23V	-	65h48m
		pBO	$50_{init} + 5 \times 140_{batch}$	0.49V	-	19h52m
		HDBO	$50_{init} + 5 \times 250_{batch}$	0.51V	-	32h25m
		Rev-Gate	$1000_{training} + 50_{init} + 5 \times 50_{batch}$	**0.53V**	**1248**	**3h29m**
Load regulation	58.2%	MC	$330,000$	36.0%	-	47h45m
		EI	$50_{init} + 1250_{seq}$	28.9%	-	104h39m
		PI	$50_{init} + 1050_{seq}$	13.9%	-	65h04m
		pBO	$50_{init} + 5 \times 140_{batch}$	58.1%	-	19h47m
		HDBO	$50_{init} + 5 \times 250_{batch}$	58.1%	-	32h19m
		Rev-Gate	$1000_{training} + 50_{init} + 5 \times 50_{batch}$	**58.4%**	**1241**	**3h41m**

Table 4.4: Failure detection result comparison for the DC-DC converter (44 dimension).

Spec	Target	Method	# Sim	Worst Case	1st Failure Hit	Runtime
Output accuracy	58mV	MC	40,800	42.4mV	-	47h50m
		EI	$50_{init} + 1250_{seq}$	25.8mV	-	92h44m
		PI	$50_{init} + 1250_{seq}$	22.4mV	-	92h13m
		pBO	$50_{init} + 5 \times 250_{batch}$	57.3mV	-	97h08m
		HDBO	$50_{init} + 5 \times 250_{batch}$	57.7mV	-	21h21m
		Rev-Gate	$1000_{training} + 50_{init} + 5 \times 50_{batch}$	**58.1mV**	**1177**	**4h53m**
Overshoot	8.8mV	MC	40,800	8.29mV	-	47h50m
		EI	$50_{init} + 1250_{seq}$	7.30mV	-	92h31m
		PI	$50_{init} + 1250_{seq}$	7.37mV	-	92h34m
		pBO	$50_{init} + 5 \times 250_{batch}$	8.65mV	-	97h46m
		HDBO	$50_{init} + 5 \times 250_{batch}$	8.77mV	-	25h21m
		Rev-Gate	$1000_{training} + 50_{init} + 5 \times 50_{batch}$	**8.84mV**	**1221**	**3h59m**

Gate architecture for dimension embedding in BO. In terms of the magnitude of the worst case detected, MC, EI, PI typically cannot find any worse case near the target for most specification under consideration. pBO and HDBO presents a better performance with its good exploration and exploitation balancing, while the proposed architecture presents the worst case detected than all other methods for all the specifications with the help of effective dimension embedding given by the Rev-Gate architecture.

Regarding simulation running time, overall BO based methods like EI, PI and pBO take much longer than our proposed Rev-Gate with BNN due to high overhead introduced by surrogate model and acquisition function in high dimension. HDBO suffers from its simple dimension embedding mechanism to achieve poor failure detection efficiency. With a smart sampling budget allocation for Rev-Gate pre-trained dimension reduction, the BO search efficiency is significantly improved leading to short runtime. Note that the runtime for Rev-Gate with BNN includes pre-training phase.

4.7.3 Worst Case Trend Analysis

Finally, the worst case trend is analyzed as shown in Fig.4.9. BO based methods such as EI and PI found the worst case slowly comparing to pBO and HDBO. During the first 500 samples, EI, PI, and pBO shows similar failure detection performance but the worst case of pBO rises after the 500 samples. The worst case of HDBO was bit larger than other BO based methods but it is stuck at local minima around 600 samples. Our proposed Rev-Gate shown in green color in the graph starts with the lowest worst case, it found its worst case much more rapidly than the other methods at the initial search process. Note that 500 samples are used for pre-training Rev-Gate architecture. From this result, it is clear that pre-training process shows significant benefits for the improved failure detection performance and efficiency.

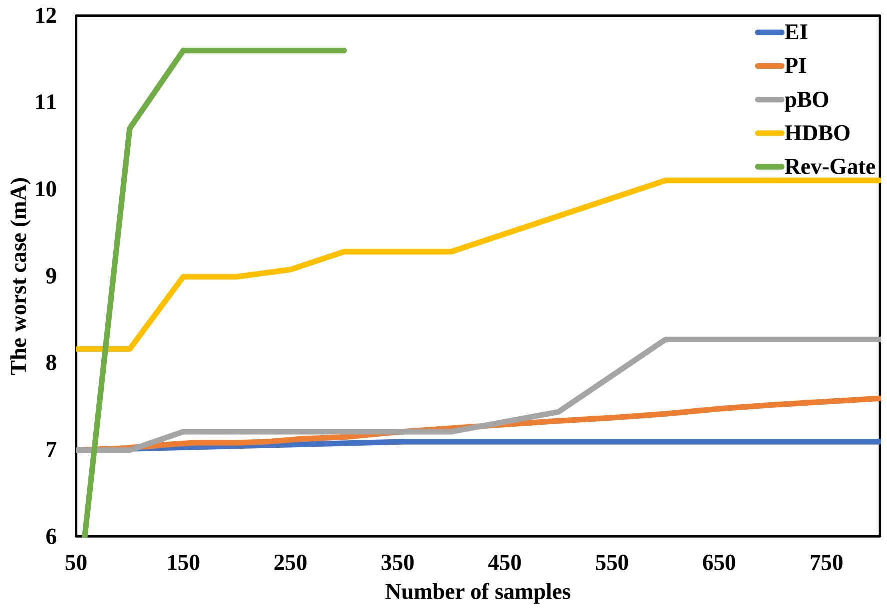


Figure 4.9: A plot of the worst case found in quiescent current in the LDO regulator. Note that Rev-Gate with BNN only runs 300 samples on the BO framework for fair comparison.

4.8 Summary

In this chapter, we present two different high-dimensional Bayesian optimization procedures for rare failure detection of analog/mixed-signal circuits. First, we utilized random embedding techniques to remove redundant features and reduce the dimensionality of Bayesian optimization search space, resulting highly-efficient failure detection. Furthermore, we present the Rev-Gate architecture with a novel restoration scheme via Bayesian neural network. The proposed algorithm works under Bayesian optimization for rare failure detection of analog mixed-signal circuits. The ARGate is adopted for the identification of important features and the RevNet is utilized for input restoration via backward computation without loss of information. The Bayesian neural network is applied for non-essential parameter estimation under the nature of conditional probability distribution given essential variational inputs. The experimental results show that both of our proposed algorithms detect rare failure cases in high dimensional space with less amount of time, while Bayesian optimization with traditional and improved acquisition function does not find anomaly during the circuit simulation.

Chapter 5

Global Adversarial Attacks for Assessing Deep Learning Robustness

In this chapter, we propose a new concept of global adversarial examples and several global attack methods to verify the robustness of neural networks. Specifically, we (**a**) propose a novel concept called global adversarial example pairs and formulate a global adversarial attack problem for assessing the model robustness over the entire input space without extra data labeling; (**b**) present two families of global adversarial attack methods: (**1**) alternating gradient adversarial attacks and (**2**) extreme-value-guided MCMC sampling attack, and demonstrate their effectiveness in generating global adversarial example pairs; (**c**) using the proposed global attack methods, demonstrate that DNNs hardened using strong projected gradient descent (PGD) based (local) adversarial training are vulnerable towards the proposed global adversarial example pairs, suggesting that global robustness must be considered while training DNNs.

©2019 arXiv. Reprinted, with permission, from Hanbin Hu, Mit Shah, Jianhua Z. Huang, and Peng Li, "Global Adversarial Attacks for Assessing Deep Learning Robustness", arXiv:1906.07920, 2019.

5.1 Local Adversarial Attacks

Deep neural networks (DNNs) have been applied to many applications including safety-critical tasks such as autonomous driving [63] and unmanned autonomous vehicles (UAVs) [64], which demand high robustness of decision making. However, recently it has been shown that DNNs are susceptible to attacks by adversarial examples [65]. For image classification, for example, adversarial examples may be generated by adding crafted perturbations indistinguishable to human eyes to legitimate inputs to alter the decision of a trained DNN into an incorrect one. Several studies attempt to reason about the underlying causes of susceptibility of deep neural networks towards adversarial examples, for instance, ascribing vulnerability to linearity of the model [65] or flatness/curvedness of the decision boundaries [66]. A widely agreed consensus is certainly desirable, which is under ongoing research.

5.1.1 Target global adversarial attack problem.

The main objectives of this work are to reveal potential vulnerability of DNNs by presenting a new type of attacks, namely *global adversarial attacks*, propose methods for generating such global attacks, and finally demonstrate that DNNs enhanced by conventional (local) adversarial training exhibit little defense to the proposed global adversarial examples. While several adversarial attack methods were proposed [65, 67, 68, 69, 70] in recent literature, we refer to these methods as *local adversarial attack methods* as they all aim to solve the *local adversarial attack problem* defined as follows.

Definition 2 ***Local adversarial attack problem***. *Given an input space Ω, one legitimate input example $\mathbf{x} \in \Omega$ with label $y \in \mathcal{Y}$, and a trained DNN $f : \Omega \to \mathcal{Y}$, find another (adversarial) input example $\mathbf{x}' \in \Omega$ within a radius of ϵ around $\mathbf{x}$ under a distance measure defined by a norm function $\|\cdot\| : \{\mathbf{x}_a - \mathbf{x}_b \,|\mathbf{x}_a \in \Omega, \mathbf{x}_b \in \Omega\} \to \mathbb{R}_{\geq 0}$ such* *that* $f(\mathbf{x}') \neq y, \|\mathbf{x}' - \mathbf{x}\| \leq \epsilon$.

Typically, the above problem is solved via optimization governed by a loss function, $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, measuring the difference between the predicted label $f(\mathbf{x}')$ and y:

$$\mathbf{x}'^{*} = \underset{\|\mathbf{x}' - \mathbf{x}\| \leq \epsilon, \mathbf{x}' \in \Omega}{\arg\max} \mathcal{L}\left(f\left(\mathbf{x}'\right), y\right). \tag{5.1}$$

Importantly, the above problem formulation has two key limitations. **(1)** it is deemed *local* in the sense that it only examines model robustness inside a *local* region centered at each given input $\mathbf{x}$, which in practice is chosen from the training or testing dataset. As such, local adversarial attacks are not adequate since evaluating the DNN robustness around the training and testing data does not provide a complete picture of robustness globally, i.e. in the entire space Ω. On the other hand, assessment of global robustness is essential, e.g. for safety-critical applications. **(2)** local attack methods assume that for each clean example $\mathbf{x}$ the label y is known. As a result, they are inapplicable to attack the DNN around locations where no labeled data are available.

In this paper, we propose a notion of global DNN robustness and a global adversarial attack problem formulation to assess it. We evaluate global robustness of a given DNN by assessing the potential high sensitivity of its decision function with respect to small input perturbation leading to change in the predicted label, globally in the entire Ω. By solving the global attack problem, we generate multiple *global adversarial example pairs* such that each pair of two examples are close to each other but have different labels predicted by the DNN. The detailed definitions are presented in Section 5.2.

5.1.2 Related works.

Apart from the local adversarial attacks, several other approaches for DNN robustness evaluation have been reported. The Lipschitz constant is utilized to bound DNNs' vulnerability to adversarial attacks [71, 72]. As argued in [73, 74], however, currently there is no accurate method for estimating the Lipschitz constant, and the resulting overestimation can easily render its use unpractical. [75, 76] propose to train a generative model for generating unseen samples for which misclassification happens. However, the ground-truth labels of generated examples must be provided for final assessment and these examples do not capture model's vulnerability due to high sensitivity to small input perturbation.

5.2 Global adversarial attacks

We formulate a new global adversarial attack problem as follows.

Definition 3 ***Global adversarial attack problem***. *Given an input space Ω and an DNN model $f : \Omega \to \mathcal{Y}$, find one or more global adversarial example pairs $(\mathbf{x}_1, \mathbf{x}_2) \in \Omega \times \Omega$ within a radius of ϵ under a distance measure defined by a norm function $\|\cdot\| : \{\mathbf{x}_a - \mathbf{x}_b \,|\mathbf{x}_a \in \Omega, \mathbf{x}_b \in \Omega\} \to \mathbb{R}_{\geq 0}$ such that $f(\mathbf{x}_1) \neq f(\mathbf{x}_2), \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \epsilon$.*

When no confusion occurs, *global adversarial example pair* and *global adversarial examples* are used interchangeably throughout this paper. The above problem formulation can be cast into an optimization problem w.r.t. a certain loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ in the following form:

$$\mathbf{x}_1^*, \mathbf{x}_2^* = \underset{\|\mathbf{x}_1 - \mathbf{x}_2\| \leq \epsilon, (\mathbf{x}_1, \mathbf{x}_2) \in \Omega \times \Omega}{\arg\max} \mathcal{L}\left(f(\mathbf{x}_1), f(\mathbf{x}_2)\right) \tag{5.2}$$

For convenience of notation, we use $\mathcal{L}_f(\mathbf{x}_1, \mathbf{x}_2)$ to denote $\mathcal{L}(f(\mathbf{x}_1), f(\mathbf{x}_2))$. The above definition and problem formulation have several favorable characteristics. A robust DNN model should be insensitive to small input perturbations. Therefore, two nearby inputs shall share the same (or similar) model output. Conversely, any large $\mathcal{L}_f(\mathbf{x}_1, \mathbf{x}_2)$ value of two nearby inputs reveals a sharp transition over the decision boundary in a classification task or an unstable region in a regression task.

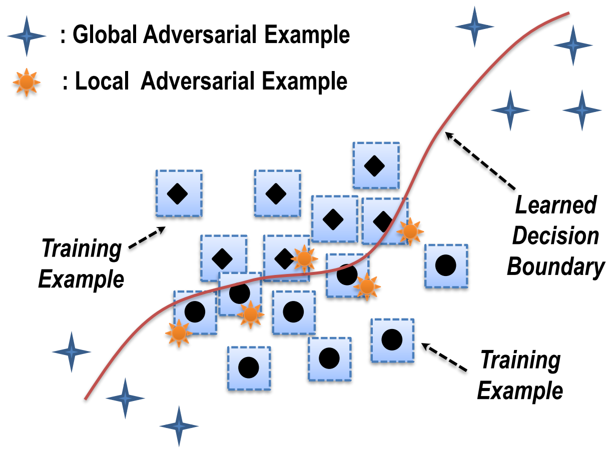

: Global Adversarial Example
: Local Adversarial Example
Learned Decision Boundary
Training Example
Training Example

Figure 5.1: Global vs. local adversarial examples.

Loosely speaking, the maximum of $\mathcal{L}_f(\mathbf{x}_1, \mathbf{x}_2)$ over the entire input space can serve as a measure of global model robustness. If it is larger than a preset threshold, the DNN may be deemed as vulnerable towards global adversarial attacks. On the other hand, (5.2) is a global optimization problem for which multiple near-optimal solutions may be reached by starting from different initial solutions or employing different optimization methods. Practically, any pair of two close inputs $\mathbf{x}_1, \mathbf{x}_2$ with different model predictions in the case of classification or a sufficiently large value of $\mathcal{L}_f(\mathbf{x}_1, \mathbf{x}_2)$ in the case of regression may be considered as a global adversarial example pair.

It is important to note that our problem formulation doesn't restrict adversarial examples to be around a certain input example as in the case of the existing local adversarial attacks; it only sets an indistinguishable distance between a pair of two input examples in order to examine the entire input space, e.g. at locations far away from the training or testing dataset. Fig. 5.1 contrasts the conventional local adversarial examples with the proposed global adversarial examples. Importantly, our global attack formulation does not require additional labeled data; it directly measures the model's sensitivity to input perturbation and be applied globally in the entirety of the input space.

We propose two families of attack methods to solve (5.2) as a way of generating global adversarial examples: **1**) alternating gradient global adversarial attacks and **2**) extreme-value-guided MCMC sampling global adversarial attack, as discussed in Section 5.3 and Section 5.4, respectively.

5.3 Alternating gradient global adversarial attacks

$$\begin{tikzcd}
\mathbf{x}_1^{(0)} \arrow[d] & \mathbf{x}_2^{(0)} \arrow[d] \arrow[dl] \\
\mathbf{x}_1^{(1)} \arrow[d] \arrow[r] & \mathbf{x}_2^{(1)} \arrow[d] \arrow[dl] \\
\mathbf{x}_1^{(2)} \arrow[r] & \mathbf{x}_2^{(2)} \\
\vdots \arrow[d] & \vdots \arrow[d] \arrow[dl] \\
\mathbf{x}_1^{(n)} \arrow[r] & \mathbf{x}_2^{(n)}
\end{tikzcd}$$

Figure 5.2: Alternating attack illustration.

For global adversarial example pair generation per (5.2), a pair of two examples $(\mathbf{x}_1, \mathbf{x}_2)$ shall be be optimized to maximize the loss $\mathcal{L}_f(\mathbf{x}_1, \mathbf{x}_2)$ under a distance constraint. We propose a family of attack methods called alternating gradient global adversarial attacks which proceed as follows: 1) start from an initial input pair; 2) fix the first example, and then attack (move) the second under the distance constraint while maximizing the loss $\mathcal{L}_f(\mathbf{x}_1, \mathbf{x}_2)$ using a gradient-based local adversarial attack method, referred to as a *sub-attack method* here, 3) swap the roles of the first and (updated) second examples, i.e. fix the second example while attacking the first, 4) repeat this process for a number of iterations, as shown in Fig. 5.2.

Given a DNN model f and a loss function $\mathcal{L}$, a sub-attack method can be characterized using a function $\mathbf{x}' = f_{s\_attk}\left(\mathbf{x}, y, \mathcal{R}_{\mathbf{x},\epsilon}; f, \mathcal{L}\right)$ constructing an adversarial example $\mathbf{x}'$ w.r.t example $\mathbf{x}$ and its corresponding label y, where $\mathcal{R}_{\mathbf{x},\epsilon} = \{\mathbf{x}+\delta|\, \|\delta\| \leq \epsilon\}$ specifies the region for adversarial sample generation. Suppose we start with an initial example pair $\left(\mathbf{x}_1^{(0)}, \mathbf{x}_2^{(0)}\right)$, then we get the $(i+1)$-round example pair from the (i)-round sample pair by:

$$\begin{aligned}
\mathbf{x}_1^{(i+1)} &= f_{s\_attk}\left(\mathbf{x}_1^{(i)}, f\left(\mathbf{x}_2^{(i)}\right), \mathcal{R}_{\mathbf{x}_2^{(i)},\epsilon}; f, \mathcal{L}\right) && (5.3) \\
\mathbf{x}_2^{(i+1)} &= f_{s\_attk}\left(\mathbf{x}_2^{(i)}, f\left(\mathbf{x}_1^{(i+1)}\right), \mathcal{R}_{\mathbf{x}_1^{(i+1)},\epsilon}; f, \mathcal{L}\right) && (5.4)
\end{aligned}$$

Here, to attack one example, we use the other example's model prediction as the label when applying the sub-attack method f_{s_attk} so that the difference between two examples' model predictions is maximized. In the meanwhile, the search region for attacking one example is constrained (centered) by the other example. Hence, the distance between the two examples is always less than ϵ. Equations 5.3 and 5.4 are invoked to alternately attack $\mathbf{x}_1^{(i)}$ and $\mathbf{x}_2^{(i)}$ to generate an updated example pair for the next round. As the global attack continues, multiple global adversarial sample pairs may be generated along the way.

5.3.1 Choice of sub-attack methods.

We leverage the popular gradient-based local adversarial attack methods as sub-attack methods under the above family of global attacks. Particularly, the fast gradient sign method (FGSM) [65], iterative FGSM (IFGSM) [69], and projected gradient descent (PGD) [70] may be considered for sub-attack method f_{s_attk}. As one example, we present the algorithm flow for the global PGD (G-PGD) attack targeting classifiers with PGD employed as the sub-attack in Algorithm 5, where the $Clip\left(\mathbf{x}, \mathcal{R}\right)$ function drags the input

Algorithm 5: Global PGD attack algorithm for classification.

```
Require:
    Initial starting example pair $\left(\mathbf{x}_1^{(0)}, \mathbf{x}_2^{(0)}\right)$; Example vector dimension $D$; distance constraint $\epsilon$; Number of total rounds $N$; Number of sub-attack steps $S$; Step size $a$ for the sub-attack.
Ensure: The set of generated global adversarial sample pairs $T$;
 1: $T = \{\}$
 2: for $i \leftarrow 1$ to $N$ do
 3:     Sample a uniform distribution perturbation $\delta \sim \mathrm{U}\left[-\epsilon, \epsilon\right]^D$
 4:     $\mathbf{x}_1^{(i)} \leftarrow Clip\left(\mathbf{x}_1^{(i-1)} + \delta, \mathcal{R}_{\mathbf{x}_2^{(i-1)},\epsilon}\right)$
 5:     for $j \leftarrow 1$ to $S$ do
 6:         $\mathbf{x}_1^{(i)} \leftarrow Clip\left(\mathbf{x}_1^{(i)} + a \cdot \mathrm{sign}\left(\nabla_{\mathbf{x}_1}\mathcal{L}_f\left(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i-1)}\right)\right), \mathcal{R}_{\mathbf{x}_2^{(i-1)},\epsilon}\right)$
 7:     end for
 8:     Sample a uniform distribution perturbation $\delta \sim \mathrm{U}\left[-\epsilon, \epsilon\right]^D$
 9:     $\mathbf{x}_2^{(i)} \leftarrow Clip\left(\mathbf{x}_2^{(i-1)} + \delta, \mathcal{R}_{\mathbf{x}_1^{(i)},\epsilon}\right)$
10:     for $j \leftarrow 1$ to $S$ do
11:         $\mathbf{x}_2^{(i)} \leftarrow Clip\left(\mathbf{x}_2^{(i)} + a \cdot \mathrm{sign}\left(\nabla_{\mathbf{x}_2}\mathcal{L}_f\left(\mathbf{x}_2^{(i)}, \mathbf{x}_1^{(i)}\right)\right), \mathcal{R}_{\mathbf{x}_1^{(i)},\epsilon}\right)$
12:     end for
13:     $T \leftarrow T \cup \left\{\left(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}\right)\right\}$
14: end for
```

$\mathbf{x}$ outside the preset region $\mathcal{R}$ onto the boundary of $\mathcal{R}$. The global IFGSM (G-IFGSM) skips the random noise perturbations (steps 3 and 8). The global FGSM (G-FGSM) sets the number of sub-attack steps S to be 1 and the step size for sub-attack a to be ϵ in addition to ignoring the random noise perturbation steps.

5.4 Extreme-value-guided MCMC sampling attack

While the family of alternating gradient global adversarial attacks discussed in Section 5.3 can work effectively in practice, such methods may get trapped at a local maximum, degrading the quality of global attack. To this end, we propose a stochastic optimization approach based on the extreme value distribution theory and Markov Chain Monte Carlo (MCMC) method, which is more advantageous from a global optimization point of view.

5.4.1 Extreme value distribution.

Consider sampling a set of i.i.d input example pairs $\{(\mathbf{x}_{1,1}, \mathbf{x}_{2,1}), \cdots, (\mathbf{x}_{1,n}, \mathbf{x}_{2,n})\}$ in the input pair space $\Omega \times \Omega$. The greatest loss value $L^* = \max_{i \in [1,n]} \mathcal{L}_f(\mathbf{x}_{1,i}, \mathbf{x}_{2,i})$ can be regarded as a random variable following a certain distribution characterized by its density function $p_{L^*}(l)$. The Fisher-Tippett-Gnedenko theorem says that the distribution of the maximum value of examples, if exists, can only be one of the three families of extreme value distribution: the Gumbel class, the Fréchet and the reverse Weibull class [77]. Hence, $p_{L^*}(l)$ falls into one of the three families as well, whose cumulative density function (CDF) $F_{L^*}(l)$ can be written in a unified form, called the generalized extreme value (GEV) distribution:

$$F_{L^*}(l) = \begin{cases} \exp\left(-\left(1+\xi\frac{l-\mu}{\sigma}\right)^{-1/\xi}\right) & \xi \neq 0 \\ \exp\left(-\exp\left(-\frac{l-\mu}{\sigma}\right)\right) & \xi = 0 \end{cases} \tag{5.5}$$

where μ, σ and ξ are the location, scale, and shape parameter for the GEV distribution and may be obtained through the maximum likelihood estimation (MLE).

Assuming that the desired generalized extreme value (GEV) distribution of the loss function is available, multiple large loss values, corresponding to potential global adversarial example pairs, may be generated by sampling the GEV distribution. The added benefit here is that the inherent randomness in this sampling process may be explored to find more globally optimal solutions. Nevertheless, since the GEV distribution may not be easily sampled directly, We adopt the Markov Chain Monte Carlo (MCMC) method to sample the GEV distribution [78]. In reality, the GEV distribution is not known *a priori* and shall be estimated using MLE based on a sample of data as described below.

5.4.2 Extreme-value-guided MCMC sampling algorithm (GEVMCMC).

The proposed GEVMCMC algorithm is shown in Algorithm 6 for the case of classification problems. There exist two essential components for MCMC sampling: the target distribution, which is in this case the desired GEV distribution, and the proposal distribution, which is a surrogate distribution easy to sample. For each MCMC round, we collect an example from the proposal distribution, and then accept this example or discard it while keeping the previous one based on an acceptance ratio p_A in Step 6 of Algorithm 6 [78]. In order to sample the block maximum for the extreme value distribution, a block of example pairs are collected from the proposal distribution in each round as in Step 3 of the algorithm instead of a single example. As the MCMC sampling process proceeds, the actual sampling distribution implemented converges to the target (GEV) distribution.

Algorithm 6: Extreme-value-guided MCMC sampling algorithm (GEVMCMC) for global attack.

```
Require: Initial starting example pair $\left(\mathbf{x}_1^{(0)}, \mathbf{x}_2^{(0)}\right)$; Number of total rounds $N$;
    Number of warm-up rounds $N_w$; Block size $B$; Number of example pairs $k$ used for
    GEV distribution update;
Ensure: The set of the generated global adversarial example pairs $T$;
 1: Apply G-PGD for $N_w$ rounds to warm-up and update T.
 2: for $i \leftarrow N_w + 1$ to $N$ do
 3:     Sample $B$ i.i.d. samples $\left\{\left(\mathbf{x}_1^{[1]}, \mathbf{x}_2^{[1]}\right), \cdots, \left(\mathbf{x}_1^{[B]}, \mathbf{x}_2^{[B]}\right)\right\}$ from the proposal
        distribution $q\left(\mathbf{x}_1, \mathbf{x}_2 | \mathbf{x}_1^{(i-1)}, \mathbf{x}_2^{(i-1)}\right)$
 4:     $(\mathbf{x}_1^*, \mathbf{x}_2^*) \leftarrow \arg\max_{j \in [1,B]} \mathcal{L}_f\left(\mathbf{x}_1^{[j]}, \mathbf{x}_2^{[j]}\right)$
 5:     Update the GEV distribution $p_{L^*}(l)$ using top $k$ loss values in the history.
 6:     $p_A \leftarrow \min\left\{1.0, \frac{p_{L^*}\left(\mathcal{L}_f\left(\mathbf{x}_1^*, \mathbf{x}_2^*\right)\right) q\left(\mathbf{x}_1^{(i-1)}, \mathbf{x}_2^{(i-1)} | \mathbf{x}_1^*, \mathbf{x}_2^*\right)}{p_{L^*}\left(\mathcal{L}_f\left(\mathbf{x}_1^{(i-1)}, \mathbf{x}_2^{(i-1)}\right)\right) q\left(\mathbf{x}_1^*, \mathbf{x}_2^* | \mathbf{x}_1^{(i-1)}, \mathbf{x}_2^{(i-1)}\right)}\right\}$
 7:     Sample a uniform random variable $\alpha \sim \mathrm{U}[0,1]$
 8:     if $\alpha \leq p_A$ then
 9:         Accept the new example. $\left(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}\right) = (\mathbf{x}_1^*, \mathbf{x}_2^*)$
10:     else
11:         Reject and keep the previous example. $\left(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}\right) = \left(\mathbf{x}_1^{(i-1)}, \mathbf{x}_2^{(i-1)}\right)$
12:     end if
13:     $T \leftarrow T \cup \left\{\left(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}\right)\right\}$
14: end for
```

Importantly, the generalized extreme value (GEV) distribution $p_{L^*}(l)$ of the loss function doesn't exist at the beginning of the algorithm. Therefore, it is estimated during the sampling process. Two techniques are considered to obtain an accurate GEV distribution. A warm-up procedure (Step 1 in Algorithm 6) using a few rounds of G-PGD is performed first to collect a few global adversarial example pairs of large loss values. In each round, the top k loss values among all example pairs in the history including the ones in the current block are selected to estimate $p_{L^*}(l)$ based on MLE in Step 5 of Algorithm 6.

5.4.3 Proposal distribution design.

The convergence speed of MCMC sampling towards the target distribution critically depends on the proposal distribution [78]. To efficiently generate high-quality global adversarial example pairs, we consider the following essential aspects in designing the proposal distribution: (**1**) finding large loss values; (**2**) enabling global search; (**3**) constraining two examples in each pair to be within distance ϵ.

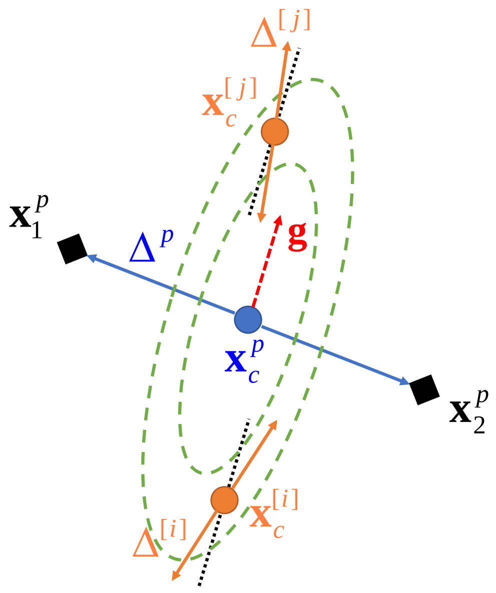


Figure 5.3: MCMC proposal distribution design illustration.

We decompose the sampling of an example pair into two sequential steps: sample the center location $\mathbf{x}_c$ and sample the difference vector Δ between the two examples. Then, we construct the example pair by $\mathbf{x}_1 = \mathbf{x}_c + \Delta, \mathbf{x}_2 = \mathbf{x}_c - \Delta$, as shown in Fig. 5.3. The two sampling steps are independent of each other, and hence, the proposal distribution conditioning on the previous example pair $(\mathbf{x}_c^p, \Delta^p)$ is split into a product of a center proposal and difference proposal distribution:

$$q\left(\mathbf{x}_c, \Delta | \mathbf{x}_c^p, \Delta^p\right) = q_{x^c}\left(\mathbf{x}_c | \mathbf{x}_c^p, \Delta^p\right) q_{\Delta}\left(\Delta | \mathbf{x}_c^p, \Delta^p\right). \tag{5.6}$$

As such, the distance constraint for the two examples is only taken into consideration in the design of $q_{\Delta}\left(\Delta | \mathbf{x}_c^p, \Delta^p\right)$ and no distance constraint is necessary when sampling the center.

We speed up the convergence of MCMC by incorporating the (normalized) gradient information

$$\mathbf{g} = \frac{\nabla_{\mathbf{x}_c} \mathcal{L}_f\left(\mathbf{x}_c^p + \Delta^p, \mathbf{x}_c^p - \Delta^p\right)}{\left|\nabla_{\mathbf{x}_c} \mathcal{L}_f\left(\mathbf{x}_c^p + \Delta^p, \mathbf{x}_c^p - \Delta^p\right)\right|} \tag{5.7}$$

into the proposal distribution design. We design the center proposal distribution to be a multi-variate Gaussian distribution centered at $\mathbf{x}_c^p$ with a covariance matrix biasing to sampling along the gradient direction $\mathbf{g}$ to increase the likelihood of finding large loss values while allowing sampling in other directions during the same time:

$$q_{x^c}\left(\mathbf{x}_c | \mathbf{x}_c^p, \Delta^p\right) = \mathcal{N}\left(\mathbf{x}_c^p, \lambda_0^2 \mathbf{I} + \left(\lambda_m^2 - \lambda_0^2\right) \mathbf{g}\mathbf{g}^{\mathrm{T}}\right), \tag{5.8}$$

where λ_m^2 sets the largest eigenvalue and $\lambda_0^2 < \lambda_m^2$ sets all other eigenvalues of the covariance matrix. The absolute values of λ_0 and λ_m control the size of the search region while the ratio between λ_m and λ_0 determines to what extent we want to focus on the gradient direction.

The gradient sign information is incorporated into the difference proposal distribution considering the distance constraint ϵ. Particularly, for the l_∞ norm based distance measure, we propose a Bernoulli distribution with parameter $p_B > 0.5$ for each element of the difference vector:

$$\Delta_i = \begin{cases} \frac{\epsilon}{2}\mathrm{sign}\left(g_i\right) & \text{with probability } p_B \\ -\frac{\epsilon}{2}\mathrm{sign}\left(g_i\right) & \text{with probability } 1 - p_B, \end{cases} \tag{5.9}$$

which ensures that the pair $(\mathbf{x}_1, \mathbf{x}_2)$ be within distance ϵ and each difference component is more likely to be set according to the corresponding gradient sign component.

5.5 Experimental results

5.5.1 Experimental settings.

We investigate several local and global methods on two popular image classification datasets: MNIST [79] and CIFAR10 [80]. To evaluate DNN robustness globally in the input space, we create an additional class "meaningless" and append 6,000 and 5,000 random noisy images (one tenth of the original training dataset) under this "meaningless" class into the original MNIST and CIFAR10 training datasets, respectively, and refer to the expanded training datasets as the augmented training datasets. All trained DNNs perform classification across 11 classes.

For MNIST, we train a neural network with two convolutional and two fully-connected layers with an accuracy of 99.43% with 40 training epochs. For CIFAR10, a VGG16 [81] network is trained for 300 epochs, reaching 94.25% accuracy. Furthermore, we globally attack adversarially-trained models, which are trained using adversarial training based on local adversarial examples. In each epoch of adversarial training, the adversarial samples are generated by attacking the DNN model from the last epoch using a 30-step local white-box PGD attack, which is considered a strong first-order attack [70]. And then an updated model is trained using both the augmented training set and the generated adversarial images. Adversarial training process is performed for additional 40 epochs for the MNIST model and 30 epochs for the CIFAR10 model, respectively. The ratio of the weighting parameters between the losses of the examples in the augmented training set and adversarial examples are 1 : 1.

5.5.2 Adversarial attack parameter settings.

The l_∞ norm based perturbation limit (the maximum allowed difference between two close images) is set to $\epsilon_{MNIST} = 0.1$ for MNIST and to $\epsilon_{CIFAR10} = 0.005$ for CIFAR10. We add another 10,000 random images with the "meaningless" class label into the original testing dataset (10,000 samples) to create an augmented testing dataset. We experiment three common local adversarial attack methods: FGSM [65], IFGSM [69], and PGD [70], referred to as L-FGSM, L-IFGSM and L-PGD in this paper. Both L-IFGSM and L-PGD perform a 30-step attack with a l_∞ step size of $\epsilon/10$. All local attacks are performed on the augmented testing set.

All four proposed global adversarial attack methods are considered: alternating gradient global adversarial attack with different sub-attack methods of FGSM (G-FGSM), IFGSM(G-IFGSM) and PGD (G-PGD); extreme-value-guided sampling global attack (GEVMCMC). For all global attack methods, we randomly pick 100 images from the original testing dataset and from the appended random testing dataset, respectively, to form the first images of the 100 starting pairs. The second image of each pair is obtained by adding small uniformly-distributed random noise bounded by perturbation size ϵ to the first image. 100 rounds of optimization are performed by all global adversarial attack methods, generating a two sets of 10,000 adversarial example pairs, one set for each of the two starting conditions: start with the 100 original testing images, start from the 100 appended random testing images. G-IFGSM and G-PGD share the same parameter

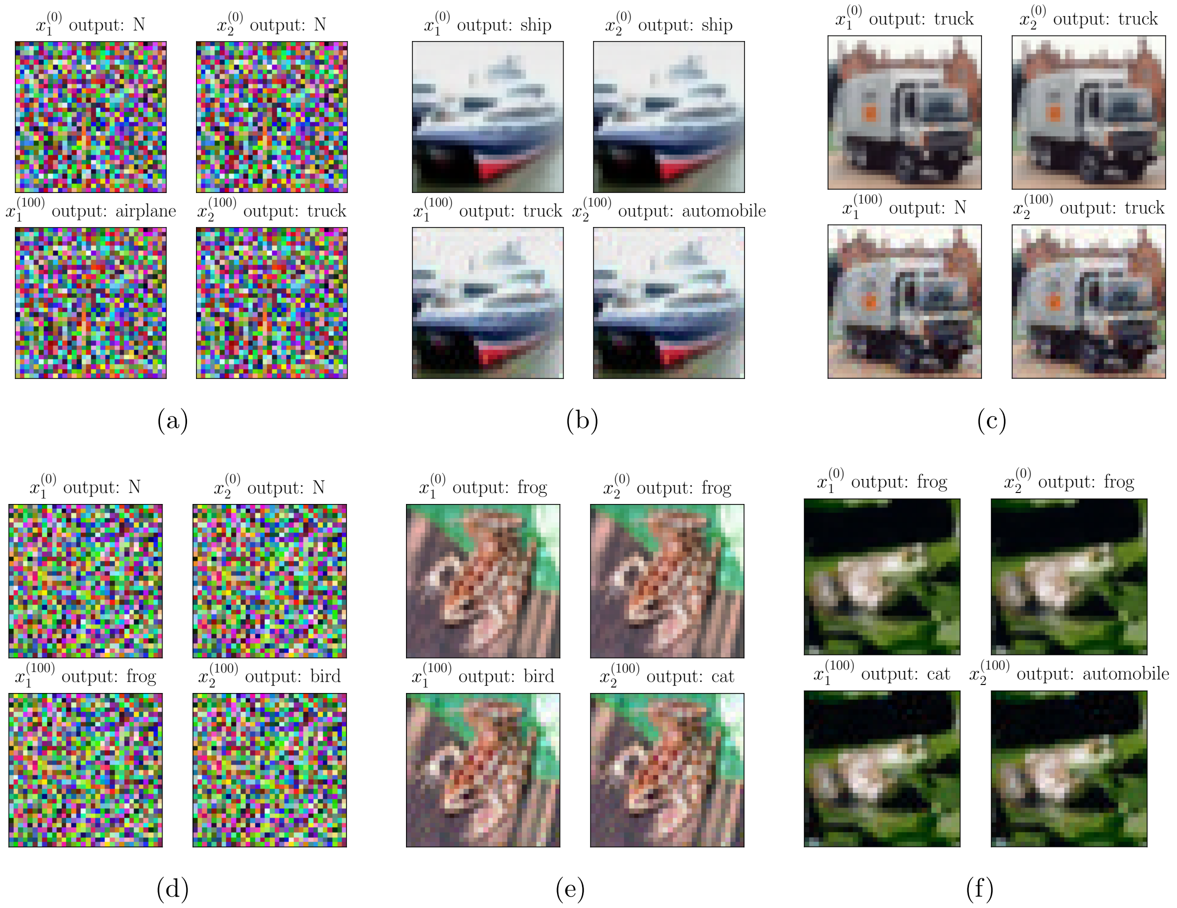

$x_1^{(0)}$ output: N
$x_2^{(0)}$ output: N
$x_1^{(100)}$ output: airplane
$x_2^{(100)}$ output: truck
(a)
$x_1^{(0)}$ output: ship
$x_2^{(0)}$ output: ship
$x_1^{(100)}$ output: truck
$x_2^{(100)}$ output: automobile
(b)
$x_1^{(0)}$ output: truck
$x_2^{(0)}$ output: truck
$x_1^{(100)}$ output: N
$x_2^{(100)}$ output: truck
(c)
$x_1^{(0)}$ output: N
$x_2^{(0)}$ output: N
$x_1^{(100)}$ output: frog
$x_2^{(100)}$ output: bird
(d)
$x_1^{(0)}$ output: frog
$x_2^{(0)}$ output: frog
$x_1^{(100)}$ output: bird
$x_2^{(100)}$ output: cat
(e)
$x_1^{(0)}$ output: frog
$x_2^{(0)}$ output: frog
$x_1^{(100)}$ output: cat
$x_2^{(100)}$ output: automobile
(f)

Figure 5.4: Global adversarial sample pairs for the CIFAR10 model generated by GEVMCMC.

settings with their local attack counterparts L-IFGSM and L-PGD, respectively. The number of GEVMCMC initial G-PGD warm-up rounds is 10 for MNIST and 30 for CIFAR10. The block size B is set to be 59. Three parameters for the proposal distribution for MNIST are $\lambda_m = 1.2\epsilon_{MNIST}, \lambda_0 = 0.3\epsilon_{MNIST}, p_B = 0.95$, and for CIFAR10 they are set to be $\lambda_m = 4.8\epsilon_{CIFAR10}, \lambda_0 = 0.6\epsilon_{CIFAR10}, p_B = 0.99$.

5.5.3 Generated global adversarial sample pairs.

The proposed global adversarial attack methods can generate diverse global adversarial pairs which are rather different from typical local attacks, representing a new type of

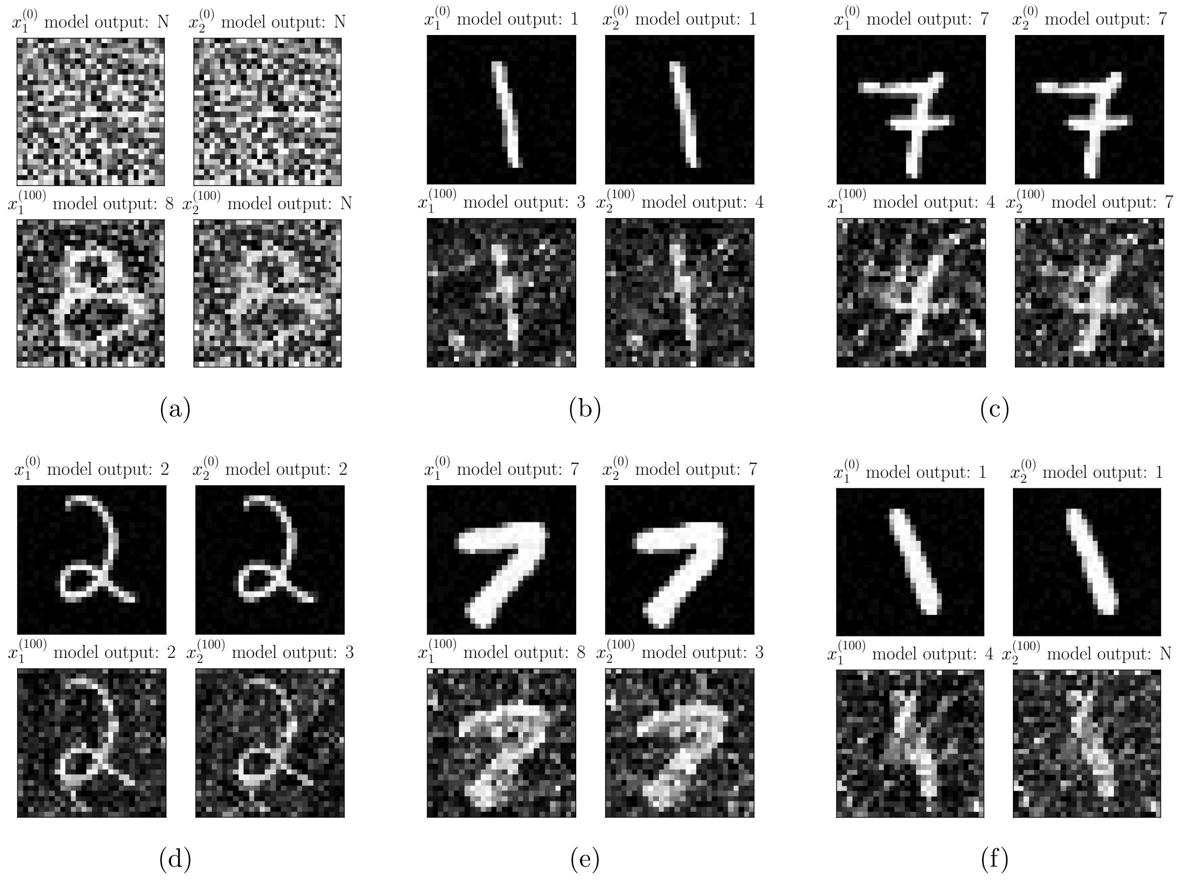

$x_1^{(0)}$ model output: N
$x_2^{(0)}$ model output: N
$x_1^{(100)}$ model output: 8
$x_2^{(100)}$ model output: N
(a)
$x_1^{(0)}$ model output: 1
$x_2^{(0)}$ model output: 1
$x_1^{(100)}$ model output: 3
$x_2^{(100)}$ model output: 4
(b)
$x_1^{(0)}$ model output: 7
$x_2^{(0)}$ model output: 7
$x_1^{(100)}$ model output: 4
$x_2^{(100)}$ model output: 7
(c)
$x_1^{(0)}$ model output: 2
$x_2^{(0)}$ model output: 2
$x_1^{(100)}$ model output: 2
$x_2^{(100)}$ model output: 3
(d)
$x_1^{(0)}$ model output: 7
$x_2^{(0)}$ model output: 7
$x_1^{(100)}$ model output: 8
$x_2^{(100)}$ model output: 3
(e)
$x_1^{(0)}$ model output: 1
$x_2^{(0)}$ model output: 1
$x_1^{(100)}$ model output: 4
$x_2^{(100)}$ model output: N
(f)

Figure 5.5: Global adversarial sample pairs for the MNIST model generated by GEVMCMC.

DNN vulnerability. Fig. 5.4 and Fig. 5.5 show a set of global adversarial example pairs generated by GEVMCMC for the two datasets. In each sub-figure, the two images on the top are the starting pair of two identical starting images. The two at the bottom are the final global adversarial pair generated after 100 rounds of optimization steps. "N" indicates that the class label predicated by the model is "meaningless".

Compared to standard local adversarial attacks, the proposed global adversarial pairs are much more diverse and intriguing. For instance, it is possible to start with two identical random meaningless image but end up with some other two random images that are very similar to each other but have different legitimate class labels predicted by the model such as ones in Fig. 5.4(a) and 5.4(d). We can also start from two identical images of a legitimate predicted class, and end up with two images with predicated labels that are different from each other and are also different from the starting label, as shown in Fig. 5.4(b), 5.4(e), 5.4(f), 5.5(b) and 5.5(e). Clearly, the existing local attacks such as FGSM, IFGSM, and PGD are not able to generate such complex global adversarial scenarios, which reveal additional hidden vulnerabilities of the model.

Importantly, the existing local adversarial attacks cannot explore the input space beyond the training or testing dataset due to the perturbation constraint. In contrast, the proposed global adversarial attacks are very appealing in the following way: they may find a path towards unseen input space and check the model robustness along the way. For instance, we may start from some random testing image (Fig. 5.5(a)) or original testing image (Fig. 5.5(c)), and end up with a completely different image pair which may be both recognized by the humans as a legitimate class, however, the model predicts different labels for them. For instance, the final pairs in the Fig. 5.5(a) and Fig. 5.5(c) may be recognized as "8" and "4", respectively, which are completely different from their starting label.

Table 5.1: Natural MNIST model (w/o local adversarial training) adversarial attack results.

Method	Start from Original Test Image			Start from Random Test Image		
	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$
L-FGSM	18.18%	20.94	0.77	5.82%	6.43	0.16
L-IFGSM	29.80%	21.76	1.17	51.96%	7.64	1.18
L-PGD	29.59%	21.73	1.16	49.44%	7.57	1.13
G-FGSM	99.14%	23.40	10.78	99.01%	20.04	9.62
G-IFGSM	99.92%	22.21	9.19	100.00%	12.05	5.94
G-PGD	99.94%	20.44	10.18	100.00%	11.94	6.58
GEVMCMC	99.93%	24.93	14.91	100.00%	19.98	12.67

Table 5.2: Adversarially-trained MNIST model adversarial attack results.

Method	Start from Original Test Image			Start from Random Test Image		
	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$
L-FGSM	3.28%	15.14	0.11	0.00%	0.00	0.00
L-IFGSM	3.89%	15.56	0.13	0.00%	0.06	0.00
L-PGD	3.88%	15.57	0.13	0.00%	0.05	0.00
G-FGSM	98.07%	17.79	8.19	97.50%	19.09	7.63
G-IFGSM	99.47%	14.25	7.84	99.56%	17.81	11.43
G-PGD	99.50%	18.08	8.61	99.58%	19.59	12.07
GEVMCMC	99.38%	18.28	9.91	99.55%	18.06	12.04

Table 5.3: Natural CIFAR10 model (w/o local adversarial training) adversarial attack results.

Method	Start from Original Test Image			Start from Random Test Image		
	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$
L-FGSM	26.77%	11.42	1.90	0.00%	0.01	0.00
L-IFGSM	44.80%	12.06	3.58	0.00%	0.02	0.00
L-PGD	43.92%	12.04	3.51	0.00%	0.02	0.00
G-FGSM	95.96%	12.75	8.72	95.15%	11.10	5.48
G-IFGSM	99.68%	12.90	11.02	98.36%	10.76	6.45
G-PGD	99.71%	13.55	11.30	98.33%	11.02	6.70
GEVMCMC	99.58%	13.18	8.94	98.37%	11.02	7.18

Table 5.4: Adversarially-trained CIFAR10 model adversarial attack results.

Method	Start from Original Test Image			Start from Random Test Image		
	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$	Attack Rate	Max $\mathcal{L}$	Avg. $\mathcal{L}$
L-FGSM	19.29%	10.97	0.83	0.00%	0.00	0.00
L-IFGSM	21.25%	11.03	0.92	0.00%	0.00	0.00
L-PGD	21.19%	11.03	0.92	0.00%	0.00	0.00
G-FGSM	95.29%	9.62	4.11	90.47%	6.57	2.52
G-IFGSM	98.23%	9.83	4.60	95.85%	6.99	2.94
G-PGD	98.26%	10.54	4.77	95.88%	6.97	3.03
GEVMCMC	98.25%	10.89	5.47	95.89%	6.66	3.67

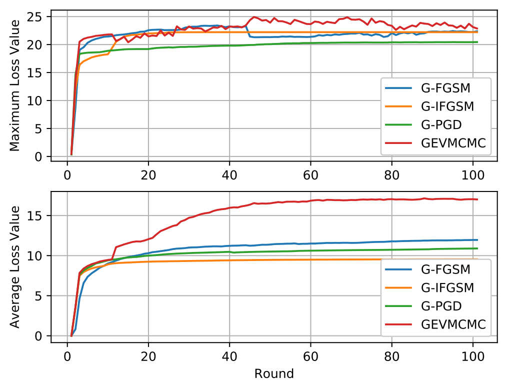
5.5.4 Local vs. global adversarial attacks

Table 5.1-5.4 show the global adversarial attack results for the MNIST and CIFAR10 models without and with local adversarial training. If the model predictions for the two examples in a generated global adversarial pair are different, we regard this pair as one successful global attack. In these tables, the attack rate is defined as the ratio between the successful number of attacks over the total number of trails which is 10,000. The Max loss and Avg. loss are the maximum loss and average loss found in the attack process.

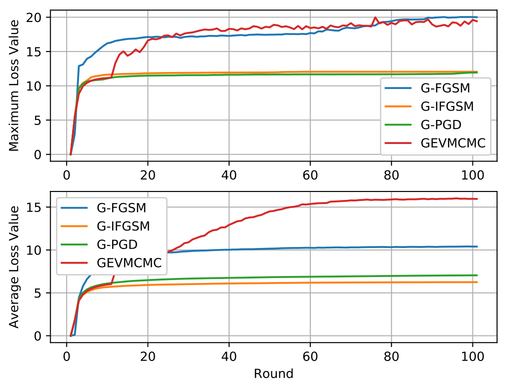
Attacking the natural MNIST and CIFAR10 models using a local attack method can reach a reasonablly high attack rate. For instance, L-PGD has an attack rate of 29.59% in the case of the natural MNIST, which drops down to 3.88% for the case of the adversarially-trained MNIST model, showing that adversarial training using local adversarial examples does improve the defense to such local attacks. However, all global adversarial attack methods achieve almost 100.00% attack rate and produce much higher average loss values compared to the local attack methods, regardless whether local adversarial training is performed or not. It is evident that adversarial training based on local adversarial examples shows none or little defense to global attacks. This indicates the effectiveness of the proposed global attack methods, and equally importantly, suggests

Table 5.5: Comparison between GEVMCMC and other proposed global attacks when starting from the same 100 original test (“Test”) or 100 random testing (“Rand”) example pairs. Each entry shows the number of cases out of the total 100 cases where the final adversarial pair generated by GEVMCMC has a loss higher than the one generated by the other method.

Method	MNIST				CIFAR10			
	Natural		Adv.-Trained		Natural		Adv.-Trained	
	Test	Rand	Test	Rand	Test	Rand	Test	Rand
G-FGSM	86	95	69	81	33	72	76	67
G-IFGSM	98	100	92	90	69	85	84	92
G-PGD	97	100	80	67	12	73	74	90



(a) Testing image starting image pair



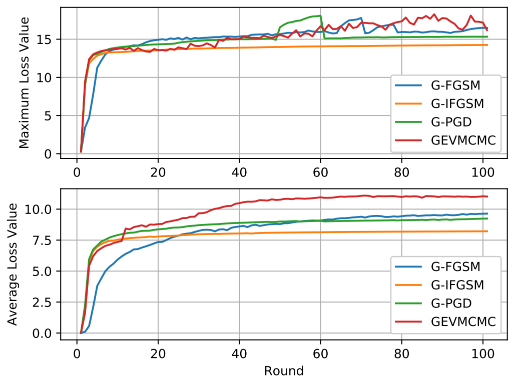
(b) Random image starting image pair

Figure 5.6: Loss value found by global adversarial attack in each round for the natural MNIST model.

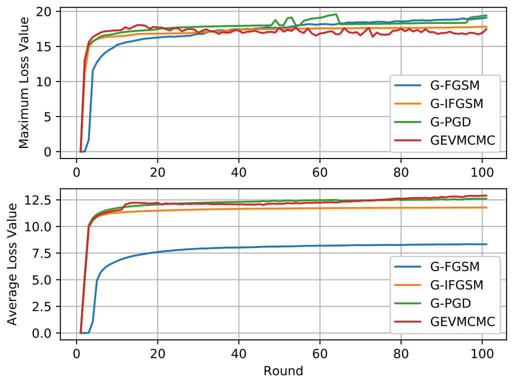
that global adversarial examples defined in this paper must be coped with when training robust DNN models.

5.5.5 Comparison of the proposed global adversarial attack methods

Table 5.5 compares the two types of the proposed global attack methods when starting from the same 100 original test (“Test”) or 100 random testing (“Rand”) example pairs.

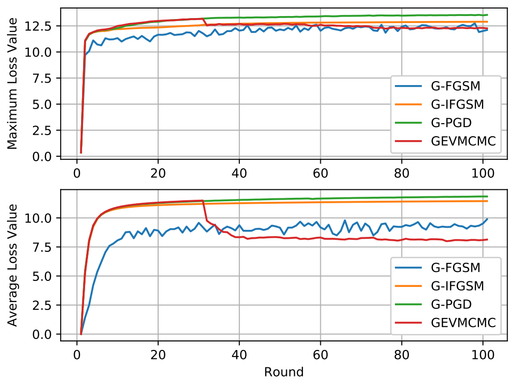


(a) Testing image starting image pair

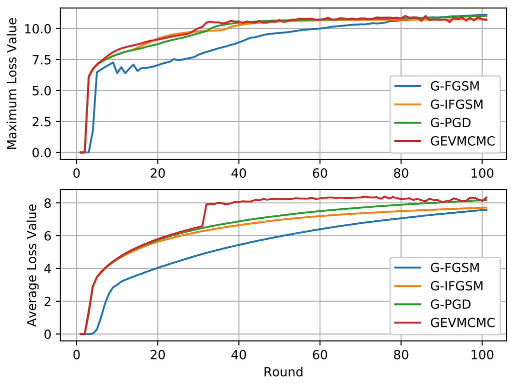


(b) Random image starting image pair

Figure 5.7: Loss value found by global adversarial attack in each round for the adversarially-trained MNIST model.

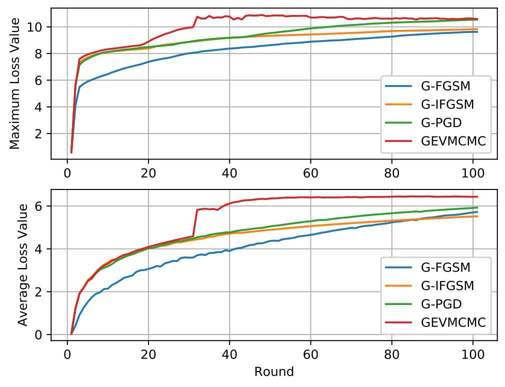


(a) Testing image starting image pair

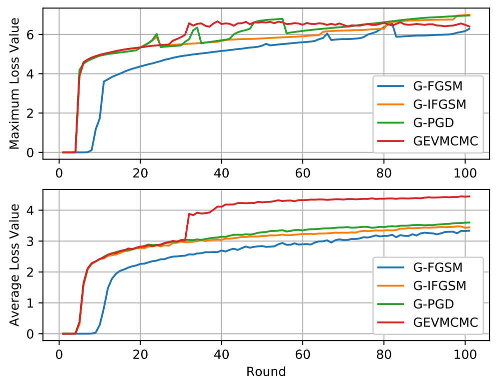


(b) Random image starting image pair

Figure 5.8: Loss value found by global adversarial attack in each round for the natural CIFAR10 model.



(a) Testing image starting image pair



(b) Random image starting image pair

Figure 5.9: Loss value found by global adversarial attack in each round for the adversarially-trained CIFAR10 model.

Each entry shows the number of cases out of the total 100 cases where the final adversarial pair generated by GEVMCMC has a loss higher than the one generated by the other method. Most entries in the table are larger than 50, implying that GEVMCMC finds worse adversarial example pairs than other global adversarial attack methods. We further show the maximum loss value and average loss value of the adversarial example pairs in each round in Fig. 5.6 - 5.9. After the initial warm-up rounds using G-PGD, the loss found by GEVMCMC increases rapidly, and ends up with a much larger value compared to that of the other global adversarial attack methods, which tend to converge at a local maximum. The only case in which GEVMCMC does not beat other global adversarial attack methods is the attacking of the natural CIFAR10 model starting with the original testing images, showing the overall better effectiveness of GEVMCMC.

5.6 Summary

We propose a new global adversarial example pair concept and formulate the corresponding global adversarial attack problem to assess the robustness of DNNs over the entire input space without human data labeling. We further propose two families of global adversarial attack methods: (**1**) alternating gradient global adversarial attacks and (**2**) extreme-value-guided MCMC sampling global attack (GEVMCMC), demonstrating that DNN models even trained with local adversarial training are vulnerable to this new type of global attacks. Our attack methods are able to generate diverse and intriguing global adversarial which are very different from typical local attacks and shall be taken into consideration when training a robust model. GEVMCMC demonstrates the overall best performance among all proposed global attack methods due to its probabilistic nature.

Chapter 6

Advanced Outlier Detection Using Unsupervised Learning for Screening Potential Customer Returns

This chapter aims to study modern machine learning techniques on outlier detection in view of screening defect escapes to customers. The purposes of this chapter are two-fold. First, we assess the application of the advancements in anomaly detection [82, 83] from the field of machine learning to the targeted testing problem and observe their limitations. We consider several popular unsupervised anomaly detection methods trained using normal data and proposed in machine learning. Tree-based methods such as isolation forest [84] flag a detected anomaly when the average path length to the leaves in the forest falls below a threshold. One-class support vector machine (OCSVM) [85, 86, 87] bounds the normal training data within a tight boundary, which is used to separate normal data

©2020 IEEE. Reprinted, with permission, from Hanbin Hu, Nguyen Nguyen, Chen He, and Peng Li, "Advanced Outlier Detection Using Unsupervised Learning for Screening Potential Customer Returns", IEEE International Test Conference (ITC), 2020.

from abnormal data. Autoencoders [88, 89] or generative adversarial networks (GAN) [90, 91] are among the most successful methods where any observed large reconstruction error signifies anomaly.

While demonstrating certain degrees of success in other anomaly detection problems, we show that the aforementioned methods do not work well for the challenging problem of identifying extremely-rare customer failures so as to minimize defect escape to customers. Hence, the second purpose of this chapter is to bring a new perspective to post-silicon testing by adapting the geometric transformation based deep anomaly image detection approach [29], which leverages supervising learning for solving the unsupervised learning problem. More specifically, [29] creates a set of self-labeled images by transforming each example in the given raw training dataset using a number of geometric transformations. Each transformed image is labeled using the index of the transformation applied. A multi-class classifier is trained using the self-labeled training data. During inference, an unseen image is first transformed using the same set of geometric transformations. The resulting transformed images are classified by the trained multi-class classifier. The goodness of the classification decisions is considered as a normality score, which is used to signify detection of abnormality when the normality score drops to a low value. We introduce two key modifications to make this self-labeling approach viable for the intended extremely-rare customer failure detection problem. First, we replace geometric transformations used for images by nonlinear transformations suitable for processing test data. Second, we introduce a family of reversible information lossless transformations to boost the performance and robustness of the self-labeling methods. Experimentally, we demonstrate that the proposed self-labeling approach significantly outperforms the other methods in terms of prediction accuracy and robustness using a large set of public datasets and real industrial post-silicon test data.

6.1 Unsupervised Outlier Detection

6.1.1 Problem Formulation

Consider a D-dimensional input space $\mathcal{X} \subseteq \mathbb{R}^D$ containing all potential inputs, e.g. parametric post-silicon test results. Let $\mathcal{X}_N \subseteq \mathcal{X}$ and $\mathcal{X}_A \subseteq \mathcal{X}$ represent normal and abnormal inputs, e.g. the test results of good vs. failing (outlier) chips, with $\mathcal{X}_N \cap \mathcal{X}_A = \emptyset$ and $\mathcal{X}_N \cup \mathcal{X}_A = \mathcal{X}$. To classify an input $\mathbf{x} \in \mathcal{X}$ as normal or abnormal, an unsupervised learning method learns a binary classification function $f : \mathcal{X} \rightarrow \{0, 1\}$, where "0" indicates normality (true negative example), i.e., $\mathbf{x} \in \mathcal{X}_N$; "1" represents outlier (true positive example), i.e., $\mathbf{x} \in \mathcal{X}_A$.

Without labeled outlier data due to its scarcity, well developed supervised learning cannot be applied. Instead, as shown in Fig. 6.1, one learns a score function $s : \mathcal{X} \rightarrow \mathbb{R}$ through certain unsupervised learning algorithm to assess the normality of the seen normal data. During inference phase, the larger value of $s(\mathbf{x})$, the more likely the unseen data is normal. With a specified decision threshold s_{Th}, the binary classification model f trained without supervision is

$$f_{Th}(\mathbf{x}) = \begin{cases} 0 & s(\mathbf{x}) \geq s_{Th} \\ 1 & s(\mathbf{x}) < s_{Th} \end{cases} \tag{6.1}$$

Note that the input features may not fully reveal the outlier information in practice, implying $\mathcal{X}_N \cap \mathcal{X}_A \neq \emptyset$. However, the score function definition still works properly under this case, indicating how likely the given input x appears to be an outlier.

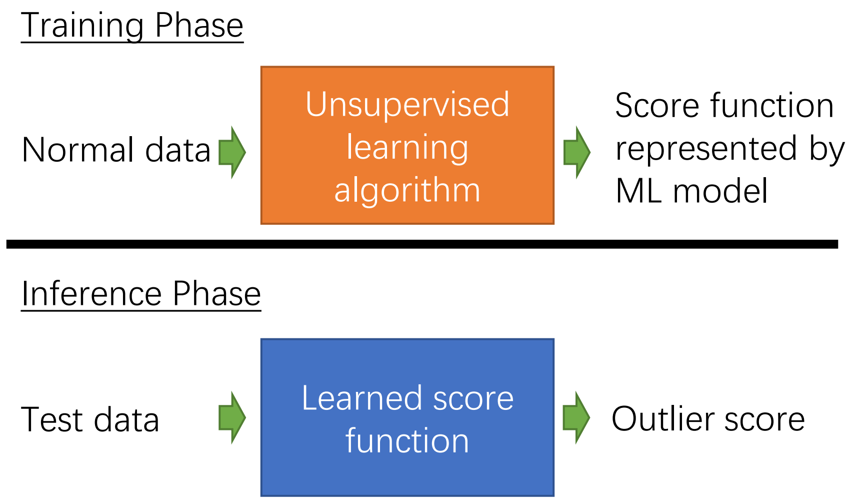
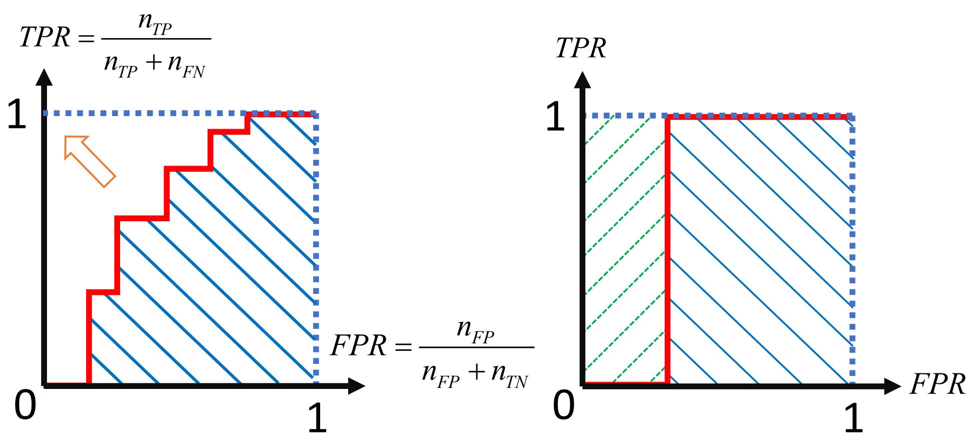


Figure 6.1: Unsupervised outlier detection illustration.

6.1.2 Performance Metric of Machine Learning Models

The choice of the decision threshold s_{Th} in (6.1) has a large impact on the quality of outlier detection. It is unfair to select specific thresholds when comparing different outlier detection methods since the scale and the distribution of the score function value vary widely from methods to methods. We apply the widely adopted Area Under Receiver Operation Characteristic (ROC) curve (AUROC) to compare different models without relying on specific thresholds.



(a) General case.

(b) Single outlier case

Figure 6.2: ROC curve and AUROC.

As shown in Fig. 6.2, the ROC curve characterizes the true positive rate (TPR) and false positive rate (FPR) of a model as the threshold s_{Th} is swept. If s_{Th} exceeds the maximum score function value of a testing dataset, all data will be considered as outliers, making both TPR and FPR 1.0; the other way around will make both TPR and FPR 0. The ROC curve is monotonically increasing between $(0, 0)$ and $(1, 1)$. Improved outlier detection performance leads to larger TPR and lower FPR values. Correspondingly, the ROC curve would be pushed towards the top left corner as shown in Fig. 6.2(a), with the best possible AUROC value of 1.0. For example, there is only one customer failure in the industrial automotive microcontroller datasets we use. Correspondingly, there is a sharp transition in the ROC as shown in Fig. 6.2(b), where the green area ($1 - AUROC$ where only one failure exists) can be interpreted as the yield loss when no defect escape occurs.

Compared to other metrics like Area Under Precision-Recall (AUPR), AUROC takes a more balanced consideration over both abnormal and normal data, which efficiently captures outliers and minimizes yield loss.

6.1.3 Review of Traditional Unsupervised Learning Models

Here gives a brief review of 4 different unsupervised anomaly detection methods: Gaussian model, One-class SVM, Isolation forest and Autoencoder.

Gaussian Model

One typical category of anomaly detection methods is to estimate the data distribution given the training samples, and then mark low-probability instances as anomaly. Gaussian model [92] is one of the most popular assumptions for the data distribution, which especially suits for the post-silicon test, as most of data follows a Gaussian distribution. Specifically, given N normal training samples $\left\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\right\}$, the mean and covariance matrix are estimated as follows.

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} \tag{6.2}$$

$$\hat{\mathbf{C}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}^{(i)} - \hat{\mu}\right)\left(\mathbf{x}^{(i)} - \hat{\mu}\right)^{\mathrm{T}} \tag{6.3}$$

Thus, clearly, the score function is defined as the probability distribution of the estimated Gaussian distribution as below.

$$s\left(\mathbf{x}\right) = \mathcal{N}\left(\mathbf{x}; \hat{\mu}, \hat{\mathbf{C}}\right) \tag{6.4}$$

One-class SVM

One-class support vector machine (OCSVM) proposed in [85] separates all the data from the origin with maximum margin in the feature space corresponding to the kernel function. In general, instead of directly looking at the probability distribution of the normal sample occurrence, one-class SVM attempts to map the data into a feature space and enclose the normal data into a small region. This results in a binary function which captures regions in the input space where most of the normal data live. The training process of one-class SVM is governed by a quadratic programming minimization problem as stated below.

$$\min_{\omega,\xi_i,\rho}\ \frac{1}{2}\left\|\omega\right\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho; \tag{6.5}$$

$$\text{s.t. } \omega \cdot \phi\left(\mathbf{x}^{(i)}\right) \geq \rho - \xi_i; \tag{6.6}$$

$$\xi_i \geq 0, \tag{6.7}$$

where $\phi(\cdot)$ specifies the kernel function to be used for which radial basis function (RBF) is a common choice, ν is a hyperparameter characterizing the solution by setting the upper bound of the outliers inside the training dataset and the lower bound for the number of support vectors. According to the distance to the decision boundary in the feature space, represented by ρ, the score function can be depicted using

$$s(\mathbf{x}) = \omega \cdot \phi\left(\mathbf{x}^{(i)}\right) - \rho \tag{6.8}$$

Note that the signed version of the previous function is used to give a binary classification of the anomaly detection in [85].

Isolation forest

In addition to the distribution estimation and data separation in the feature space, a tree-based method called isolation forest [84] takes a disparate approach to distinguish the anomaly from the normal data. The isolation forest consists of an ensemble of isolation trees (iTrees). More formally, for each node T in an iTree, T is either an external node with no child or an internal node with one test and exactly two children (T_L, T_R). The test at node T consists of an attribute q and a split value p, and based on whether $q < p$ it will traverse the data point to either T_L or T_R. In order to build such an iTree, a subset of entire dataset $\mathbf{X}' \subset \left\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\right\}$ is randomly selected. The iTree is generated by recursively partitioning, and then a training algorithm recursively partitions $\mathbf{X}'$ by randomly selecting an attribute q and a split value p until there is only one data point in the node or all the data share the same value.

The underlying principle of anomaly detection for isolation forest is that the anomaly data is more likely to reach an external node with a smaller height (distance to the iTree root), as all the iTrees are generated randomly. Therefore, its score function is characterized by the average height to reach the external nodes in the ensemble of iTrees as follow.

$$s(\mathbf{x}) = -2^{-\mathbf{E}h(\mathbf{x})/c(N)}, \tag{6.9}$$

where $c(N)$ is a constant normalization factor related to the sample size N. The negative sign is added to be consistent with the definition of score function in this chapter.

Autoencoder

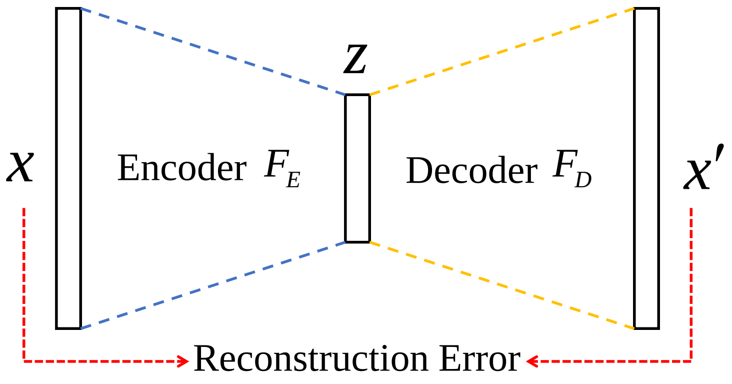


Figure 6.3: Autoencoder illustration.

There exists another popular anomaly detection method which is reconstruction-based enabled by an autoencoder. Similar ideas using reconstruction for anomaly detection are widely used in recent work [83, 88, 89]. First, the original data $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ is encoded (compressed) into a latent variable $\mathbf{z} = F_E(\mathbf{x}) \in \mathcal{Z} \subseteq \mathbb{R}^d$ with $d \ll D$ typically, and then decoded (reconstructed) into the original space with $\mathbf{x}' = F_D(\mathbf{z})$, as illustrated in Fig. 6.3. The encoder and the decoder are usually fully-connected neural networks for numerical data processing or convolutional neural networks for image processing. The entire autoencoder is trained to minimize the overall reconstruction error (loss function for training as shown below) so that a good embedding representation of the normal data can be learnt in the low-dimensional latent space $\mathcal{Z}$.

$$\mathcal{L} = \sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - \mathbf{x}'^{(i)} \right\|^2 \tag{6.10}$$

With an outlier data, since it doesn't follow the normal data embedding representation, it is expected to observe a large reconstruction error, which is used to defined the score function for the autoencoder.

$$s(\mathbf{x}) = - \left\| \mathbf{x} - F_D \left(F_E (\mathbf{x}) \right) \right\|^2 \tag{6.11}$$

6.2 Self-Labeling Unsupervised Outlier Detection

6.2.1 Self-Labeling via Transformation

Based on the discussions in the Section 6.1, we aim to learn a robust and reliable score function $s : \mathcal{X} \rightarrow \mathbb{R}$ using normal training data only. Unsupervised learning for extremely-rare failure detection is challenging. Motivated by the self-labeling approach for anomaly image detection of [29], we convert this unsupervised learning problem to one that is based on multi-class classification with self-labeled training data.

Consider K distinct transformation functions $T^{(1)}, T^{(2)}, \cdots, T^{(K)}$, each defining a mapping to a m-dimensional feature space: $T^{(i)} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, where $i \in [1, K]$. For a given training dataset X with N examples, we apply all K transformations to each sample, resulting in a transformed training dataset with KN examples. Each newly transformed example is labeled by its corresponding transformation applied. Formally, the resulting labeled training dataset is

$$\left\{ \left(T^{(i)} \left(\mathbf{x}^{[j]} \right), T^{(i)} \right) | i \in [1, K], j \in [1, N] \right\} \tag{6.12}$$

Note that each label is given by the type of the transformation performed without involving any actual labeling effort nor abnormal data. The adopted transformations can be regarded as an approach for nonlinear feature extraction through which the original input space $\mathcal{X} \subseteq \mathbb{R}^n$ is mapped into a feature space $\mathcal{F} \subseteq \mathbb{R}^m$.

6.2.2 Proposed Self-labeling Unsupervised Outlier Detection

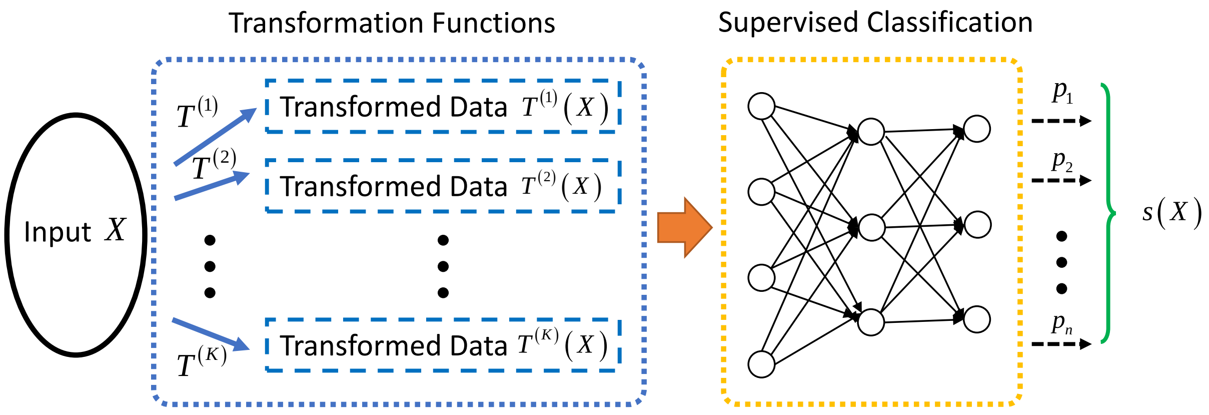

Transformation Functions
Supervised Classification
Input X
$T^{(1)}$
$T^{(2)}$
$T^{(K)}$
Transformed Data $T^{(1)}(X)$
Transformed Data $T^{(2)}(X)$
Transformed Data $T^{(K)}(X)$
p_1
p_2
p_n
$s(X)$

Figure 6.4: Self-labeling unsupervised outlier detection framework.

Fig. 6.4 highlights the proposed self-labeling outlier detection approach. During training, self-labeling is executed first per (6.12) to generate the transformed training dataset over which a multi-class classifier is trained to well classify each example to the corresponding label (transformation). During inference, similarly, given an input $\mathbf{x}$, K transformed inputs $\left\{T^{(1)}(\mathbf{x}), T^{(2)}(\mathbf{x}), \cdots, T^{(K)}(\mathbf{x})\right\}$ are obtained by applying the K transformation functions. For each transformed $T^{(i)}(\mathbf{x})$, the classifier outputs a K-dimensional probability vector $\mathbf{p}\left(T^{(i)}(\mathbf{x})\right)$ with each k-th element specifying the predicted likelihood for $T^{(i)}(\mathbf{x})$ to fall under class k, i.e. transformed by k-th transformation.

When the classifier is well trained using the normal data, it would classify a new unseen normal input $\mathbf{x}$ by outputting: $\mathbf{p}_i\left(T^{(i)}(\mathbf{x})\right) \approx 1$ and $\mathbf{p}_i\left(T^{(k)}(\mathbf{x})\right) \approx 0, i \neq k$ as the transformed unseen normal data is likely to locate within the transformed normal training data distribution. However, for an abnormal input, it is likely that $\mathbf{p}_i\left(T^{(i)}(\mathbf{x})\right)$ is significantly lower than 1.0 as the transform outlier data may deviate from the transformed normal data distribution. Accordingly, we select the following score function

$$s(\mathbf{x}) = \sum_{i=1}^{K} \mathbf{p}_i\left(T^{(i)}(\mathbf{x})\right) \tag{6.13}$$

An input is detected as an outlier if $s(\mathbf{x}) \ll K$. This self-labeling outlier detection is illustrated in Fig. 6.5.

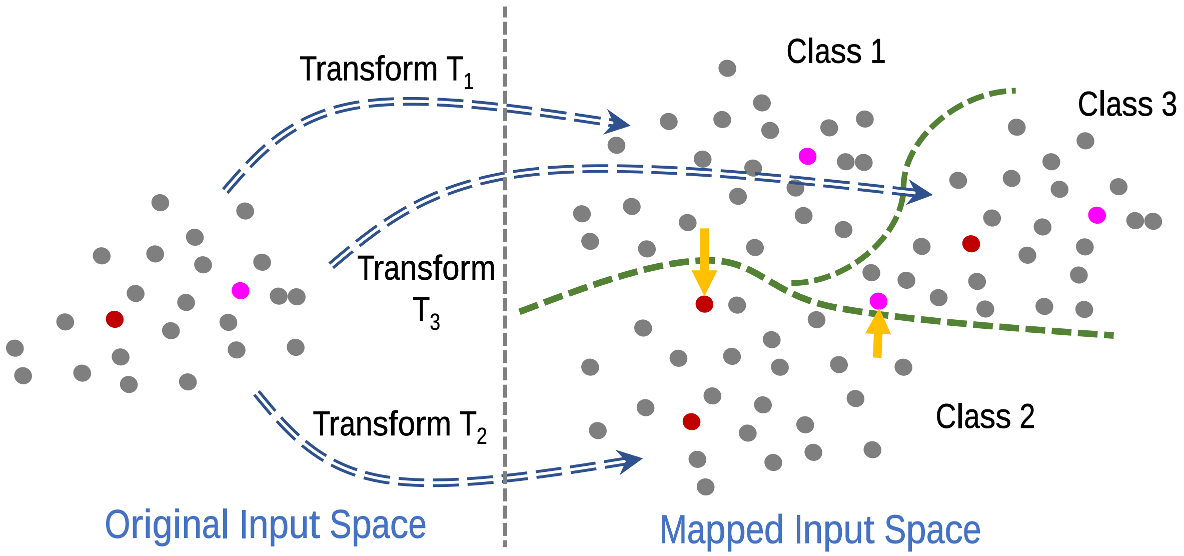


Figure 6.5: Outlier detection via 3 transformations. Gray points: normal data; Red/pink points: outliers. Yellow arrows: misclassification of mapped outliers signifies anomaly detection.

6.3 Design of Transformation Functions

One major challenge in the proposed self-labeling approach is to select proper transformation functions. For example, if $T^{(i)} = T^{(j)}$, the classifier cannot distinguish between the i-th and j-th class as they correspond to identical transformed input data location. Our key idea is to select a set of distinct information lossless transformation functions to retain sufficient statistics for the original data. Consider that our original data $\mathbf{x}$, the transformed data $T(\mathbf{x})$, and the final score function $s(\mathbf{x})$ formulate a Markov chain $s(\mathbf{x}) \longleftrightarrow T(\mathbf{x}) \longleftrightarrow \mathbf{x}$. According to the data processing inequality, we have the mutual information follows $I(s(\mathbf{x}); T(\mathbf{x})) \leq I(s(\mathbf{x}); \mathbf{x})$. In order to maximize the mutual information after the transformation, our method is to make the transformed data fully recoverable, without information loss, after the transformation.

Inspired by the recent developments in reversible neural networks [61, 57], we propose an reversible architecture for the transformation functions to fully retain the original data information. Suppose $\mathbf{x} = (x_1, x_2, \cdots, x_D)^{\mathrm{T}}$ has D features, we partition the indices of the D features into two sets p_1 and p_2 with equal size (add one artificial feature if D is odd), where $p_1 \cap p_2 = \emptyset$ and $p_1 \cup p_2 = [1, D] \cap \mathbb{N}$. With that, we can obtain two new vectors with the corresponding features as $\mathbf{x}_{p1}$ and $\mathbf{x}_{p2}$. Then the transformed data $\mathbf{y}$ (combining two parts $\mathbf{y}_{p1}$ and $\mathbf{y}_{p2}$) is given by the reversible transformation as follows.

$$\mathbf{y}_{p1} = \mathbf{x}_{p1} + G(\mathbf{y}_{p2}) \tag{6.14}$$

$$\mathbf{y}_{p2} = \mathbf{x}_{p2} + F(\mathbf{x}_{p1}) \tag{6.15}$$

where F and G are two arbitrary functions. Through solving the previous equations, the original input $\mathbf{x}$ can be fully recovered with knowing the output $\mathbf{y}$, showing the information lossloss property of the reversible transformation block. In order to easily generate a large number of distinct transformations, the transformation function should be flexible enough to be configured. We considered three different approaches to increase the flexibility of the transformation: 1) function choices for the reversible block; 2) feature permutation; 3) cascade architecture.

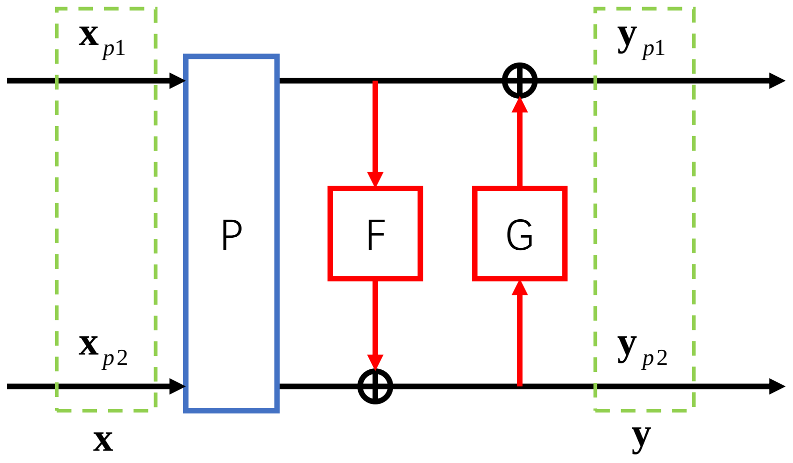


Figure 6.6: Reversible lossless transformation block.

6.3.1 Function Choices for the Reversible Block

The arbitrariness of the two functions F and G provides a lot of freedom to be designed. Without adding additional training and computational cost for F and G, which are usually two neural networks, from our empirical experience, we suggest to apply two simple univariate nonlinear functions f and g for each element of the input vector. Therefore, we have $F(\mathbf{x}) = \left(f(x_1;\theta_f), f(x_2;\theta_f), \cdots, f\left(x_{D/2};\theta_f\right)\right)^{\mathrm{T}}$ and $G(\mathbf{x}) = \left(g(x_1;\theta_g), g(x_2;\theta_g), \cdots, g\left(x_{D/2};\theta_g\right)\right)^{\mathrm{T}}$, where $\theta_f \in \Theta$ and $\theta_g \in \Theta$ are the parameters for the two univariate functions from a parameter space Θ. Hence, we can create a pool of function choices, for each reversible block, we can randomly assign two functions to f and g and sample the parameters θ_f and θ_g from Θ to generate a large number of distinct transformations before training phase.

In particular, for our experimental settings, a pool of three different polynomial functions with randomized order parameter θ uniformly chosen from $\Theta = [2, \theta_{max}] \cap \mathbb{N}$ is applied, including: power polynomial functions,

$$p^{(\theta)}(x) = x^{\theta} \tag{6.16}$$

Legendre polynomial functions,

$$\begin{aligned} l^{(0)}(x) &= 1 && (6.17)\\ l^{(1)}(x) &= x && (6.18)\\ l^{(\theta)}(x) &= \frac{2\theta-1}{\theta} x l^{(\theta-1)}(x) - \frac{\theta-1}{\theta} l^{(\theta-2)}(x) && (6.19) \end{aligned}$$

and Chebyshev polynomial functions,

$$\begin{aligned} c^{(0)}(x) &= 1 && (6.20)\\ c^{(1)}(x) &= x && (6.21)\\ c^{(\theta)}(x) &= 2x c^{(\theta-1)}(x) - c^{(\theta-2)}(x). && (6.22) \end{aligned}$$

6.3.2 Feature Permutation

Beyond the function choice for each reversible block, we also add one more permutation block, as shown in Fig. 6.6, to boost the feature mixing and increase flexibility for distinct function generation. For the input of each reversible block, a random feature permutation (partition) is generated before training process and then fixed during the training and inference phase to produce two groups of data.

This technique boosts the diversity among different reversible blocks. Furthermore, with the help of the cascade architecture (introduced in the next section), this permutation will also boost feature mixing among multiple features, which serves a nonlinear feature extraction process providing a distinct view of the normal data for each transformation.

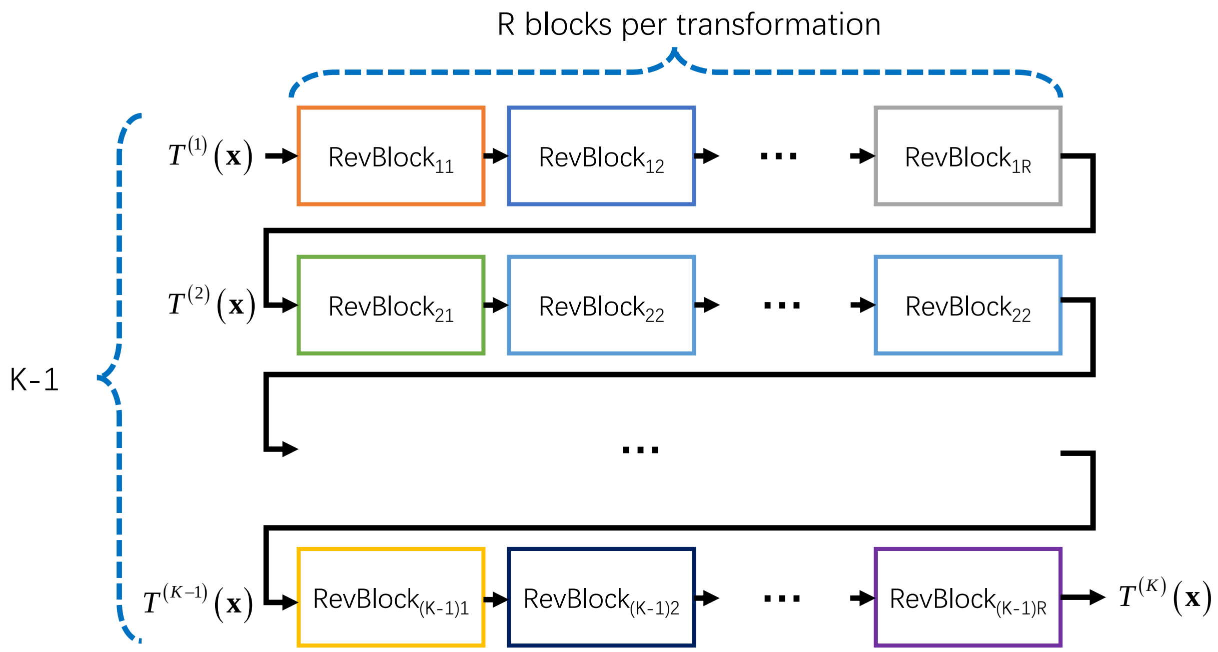


Figure 6.7: Reversible transformation architecture.

6.3.3 Overall Cascade Reversible Architecture

The overall reversible lossless transformation architecture is illustrated in Fig. 6.7. Each reversible block is marked in different color to indicate they are all distinct. We apply both techniques: function choices and feature permutation, described in Section 6.3.1 and 6.3.2, to generate a large number of distinct reversible blocks.

Here, a cascade architecture is applied. Note that the reversible lossless tranformation block can be repeated multiple times to get a single tranformation. Hence, the data will go through R reversible blocks to acquire $T^{(i+1)}(\mathbf{x})$ from $T^{(i)}(\mathbf{x})$. In order to generate K different transformations, this computation is performed $K-1$ times, assuming the first transformation is the original data point $T^{(1)}(\mathbf{x}) = \mathbf{x}$, which is already available. Thus, there exists $(K-1)R$ different reversible blocks in the entire architecture, providing a large number of distinct transformations without losing any information.

6.4 Experimental Results

6.4.1 Methods and Datasets

Using six public outlier detection datasets [93, 94] and post-silicon testing datasets for six real customer failures/returns of an advanced industrial automotive microcontroller, We demonstrate and compare the performances of several unsupervised learning methods: Gaussian model [92], one-class SVM [85], isolation forest [84], autoencoder [83], and four configurations of the proposed self-labeling outlier detection method.

Table 6.1: Public anomaly detection datasets.

Public	# Samples	# Features	# Anomaly
thyroid	3772	6	93 (2.5%)
glass	214	9	9 (4.2%)
Satimage-2	5803	36	71 (1.2%)
shuttle	49097	9	3511 (7%)
smtp	95156	3	30 (0.03%)
speech	3686	400	61 (1.65%)

For the industrial cases, the six customer failures were from several millions of parts shipped, showcasing the real-life challenges in extremely-rare defect detection. We pulled out the post-silicon testing data for the wafer lot containing the customer failure to be learned. There were around tens of thousands of parts in a wafer lot which were all manufactured around the same time with similar tools as the customer failure chip, thus providing relevant data for statistical outlier analysis. Parts in a wafer lot went through wafer test and final test at different temperatures, which we call test inserts. Each dataset (with one customer failure chip) consists of five to six test inserts. In total, we had 32 test inserts corresponding to the 6 datasets for the 6 customer failure chips. Each test insert contains up to a few thousands of parametric tests for tens of thousands of parts from the wafer lot which contains the single customer failure part. In addition, we also generate 6 additional datasets using the test inserts containing only the "critical" parametric

Table 6.2: Industrial automotive microcontroller datasets.

Datasets		Entire tests		"Critical" tests	
Chip	Insert	# Samples	# Features	# Samples	# Features
Chip 1	A	47006	104	50101	6
	B	45617	1134	45658	103
	C	41828	989	42694	28
	D	38940	769	42396	28
	E	42293	1054	42293	31
Chip 2	A	44579	104	50498	6
	B	46529	1135	46563	103
	C	44833	989	45472	28
	D	42905	780	44843	28
	E	44586	1051	44586	31
Chip 3	A	38897	102	39840	6
	B	37826	695	37831	327
	C	34544	369	34544	181
	D	34245	352	34246	153
	E	34228	1377	34228	690
Chip 4	A	16115	184	17356	6
	B	14892	379	15238	24
	C	13581	2558	13692	128
	D	13169	2587	13169	58
	E	13272	2396	13272	47
	F	7889	5592	7888	59
Chip 5	A	44282	105	50029	6
	B	47257	1151	47350	117
	C	43007	982	44862	32
	D	43893	800	44522	32
	E	44036	1108	44036	37
Chip 6	A	16591	241	18403	6
	B	16514	1521	16530	255
	C	15268	3009	16157	129
	D	15916	2645	16081	60
	E	16068	4224	16068	47
	F	16054	5526	16054	63

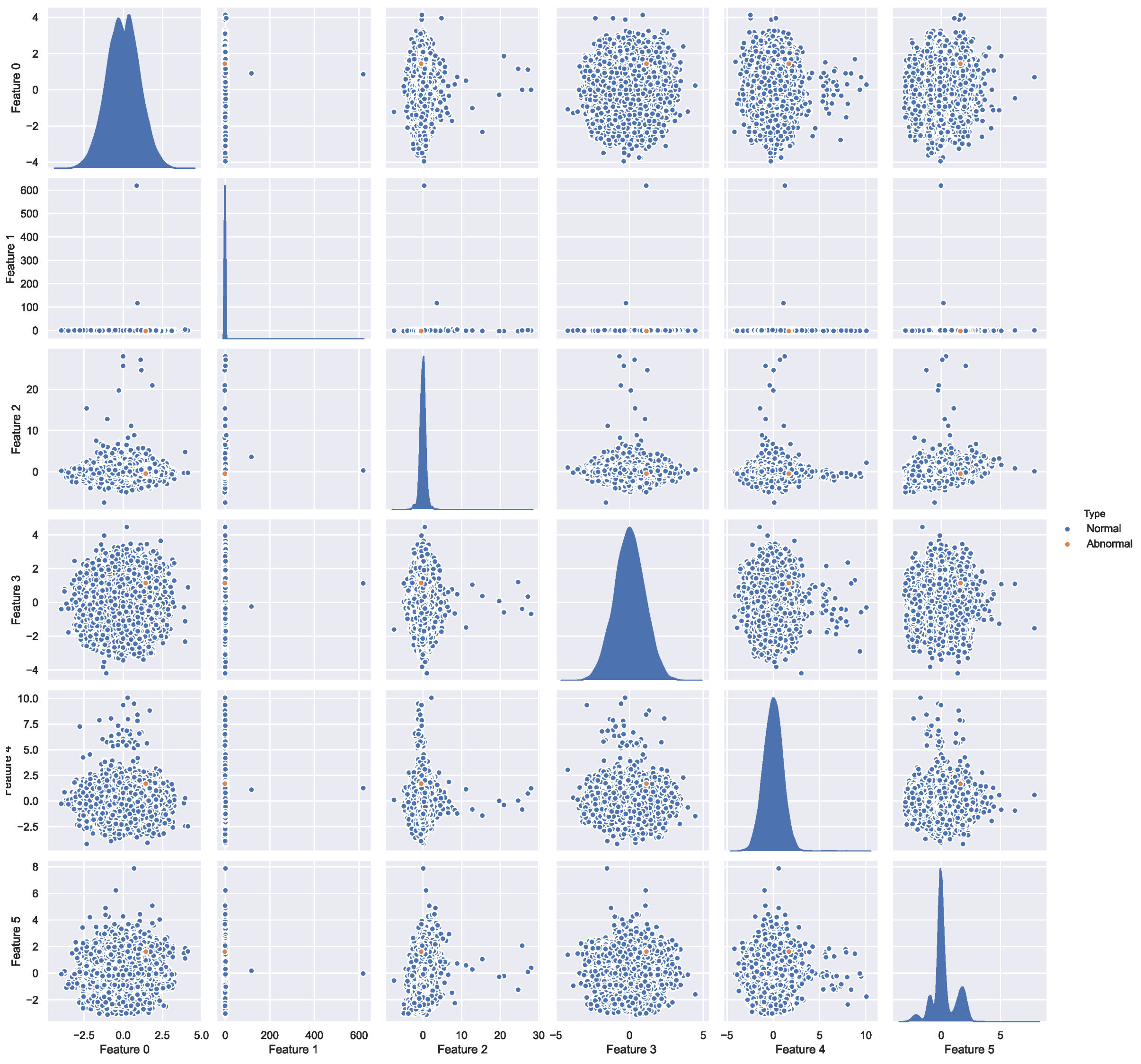


Figure 6.8: Dataset example visualization.

tests (features) suggested by expert engineers empirically based on the customer failure modes, which we call “critical” tests (inserts). Hence, the number of parametric tests (features) varied from 6 to 5,592 in the test inserts. The detailed numbers of features and examples of the public and industrial datasets are listed in Table 6.1 and Table 6.2, respectively. Furthermore, the samples with missing or invalid test values are discarded here. Therefore, For each test insert, the number of samples in the “critical” tests are larger than or equal to the one for the entire tests.

Fig. 6.8 provides the visualization of the data distribution of one example test insert (Chip 6 with the “critical” test insert A), with the single customer failure marked in orange. As we can see, the outlier point almost locates at the center of the data distribution in every dimension, which makes it extremely challenging to separate the anomaly from the normal data by only checking each single parameter or test feature.

6.4.2 Experimented method settings

For each test insert, we trained a machine learning model based on each method to screen out the customer failure part. 90% of the normal data were randomly sampled using a uniform distribution as the training dataset, and remaining 10% of normal data and the single customer failure were used as testing data.

We compared our proposed self-labeling method with several popular families of existing outlier detection techniques as reviewed in Section 6.1. Under the category of input-distribution-based approaches, we applied a Gaussian distribution model [92] to capture the mean vector and covariance matrix of normal data, and employed the probability density function as the score function. Furthermore, we adopted one-class support vector machine (OCSVM) [85] which utilized the radial basis function (RBF) kernel and its decision function as its score function. The isolation forest method [84] generated a forest of 250 trees with a resampling size of 1024, the average path length metric suggested in [84] was regarded as the score function. Moreover, we also investigated the popular reconstruction-based method, autoencoder [83]. An autoencoder with a latent size of 32 and a hidden layer of size 64 was trained over 100 epochs with a batch size of 64, and the reconstruction error of the autoencoder was selected as its corresponding score function.

For the proposed self-labeling outlier detection method, we experimented four variants of the proposed self-labeling outlier detection method. This first one, marked as *PolyTrans* later, directly used the polynomial functions as the transformation functions, and the other three adopted the reversible lossless transformation blocks. As mentioned in Section 6.3, the function pool for reversible block consists of three families of univariate polynomial functions including polynomial functions, Legendre functions and Chebyshev functions. The detailed configurations for the three reversible transformations are listed: 1) $\theta_{max} = 4$ and $R = 1$ for *RevTransA*; 2) $\theta_{max} = 5$ and $R = 1$ for *RevTransB*; 3) $\theta_{max} = 5$ and $R = 2$ for *RevTransC*. In order to obtain a fair number of distinct transformations, all the experimented variants used $K = 22$ transformations. The classifier used was an artificial neural network with two hidden layers of size 64 and 16 trained over 20 epochs using a batch size of 64. The same settings were used for all the public datasets and all the real industrial automotive microcontroller datasets. Furthermore, we applied L2 regularization with a weight decay factor of 0.0005 for the classifier training to avoid overfitting and improve the overall performance.

6.4.3 Public Dataset Performance

Table 6.3 provided a detailed comparison of different methods for all the public datasets. As discussed in Section 6.1.2, we use AUROC as a measure of performance

Table 6.3: Comparison of outlier detection performance using AUROC for public datasets.

Datasets	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
thyroid	0.974	0.937	0.989	0.946	0.955	0.962	0.971	0.94
glass	0.683	0.513	0.794	0.624	0.899	0.825	0.767	0.815
satimage-2	0.995	0.981	0.996	0.93	0.995	0.991	0.992	0.985
shuttle	0.994	0.983	0.998	0.998	1	0.995	0.992	0.999
smtp	0.815	0.773	0.921	0.859	0.729	0.86	0.866	0.963
speech	0.515	0.461	0.45	0.482	0.542	0.508	0.505	0.525
Avg. AUROC	0.829	0.775	0.858	0.807	0.853	0.857	0.849	**0.871**

for fair comparison between different methods. Although the public datasets in 6.1 don't have extremely rare outliers compared to the industrial post-silicon testing datasets, they still serve as good basis for general outlier detection method comparison. Among all the 8 methods considered, *RevTransC* gives the best AUROC results on average. Although the other three variants of the proposed methods didn't outperform isolation forest in terms of average AUROC, they are still comparable (within 1%) to it, which is the best reference method, and surpass other three reference methods a lot, demonstrating that the proposed methods can be applied as a robust general-purpose outlier detection method as well.

6.4.4 Industrial Dataset Performance

The performance comparison for the the industrial automotive microcontroller datasets are reported in Table 6.4, 6.5 and 6.6. Each industrial dataset (chip) contains multiple test inserts with a single customer failure, and we learnt a machine learning model for each test insert to screen out the particular customer failure based on each method or setting.

During the advanced outlier detection phase performed at the very end of the testing process, as shown in Fig. 1.2, experienced engineers may determine whether it is worthwhile to discard a certain number of chips, i.e. at a given yield loss level, in order to screen out the customer failure based on outlier detection results collected from all test inserts. As long as the outlier detection results from one of the test inserts can screen out the customer failure, the outlier detection performance based on other test inserts doesn't matter. Therefore, we report the best performance value among all the test inserts for each method in these three tables.

Beyond AUROC introduced in Section 6.1.2, we also used another performance met-

Table 6.4: Comparison of outlier detection performance using AUROC for industrial datasets.

Datasets	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
Chip 1	0.685	0.872	0.882	0.902	0.461	0.889	0.938	0.949
Chip 2	0.823	0.964	0.984	0.979	0.898	0.928	0.939	0.895
Chip 3	0.882	0.94	0.937	0.977	0.928	0.934	0.954	0.973
Chip 4	0.921	0.788	0.838	0.931	0.826	0.828	0.814	0.843
Chip 5	0.972	0.907	0.978	0.959	0.986	0.865	0.966	0.985
Chip 6	0.929	0.738	0.758	0.888	0.916	0.891	0.967	0.999
Chip 1 ("Critical")	0.858	0.753	0.885	0.759	0.839	0.818	0.922	0.647
Chip 2 ("Critical")	0.846	0.858	0.853	0.845	0.917	0.977	0.952	0.925
Chip 3 ("Critical")	0.587	0.874	0.932	0.733	0.992	0.815	0.826	0.986
Chip 4 ("Critical")	0.81	0.77	0.586	0.944	0.778	0.809	0.529	0.762
Chip 5 ("Critical")	0.841	0.99	0.984	0.671	0.942	0.96	0.936	0.941
Chip 6 ("Critical")	0.89	0.861	0.963	0.818	0.935	0.929	0.899	0.896
Avg. AUROC	0.837	0.860	0.882	0.867	0.868	0.887	0.887	**0.900**

Table 6.5: Comparison of outlier detection performance using estimated yield for industrial datasets containing the entire parametric tests.

Datasets	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
Chip 1	0.779	0.917	0.904	0.921	0.543	0.907	0.948	0.961
Chip 2	0.853	0.971	0.987	0.982	0.914	0.944	0.953	0.919
Chip 3	0.886	0.941	0.939	0.978	0.943	0.937	0.96	0.974
Chip 4	0.978	0.914	0.912	0.978	0.932	0.91	0.915	0.933
Chip 5	0.984	0.934	0.99	0.971	0.989	0.894	0.974	0.99
Chip 6	0.947	0.792	0.787	0.916	0.931	0.904	0.974	0.999
Avg. Yield	0.905	0.912	0.920	0.958	0.875	0.916	0.954	**0.963**
# Y≥93%	3	3	3	4	4	2	**5**	**5**

Table 6.6: Comparison of outlier detection performance using estimated yield for industrial datasets containing the "critical" parametric tests.

Datasets ("Critical")	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
Chip 1	0.902	0.795	0.921	0.797	0.893	0.865	0.934	0.728
Chip 2	0.897	0.899	0.881	0.891	0.933	0.982	0.969	0.95
Chip 3	0.632	0.878	0.934	0.743	0.993	0.821	0.832	0.989
Chip 4	0.867	0.922	0.858	0.965	0.888	0.897	0.811	0.915
Chip 5	0.872	0.994	0.99	0.76	0.966	0.965	0.951	0.953
Chip 6	0.919	0.897	0.976	0.852	0.958	0.934	0.925	0.914
Avg. Yield	0.848	0.898	0.927	0.835	**0.939**	0.911	0.904	0.908
# Y$\geq$93%	0	1	3	1	**4**	3	3	3

ric, *estimated yield*, to evaluate different methods for the industrial datasets in Table 6.5 and 6.6. We define the estimated yield of each outlier detection method as one minus the percentage of normal chips we will reject together with the customer failure from the corresponding wafer lot. Recall that the proposed machine learning based advanced outlier detection takes place after the preceding wafer/package/final test steps (Fig. 1.2) in which each test insert will reject a certain number of chips based on functional tests and hard limits in parametric tests. The normal chips that we will reject together with the customer failure by performing the outlier detection at a specific test insert (e.g. wafer test) might have already failed in a subsequent test insert (e.g. final test). Hence, the estimated yield of each outlier detect method is calculated excluding those chips which already failed in subsequent test inserts. As stated before, the final decision of advanced outlier detection process is largely based on engineering experience. The estimated yield, in some sense, reflects the minimum over-reject we can expect in order to remove the particular customer failure in the advanced outlier detection phase.

Note that the adopted industrial datasets only have a single customer failure for each dataset. For each dataset, the reported AUROC and estimated yield value varies for different methods and different datasets, manifesting the real life challenges in extremely-rare defect detection. On average, the reference Gaussian method produces the worst AUROC and estimated yield performance, which mispredicts examples close to the center of the normal data distribution to be abnormal. This suggests that directly characterizing normal data using a Gaussian distribution is not effective in separating out hard latent defects. As for the other reference methods, isolation forest and autoencoder tend to give a relatively good performance among all the reference methods, as shown in Table 6.4 to 6.6.

In general, the four variants of the proposed self-labeling outlier detection method are among the very best of all the experimented methods. Specifically, the variant *RevTransC* reports best performance values in Table 6.4 and 6.5, and the variant *PolyTrans* reports the best estimated yield in Table 6.6. Under this extremely-rare outlier detection context, the proposed self-labeling approach is able to expose the uniqueness of the hard-to-detect outliers and maximize the chance of detecting such extremely-rare (latent) defects (e.g. the targeted customer failure part) as well as minimize the overkill (false positive) rate.

In practice, for cost reason, test engineers may set a minimum yield goal for an outlier detection method to screen out potential customer failures or returns. In Table 6.5 and 6.6, the number of customer failures out of the total six that can be screened out under a yield goal of 93% by each method is reported under the row "# Y≥93%". In other words, "# Y≥93%" counts all the chips with the estimated yield no less than the yield goal (93%) for each method. Given the fact that the customer failures under this study escaped from the original production testing on several millions of parts shipped, robustly screening out even one or two of these customer failures is already challenging. Among all the methods, our proposed self-labeling outlier detection methods screen out the most number of customer failures under the given yield goal. In particular, in Table 6.5 *RevTransB* and *RevTransC* can catch five customer failures out of the total six, and in Table 6.6 *PolyTrans* can catch four customer failures while meeting the minimum 93% yield goal.

As we can observe from Table 6.5 and 6.6, the overall estimated yield and the number of customer failures captured under the given yield goal reduce as we only consider the "critical" parametric tests. This may suggest that manual selection of critical test features based on empirical experiences can lead to sub-optimal outcomes and exploiting powerful machine learning techniques capable of considering a large number of features may be advantageous. Even with less information (parametric tests) about the failure part, the proposed methods still retain a high estimated yield and surpass most of the reference methods as shown in Table 6.6. Specifically, *PolyTrans* gives the best average estimated yield of 93.9% among all the experimented methods.

We compare the four variants of the proposed method based on the results from Table 6.3 to 6.5. The three variants of self-labeling method based on reversible transformations achieve a better average AUROC and estimated yield on average compared to *PolyTrans*, suggesting that maintaining the raw feature information is critical to achieve good outlier detection performance when designing the transformations for the self-labeling method. Furthermore, *RevTransC* achieves the best performance among all three reversible transformation variants, demonstrating that more distinct transformations with higher order and more complex transformations tend to work better. There exists one exception. Table 6.6 reports that *PolyTrans* gives the best yield among all methods while not all available parametric tests are considered. We expect that the the test inserts included in those datasets may not contain all information relevant to the particular customer failure, making the reversible transformations less effective.

6.5 Summary

We presented a machine learning enabled outlier detection methodology in order to facilitate the screening of extremely-rare failures that have escaped from the standard post-silicon testing flow. We proposed a self-labeling technique for unsupervised outlier detection through transformations of available test data, effectively exposing the abnormal behaviour of extremely-rare chip failures in a high-dimensional test feature space. Based on public-domain outlier detection and challenging industrial automotive microcontroller test datasets, we demonstrated through extensive experimental studies that our proposed method consistently outperforms other popular outlier detection algorithms in terms of accuracy and robustness, leading to reduction of yield loss and test escape.

Chapter 7

Semi-supervised Wafer Map Pattern Recognition using Domain-Specific Data Augmentation and Contrastive Learning

This chapter presents a domain-specific application of contrastive learning for wafer pattern recognition. While the existing contrastive learning techniques have primarily focused on conventional image recognition and natural language processing tasks, the unique characteristics of the wafer pattern recognition task presents new challenges and opportunities. First, we investigate the relevance of the transformations proposed in the literature, which act as a mechanism for data augmentation, and identify a near-optimal subset of transformations that are well-suited for meaningful characterization of

©2021 IEEE. Reprinted, with permission, from Hanbin Hu, Chen He, and Peng Li, "Semi-supervised Wafer Map Pattern Recognition using Domain-Specific Data Augmentation and Contrastive Learning", IEEE International Test Conference (ITC), 2021.

similarities and dissimilarities of practical wafer pattern data. Furthermore, we study rotation-based transformations, which are rarely employed in conventional image data analysis. We propose a novel rotation operation that transforms a given wafer pattern into a *similar* pattern by performing non-uniform rotations of the dies on the wafer for which the angle of rotation is a smooth function of the radius. Our proposed new *rotation-twist* transformation acts as a domain-specific data augmentation technique and enables automated generation of high-volumes of similar wafer data while retaining the structure of the original wafer pattern. Experimental results demonstrate that the proposed semi-supervised learning framework greatly improves recognition accuracy compared to traditional supervised methods, and the *rotation-twist* transformation further enhances the recognition accuracy in both semi-supervised and supervised tasks.

7.1 Wafer Map Pattern Recognition

As the crucial step of turning a chip design into a real product, robust semiconductor manufacturing process shall manifest high performance and yield of integrated circuits (ICs). However, the manufacturing process may suffer from different sources of systematic issues, such as stress cracking in chemical-mechanical polishing or contact to gate misalignment caused by a broken photo tool. Therefore, it is required that the manufacturing issues can be efficiently recognized and fixed; otherwise, the yield can be severely reduced and the production process quality is compromised.

One typical approach to detect such systematic manufacturing issues is via wafer map pattern recognition, which observes the die failure pattern on the wafer. Fig. 7.1 gives a few commonly-seen wafer map patterns, where each green pixel indicates a good die which passes all wafer tests, and each yellow pixel indicates a bad die which fails any wafer test. Different wafer map patterns might indicate different systematic manufacturing issues for

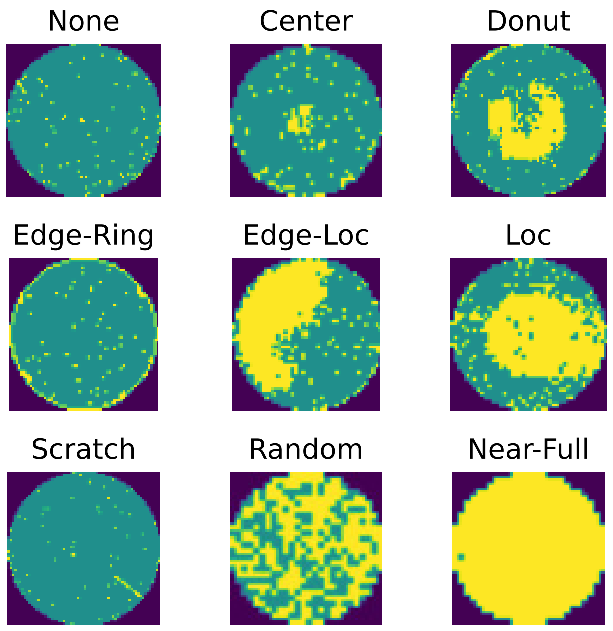


Figure 7.1: Wafer map patterns.

a specific technology node. Certain wafer patterns' implications might be straightforward and technology independent. For instance, the scratch failure pattern may imply some unexpected sharp edges touching and slipping on the surface of the wafer during the manufacturing or wafer test process. Regardless, being able to identify the wafer pattern quickly and automatically is important for manufacturing process issue detection and root cause analysis.

In order to recognize such failures on the fly, several machine learning techniques are investigated to automate the wafer map pattern recognition. [95, 96] applied support vector machine (SVM) to recognize wafer map patterns with preprocessed features using Radon transformation and geometric feature extraction. In recent years, [97, 98, 99, 100] leveraged the strong image recognition power from convolution neural networks to identify wafer map patterns.

However, there are two major challenges which are not yet fully addressed by the previous works. First, it requires huge manual effort to correctly label a large wafer map dataset. If a wafer map is unlabeled, it cannot be utilized in the traditional supervised learning setting, severely reducing the effective number of training samples for learning. On the other hand, the unlabeled data may contain unrecognized patterns, which may provide extra information for wafer pattern learning. [101] proposed an unsupervised approach to recognize wafer map patterns using a set of recognizers trained by generative adversarial networks (GAN); however, the learning process of the multiple recognizers was heavily guided and tuned manually. The second main challenge comes from that fact that wafer map pattern data is typically highly imbalanced. Stable manufacturing processes mostly produce wafer maps that show no patterns at all. The ones with patterns only constitute a small chunk in the entire dataset. Such imbalance may bias a trained recognizer towards making only correct decisions on the dominant wafers without a pattern in the dataset while more important problematic patterns are mis-predicted, causing systematic failure escape. [100, 97] suggested using an auto-encoder (AE) architecture to augment under-represented patterns; however, such augmentation still requires the data label, which may be unrealistic in many cases.

In order to fully use the unlabeled data for wafer map pattern recognition, we proposed a semi-supervised learning framework to first learn a good representation of all wafer map patterns presented in a (potentially large) unlabeled dataset, and then recognize wafer map patterns in a supervised manner with a small labeled dataset. Our methodology has been motivated by the recent development in contrastive learning for unsupervised representation learning in the machine learning community. Contrastive learning is a framework to learn good representation of given data in several applications like image recognition and natural language processing [102, 103, 30]. SimCLR [103] is one contrastive learning technique which achieves comparable or even better performance compared to its supervised counterpart for image recognition tasks demonstrated using the popular ImageNet dataset [104]. The representation is learnt in a self-supervised manner via comparison, i.e., different transformations (views) of the original data are compared to extracted an informative representation. The contrastive learning has been explored in the wafer map pattern detection field in [105], while no novel domain-specific transformations were proposed.

7.2 Semi-Supervised Wafer Map Pattern Recognition

7.2.1 Problem Formulation

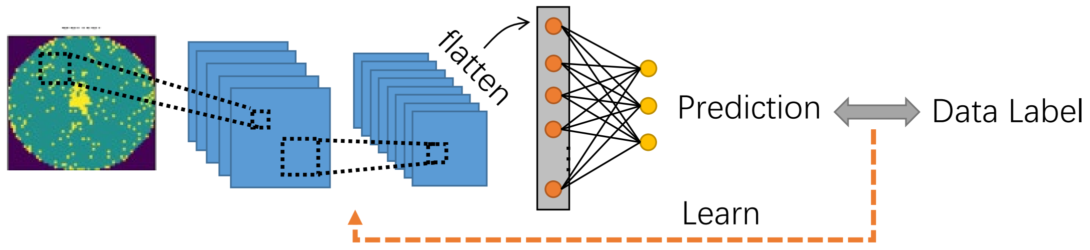
A wafer map of width W and height H can be denoted by $\mathbf{x} \in \mathcal{X} = \{-1, 0, 1\}^{W \times H}$, where 0 is used for good dies, 1 is used for bad dies, and -1 indicates that there is no die in the current location.

Consider a labeled dataset $\mathcal{D}_L = \left\{\cdots, \left(\mathbf{x}^{(l)}, y^{(l)}\right), \cdots\right\}$ with N_L labeled samples with each $\mathbf{x}^{(l)}$ standing for the l-th wafer map and each $y^{(l)} \in \mathcal{Y}$ for its corresponding pattern. In addition, since labeling all wafer maps requires huge manual effort, we consider another unlabeled dataset $\mathcal{D}_U = \left\{\cdots, \mathbf{x}^{(k)}, \cdots\right\}$ with N_U unlabeled samples. Typically, $N_U \gg N_L$.

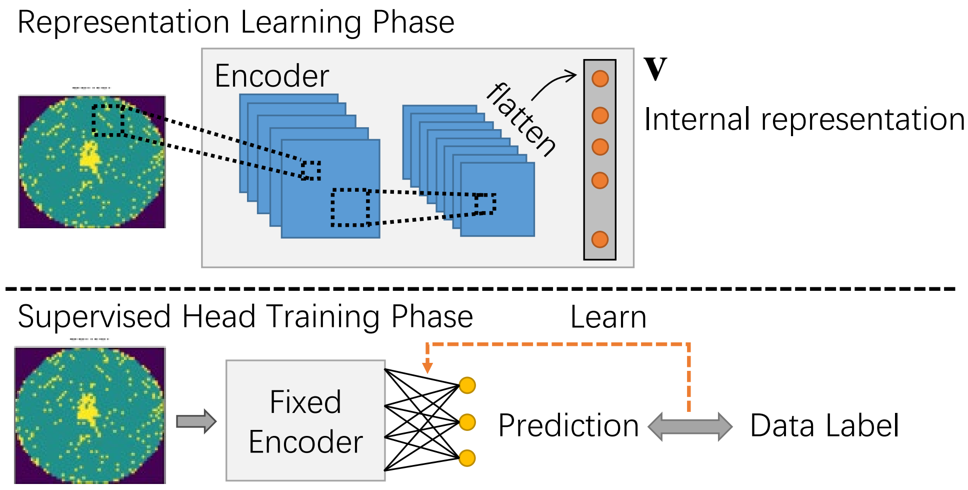
To learn a wafer map recognizer, we optimize a model represented by $f : \mathcal{X} \to \mathcal{Y}$, which is parameterized by $\boldsymbol{\theta}$, to minimize the difference between the recognizer prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$ and the true data label y governed by a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ as follows.

$$\theta^* = \arg\min \mathbb{E}\left[L\left(f(\mathbf{x}; \boldsymbol{\theta}), y\right) \mid \mathcal{D}_L, \mathcal{D}_U\right] \tag{7.1}$$

Note that in supervised learning, the model prediction $\hat{y}$ is directly compared with the true label y via the loss function. Therefore, the unlabeled dataset $\mathcal{D}_U$ cannot be used to compute the loss function, wasting a large number of unlabeled data in the wafer map pattern recognition. If the model is constructed using a neural network with $\boldsymbol{\theta}$ as its model weights, as shown in Fig. 7.2(a), only the comparison between model prediction and data label is used to tune the model weights $\boldsymbol{\theta}$ for the entire neural network, resulting in a huge search space and a potentially over-fit recognizer.



(a) Supervised learning.



(b) Semi-supervised learning.

Figure 7.2: Supervised and semi-supervised learning comparison.

7.2.2 Semi-Supervised Learning

In order to fully utilize the unlabeled dataset $\mathcal{D}_U$, a semi-supervised learning framework is introduced here. Instead of directly learning the recognizer model f, we can rewrite the recognizer as a composition of two functions $f = h_s \circ v$. Here $v : \mathcal{X} \to \mathcal{V}$, parameterized by $\boldsymbol{\theta}_e$, encodes the origin wafer map into an internal representation space $\mathcal{V}$, and $h_s : \mathcal{V} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta}_s$, is a head predicting the data label using the internal representation. Correspondingly, the original model learning procedure is split into 2 phases as follows, which is also shown in Fig. 7.2(b).

$$f\left(\mathbf{x}; \boldsymbol{\theta}\right) = h_s\left(v\left(\mathbf{x}; \boldsymbol{\theta}_e\right); \boldsymbol{\theta}_s\right) \tag{7.2}$$

A good representation $\mathbf{v}$ of the original wafer map is learnt using unlabeled data $\mathcal{D}_U$, of a potentially large size, in the first phase so that it can ease the downstream task of supervised head h_s training. How to learn a good representation from unlabeled data depends on what property the representation is expected to extract and possess. For example, a typical consideration in AE is that the representation should compress all information to reconstruct the original data. In contrastive learning, which is further described in Section 7.3, we would like ensure that representations of similar wafer maps are close to each other. The resulting proxy task to learn good representation is typically different from (7.1), eliminating the needs to use the same loss function requiring labeled data and enabling unsupervised learning for wafer map pattern recognition using $\mathcal{D}_U$.

Note that in the second phase, the encoder parameters $\boldsymbol{\theta}_e$ is fixed, and the labeled data is only used to learn the supervised head model h_s to recognize wafer map patterns. With a fixed $\boldsymbol{\theta}_e$, the parameter space for $\boldsymbol{\theta}$ is greatly reduced, accelerating the optimization process for the supervised head parameters $\boldsymbol{\theta}_s$. In addition, fixing the internal representation avoids disrupting the learnt representation during the learning process of the second phase. If the encoder weights are tuned, it is highly likely that the learnt representation is overwritten with the new information provided by the labeled data, degrading the model into a simple supervised learning model.

7.3 Semi-supervised Representation Learning via Contrastive Learning Framework

7.3.1 Learning Similarity via Data Augmentation

Recently, a novel self-supervised learning framework called contrastive learning suggests extracting a good data representation by learning the similarity among samples [103, 102, 30]. Consider two wafer maps $\mathbf{x}_1$ and $\mathbf{x}_2$. If they are similar (sharing the same wafer map pattern), their internal representations $\mathbf{v}_1$ and $\mathbf{v}_2$ should be "close" to each other.

However, as most data labels are unavailable, the similarity among samples is not revealed explicitly by data labels. Instead, contrastive learning introduces transformations to create similar variants of each given sample and maintain similarities among these variants as part of the unsupervised representation learning. Incorporating these augmented data leads to a larger dataset with more samples. Consider a transformation $T : \mathcal{X} \to \mathcal{X}$ sampled from a transformation set $\mathcal{T}$. The transformations are well selected to make each transformed variant $\tilde{\mathbf{x}}$ similar (sharing the same wafer map pattern) with the original sample $\mathbf{x}$. Therefore, for two transformations T and T', the resulting internal representations

$$\mathbf{v} = v\left(T\left(\mathbf{x}\right); \boldsymbol{\theta}_e\right), \ \mathbf{v}' = v\left(T'\left(\mathbf{x}\right); \boldsymbol{\theta}_e\right) \tag{7.3}$$

should be made close to each other. The "closeness" between internal representations is governed by a *contrastive loss* in an unsupervised manner. For two similar samples, their corresponding internal representations should be close to each other, suggesting a small contrastive loss; otherwise, a large contrastive loss should be generated for dissimilar samples.

7.3.2 Semi-supervised Contrastive Learning Framework

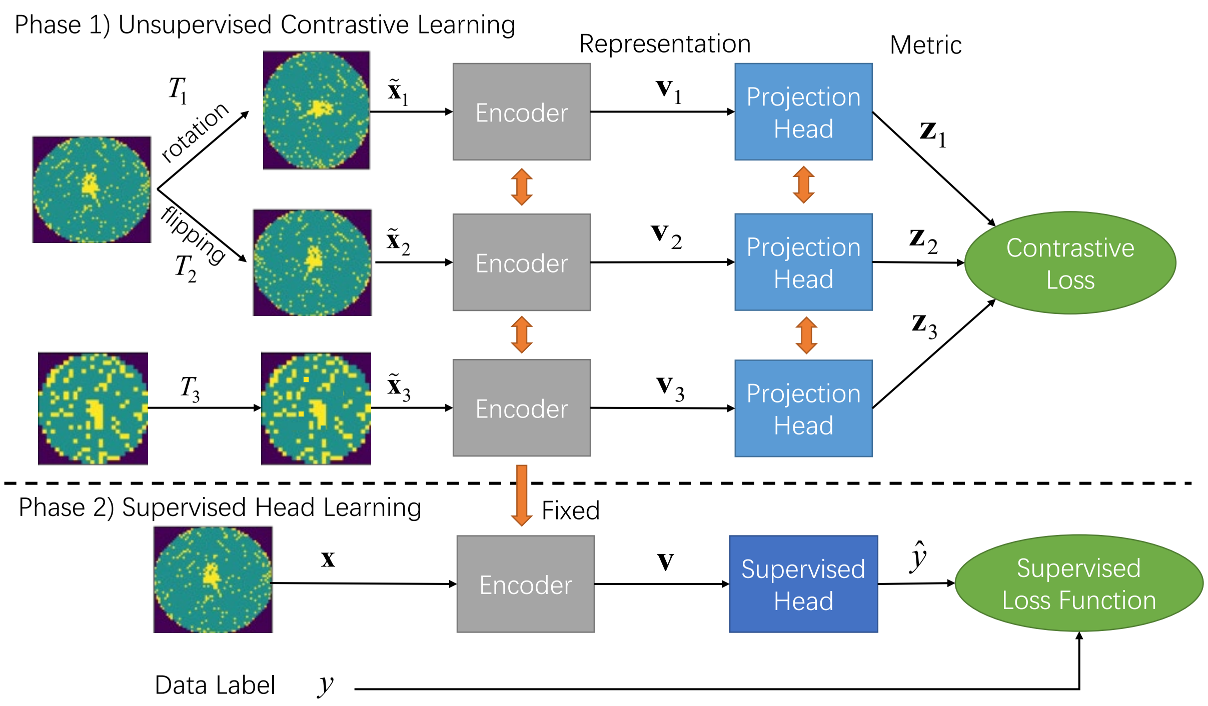


Figure 7.3: Semi-supervised contrastive learning architecture.

Framework Overview

Fig. 7.3 gives a brief illustration of how semi-supervised contrastive learning works. As mentioned in Section 7.2, the semi-supervised learning is conducted in two phases. For the first unsupervised contrastive learning phase, we carefully design a transformation set $\mathcal{T}$ to identify the similarity among samples. Each sample $\mathbf{x} \in \mathcal{X}$ is first transformed to multiple variants $\tilde{\mathbf{x}}$ in the original sample space $\mathcal{X}$. For simiplicity, let's say $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ are transformed through two randomly-selected transformations $T \sim \mathcal{T}$ (e.g., rotation) and $T' \sim \mathcal{T}$ (e.g., flipping); then they go through the same encoder to obtain their internal representations $\mathbf{v}$ and $\mathbf{v}'$ in the *internal representation space* $\mathcal{V}$; finally, the internal representation $\mathbf{v}$ is mapped into a *metric embedding space* $\mathcal{Z}$ through a projection head. The training of contrastive loss, requiring no data label, learns the similarity among samples to formulate a good internal representation via comparing the metric embeddings $\mathbf{z}$ from different variants of multiple samples in the metric embedding space $\mathcal{Z}$.

In the second supervised head training phase, the encoder is inherited and fixed from the previous phase to obtain the trained internal representation $\mathbf{v} \in \mathcal{V}$. An additional supervised head is added to make prediction $\hat{y}$ from $\mathbf{v}$. The loss function is defined by the true data labels over a small labeled dataset to guide the supervised head learning to predict the correct wafer map pattern.

During the inference stage, only the encoder and the supervised head are utilized for recognizing the patterns of the previously-unseen wafer maps, while The projection head is discarded after the first training phase.

Contrastive Learning Objective

Note that the internal representation v is employed in both training phases where two different learning objectives are targeted as shown in Fig. 7.3. We disentangle the unsupervised contrastive learning objective from that of the supervised head learning by introducing the projection head $h_p : \mathcal{V} \to \mathcal{Z}$ that is parameterized by $\boldsymbol{\theta}_p$ and maps the internal representation v to the metric embedding space $\mathcal{Z}$ in which the contrastive loss is defined. This is more beneficial than defining the contrastive loss directly based on the internal representation v. This approach avoids overly constraining the internal representation learning towards the minimization of the contrastive learning loss and allows for more flexible learning of the internal representation to better support the supervised learning task in the second phase.

To this end, the "closeness" in the internal representation space we discussed in Section 7.3.1 shall be interpreted broadly. Two internal representations can be considered "close" as long as their mapped metric embeddings are close enough. Hence, the contrastive loss between two metric embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$ from the same sample within a batch of metric embeddings $\{\mathbf{z}_k\}_{k=1}^{B}$ can be defined as

$$L_{cl}\left(\mathbf{z}_i, \mathbf{z}_j\right) = -\log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_j / \tau\right)}{\sum_{k \in \{i\}'} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_k / \tau\right)}, \tag{7.4}$$

where τ is a temperature coefficient to characterize the relative distance among metric embeddings. Here $\{i\}'$ denotes the set complement $\{1, \cdots, B\} \setminus \{i\}$ excluding element i. To minimize the contrastive loss L_{cl}, the similarity (inner product) between metric embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$ should be as large as possible, while the similarity (inner product) with other embedding $\mathbf{z}_k$ should become small, implying a good representation learnt by the encoder v.

Note that the metric embedding $\mathbf{z}$ is usually normalized with $\mathbf{z}/\left\|\mathbf{z}\right\|$ to avoid scaling issues among different sample views.

7.3.3 Semi-supervised Contrastive Learning Algorithm

With the contrastive learning framework described previously, Algorithm 7 summarizes the detailed procedure of the entire semi-supervised wafer pattern recognition algorithm. Note that for each sample $\mathbf{x}_k$, two parts of contrastive loss $L_{cl}\left(\mathbf{z}_k, \mathbf{z}'_k\right)$ and $L_{cl}\left(\mathbf{z}'_k, \mathbf{z}_k\right)$ are computed, as the two metric embeddings $\mathbf{z}_k$ and $\mathbf{z}'_k$ from the two views $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}'_k$ of the sample sample $\mathbf{x}_k$ are symmetric to each other in terms of contrastive loss computation.

Algorithm 7: Semi-supervised contrastive learning.

```
Input   : Unlabeled dataset $\mathcal{D}_U$; Labeled dataset $\mathcal{D}_L$.
Output: Model prediction parameters $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_s$.
Hyperparameters: Transformation set $\mathcal{T}$, epochs $T_U$, batch size $B_U$, and
  temperature coefficient $\tau$ for unsupervised contrastive learning; Epochs $T_L$ and
  batch size $B_L$ for supervised head learning;
/* unsupervised contrastive learning */
1  for $t \leftarrow 1$ to $T_U$ do
2      for a sampled minibatch $\{\mathbf{x}_k\}_{k=1}^{B_U}$ from $\mathcal{D}_U$ do
3          for $k \leftarrow 1$ to $B_U$ do
4              Sample transformations $T \sim \mathcal{T}$, $T' \sim \mathcal{T}$;
5              $\tilde{\mathbf{x}}_k \leftarrow T(\mathbf{x}_k)$ ; $\tilde{\mathbf{x}}'_k \leftarrow T'(\mathbf{x}_k)$;
6              $\mathbf{v}_k \leftarrow v(\tilde{\mathbf{x}}_k; \boldsymbol{\theta}_e)$ ; $\mathbf{v}'_k \leftarrow v(\tilde{\mathbf{x}}'_k; \boldsymbol{\theta}_e)$;
7              $\mathbf{z}_k \leftarrow h_p(\mathbf{v}_k; \boldsymbol{\theta}_p)$ ; $\mathbf{z}'_k \leftarrow h_p(\mathbf{v}'_k; \boldsymbol{\theta}_p)$;
8              Compute contrastive loss using
                 $l_k \leftarrow -\log \frac{\exp\left(\mathbf{z}_k \cdot \mathbf{z}'_k / \tau\right)}{\sum_{i \in \{k\}'} \exp(\mathbf{z}_k \cdot \mathbf{z}_i / \tau) + \sum_{i=1}^{B_U} \exp\left(\mathbf{z}_k \cdot \mathbf{z}'_i / \tau\right)}$;
9              Compute other contrastive loss using
                 $l'_k \leftarrow -\log \frac{\exp\left(z_k \cdot \mathbf{z}'_k / \tau\right)}{\sum_{i=1}^{B_U} \exp\left(\mathbf{z}'_k \cdot \mathbf{z}_i / \tau\right) + \sum_{i \in \{k\}'} \exp\left(\mathbf{z}'_k \cdot \mathbf{z}'_i / \tau\right)}$;
10         end
11         $\mathcal{L}_U \leftarrow \frac{1}{2B_U} \sum_{k=1}^{B_U} (l_k + l'_k)$;
12         Update $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_p$ to minimize $\mathcal{L}_U$;
13     end
14 end
/* supervised head learning */
15 Fix encoder $\boldsymbol{\theta}_e$ and drop projection head $\boldsymbol{\theta}_p$;
16 for $t \leftarrow 1$ to $T_L$ do
17     for a sampled minibatch $\{\mathbf{x}_k, y_k\}_{k=1}^{B_L}$ from $\mathcal{D}_L$ do
18         for $k \leftarrow 1$ to $B_L$ do
19             $\hat{y}_k \leftarrow h_s(v(\mathbf{x}_k; \boldsymbol{\theta}_e); \boldsymbol{\theta}_s)$;
20             $l_k \leftarrow L(\hat{y}_k, y_k)$;
21         end
22         $\mathcal{L}_L \leftarrow \frac{1}{B_L} \sum_{k=1}^{B_L} l_k$;
23         Update $\boldsymbol{\theta}_s$ to minimize $\mathcal{L}_L$;
24     end
25 end
26 return encoder parameters $\boldsymbol{\theta}_e$ and supervised head parameters $\boldsymbol{\theta}_s$.
```

7.4 Relevance of Existing Contrastive Learning Transformations for Wafer Pattern Recognition

7.4.1 Consideration for Domain Specific Transformations

One important component to determine the performance of the unsupervised contrastive learning is the transformation set $\mathcal{T}$. Transformations included in $\mathcal{T}$ suggest what kind of similarity should be maintained after the transformation, i.e., what similarity among samples should be learnt for a good representation mapped by the encoder. For example, in the image recognition task, if we consider only color distortion transformation without affine transformation in the contrastive learning, the dogs with different colors can be well coded in the internal representation space $\mathcal{V}$, however, the dogs with different sizes and locations in images may be considered distinct in $\mathcal{V}$. We consider three traits for a good transformation to be considered for the representation learning task as follows.

Suitability for the task

Obviously, the transformations under consideration should be suitable for the data under analysis. If the transformation cannot be applied to the data sample space $\mathcal{X}$, it should not be considered.

Capability to maintain similarity among samples

This is the most critical characteristic for a good transformation. The transformation should maintain the major sample property unchanged after transformation. Here, in our application, wafer map patterns should be kept the same, while the detailed good/bad die distribution in a wafer map can be varied by the transformations.

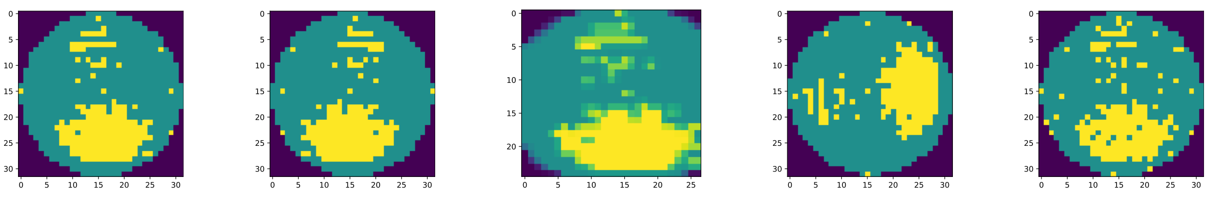
Flexibility to be randomized

Finally, the transformations should be easy to be randomized, adding more flexibility when transforming the details of samples, so that the similarity among samples can be fully captured by the transformations.

7.4.2 Existing Contrastive Learning Transformations

One of the most well-known works in contrastive learning, SimCLR [103], studied a large set of transformations for general image recognition tasks like ImageNet [104]. In particular, four transformations are recommended for standard image recognition tasks: 1) random horizontal flipping, 2) random resized cropping, 3) color jitter, and 4) Gaussian blur.

For the wafer map pattern recognition application, however, not all of the four transformations in SimCLR [103] are suitable. Since a single location x_{ij} in a wafer map may only contain three possible values $\{-1, 0, 1\}$, clearly, the Gaussian blur and the color jitter cannot be applied to the wafer map dataset. On the other hand, we argue that the other two transformations are suitable for the wafer map recognition task. 1) The *random horizontal flipping* augments the unlabeled wafer data by introducing symmetric variants of wafer maps without changing their patterns. 2) The *random resized cropping* crops a random portion of the wafer map and resize the cropped portion back to the original wafer size. It creates new variants of wafer data by extracting a local view of an original wafer map and re-projecting that onto the full wafer.



(a) Original. (b) Flipping. (c) Cropping. (d) Rotation. (e) Noise.

Figure 7.4: Several adopted transformations for wafer map pattern recognition. Note that only random horizontal flipping and random resized cropping were chosen in the SimCLR work [103] for traditional image recognition.

7.5 Proposed Domain-Specific Data Augmentation for Wafer Pattern Recognition

As mentioned in Section 7.4, we adopt *random horizontal flipping* and *random resized cropping* used in traditional image recognition [103] for wafer map pattern recognition, as shown in Fig. 7.4(b) and 7.4(c), respectively. In addition, we propose three domain-specific transformations.

7.5.1 Random Wafer Rotation

Rotations are not suitable for traditional image analysis tasks in which images often maintain a proper orientation, e.g., an upside-down pedestrian is unlikely in a realistic visual scene. However, as illustrated in Fig. 7.4(d), rotating a wafer does not alter its circular shape, but can change the orientation of the good and bad dies while maintaining the same basic wafer map pattern. To generate a variety of rotated wafer maps, we consider two random rotation schemes as follows. The first scheme rotates wafers with a degree randomly selected from a continuous range $[0°, 360°)$, denoted as "Continuous" in Table 7.2. The second scheme rotates wafers with a degree randomly selected from a finite set $\{0°, 90°, 180°, 270°\}$, denoted as "Discrete" in Table 7.2.

7.5.2 Random Die Noise

Adding Gaussian noise to images as a way of transformation was shown to be ineffective for traditional image recognition tasks [103]. Differently, as shown in Fig. 7.4(e), we randomly flip good and bad dies at position (i, j) where a valid die ($x_{ij} \neq -1$) locates as follows.

$$\tilde{x}_{ij} = \begin{cases} 1 - x_{ij} & \text{with a probability of } p_n \\ x_{ij} & \text{with a probability of } 1 - p_n \end{cases} \tag{7.5}$$

With a small p_n value, we largely maintain the present wafer map pattern while introducing a certain degree of perturbation to the wafer. In our experiments, p_n is set to be 0.05.

7.5.3 Rotation-Twist Transformation

Although the previous domain-specific transformations extract good similarity among samples for contrastive learning, these transformations don't change the detailed shape significantly in wafer maps. For example, a scratch pattern cannot be curved no matter which transformation is adopted. With the detailed shape twisted, the relative positioning and the correlation among close dies in wafer maps are modified. Note that the detailed wafer map shapes may vary a lot even for the same wafer map pattern.

Here we propose a novel transformation to twist the detailed shape inside wafer maps while maintaining the pattern for recognition unchanged. Specifically, the proposed transformation performs non-uniform rotations of the dies on the wafer for which the angle of rotation is a smooth function of the radius. As shown in Fig. 7.5, the proposed transformation can transform a straight scratch into a curved scratch. In order to do so, three steps are executed to accomplish the entire twisting. First we randomly generate a smooth angle function to provide certain randomness to the transformation. Then, the wafer map is twisted via rotation based on the generated smooth angle function. Finally, the wafer map is regularized to make it valid. The details of each step are given below, and several examples are provided at the end of the section.

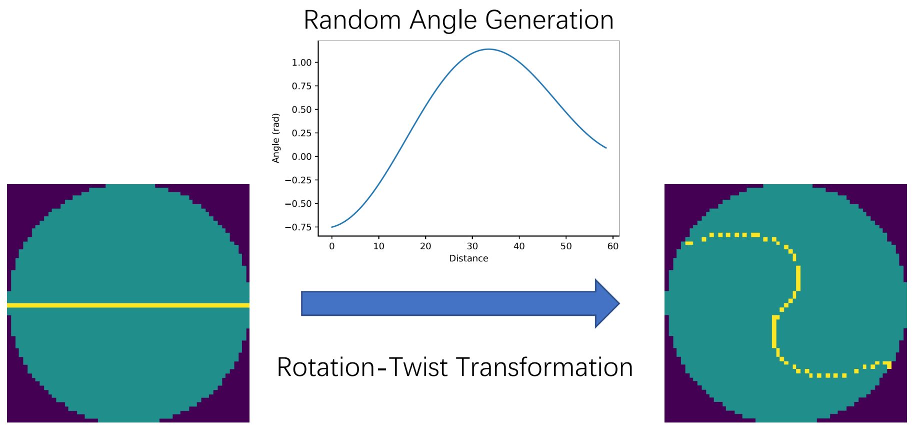
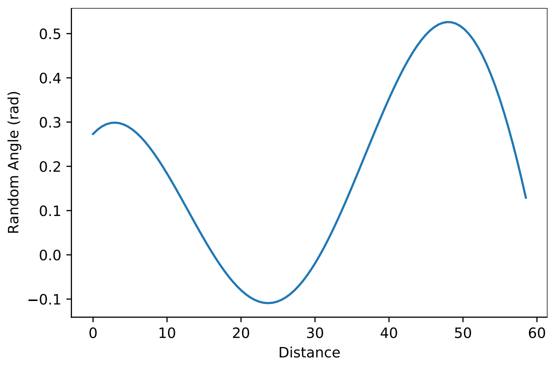
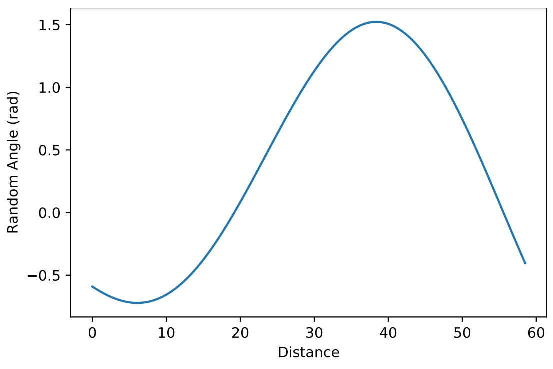


Figure 7.5: Rotation-twist transformation illustration.



(a) Example 1.



(b) Example 2.

Figure 7.6: Random angle generation examples with $m = 3$.

Random Smooth Function Generation

To add a certain degree of randomness to the final wafer map twisting results, a random function is first generated, and then the detailed twist shape is based on the generated random smooth function. To avoid the wafer map pattern altered due to the abrupt function value change, we apply a Fourier-transform-based random function sampling method in [106] to generate a random smooth function as follows.

$$\theta\left(d\right) = a_0 + \sqrt{2}\sum_{j=1}^{m}\left[a_j \cos\frac{2\pi jd}{D} + b_j \sin\frac{2\pi jd}{D}\right], \tag{7.6}$$

where D is the sum of the wafer width and height, and m is the order of the Fourier transform. Each a_j and b_j is an independent Gaussian variable from $\mathcal{N}\left(0, 1/\left(2m+1\right)\right)$. As shown in Fig. 7.6, the 2 examples of generated random function with $m = 3$ is relatively smooth within the radius of the wafer.

Twist via Rotation

After the random angle function is generated, the actual twisting is mainly executed via rotation. All the dies with the same radius to the center of the wafer map are rotated with an angle determined by the generated random angle function and its corresponding radius, ending up with the following equation to relocate the die at (i, j).

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} \cos\theta\left(\sqrt{i^2+j^2}\right) & -\sin\theta\left(\sqrt{i^2+j^2}\right) \\ \sin\theta\left(\sqrt{i^2+j^2}\right) & \cos\theta\left(\sqrt{i^2+j^2}\right) \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \tag{7.7}$$

However, we may notice that the resulting location (i', j') is not guaranteed to be an integer. Here the nearest neighbor is taken by rounding the location to the closest integer grid. Therefore, we can map the goodness of a particular die at (i, j) in the original wafer map $\mathbf{x}$ to $(\lfloor i' \rceil, \lfloor j' \rceil)$ in the twisted wafer map $\tilde{\mathbf{x}}$ as follows.

$$\tilde{x}_{\lfloor i' \rceil \lfloor j' \rceil} = x_{ij} \tag{7.8}$$

Wafer Map Regularization

After twisting the wafer map, a few additional steps are proceeded to make sure the resulting wafer map is a valid one.

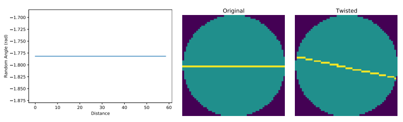
Maintain wafer map shape First, we ensure that the shape of the twisted wafer map matches the original one by guaranteeing that the resulting wafer $\tilde{\mathbf{x}}$ doesn't have die at (i, j) where $x_{ij} = -1$.

Fill in Holes using Majority Votes Secondly, according to (7.7) and (7.8), the resulting wafer map may have a few locations with no corresponding original wafer map locations. For those unfilled positions (holes) in the wafer map, we determine its die goodness via a majority vote of its surrounding 4 dies. The new wafer map is kept updating until no change occurs in two consecutive updates.

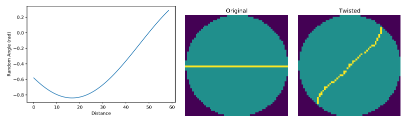
Final check Finally, we set all the remaining unfilled-in locations with good dies. With that, a valid randomly-twisted wafer map $\tilde{x}$ is generated.

Algorithm 8 summarizes the entire procedure for the *rotation-twist* transformation.

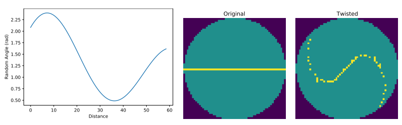
7.5.4 Rotation-Twist Transformation Examples



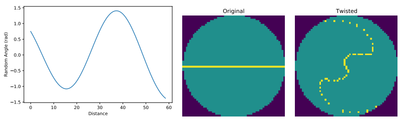
(a) $m = 1$.



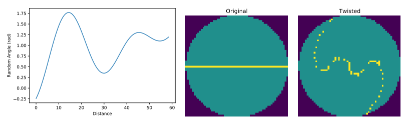
(b) $m = 2$.



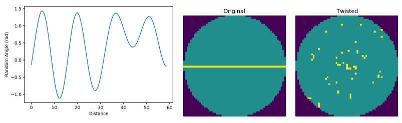
(c) $m = 3$.



(d) $m = 4$.



(e) $m = 5$.

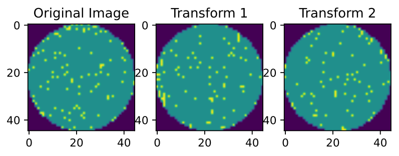


(f) $m = 10$.

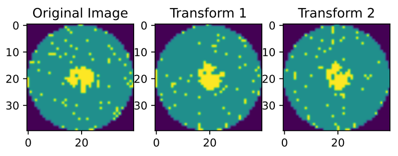
Figure 7.7: Rotation-Twist transformation with different order m.

Algorithm 8: Rotation-Twist transformation flow.

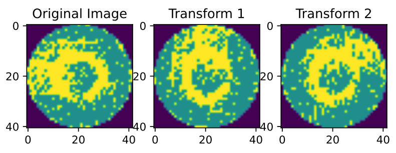
```
   Input   : A wafer map $\mathbf{x}$.
   Output: The transformed wafer map $\tilde{\mathbf{x}}$.
   Hyperparameters: Order $m$ for random smooth angle function generation;
   /* Random smooth function generation */
1  Sample $a_i \sim \mathcal{N}(0, 1/(2m+1))$ for $i = 0, \cdots, m$;
2  Sample $b_i \sim \mathcal{N}(0, 1/(2m+1))$ for $i = 1, \cdots, m$;
3  Construct the angle function $\theta(d)$ using (7.6);
   /* Twist via rotation */
4  Generate a new wafer map $\tilde{\mathbf{x}}$ using (7.7) and (7.8);
   /* Wafer map regularization */
5  Let $\tilde{x}_{ij} = -1$ where $x_{ij} = -1$;
6  do
7      Update undefined die $\tilde{x}_{ij}$ with a majority vote using the surrounding four
         dies $\tilde{x}_{i+1,j}$, $\tilde{x}_{i-1,j}$, $\tilde{x}_{i,j+1}$, and $\tilde{x}_{i,j-1}$;
8  while no update in $\tilde{\mathbf{x}}$ is observed;
9  Let all remaining undefined dies as good dies;
10 return The transformed wafer map $\tilde{\mathbf{x}}$.
```



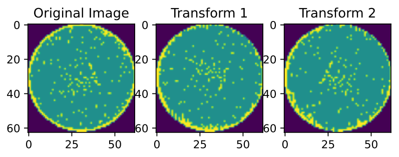
(a) None.



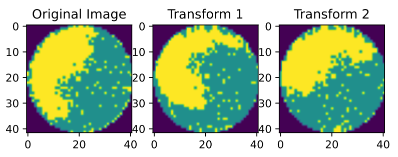
(b) Center.



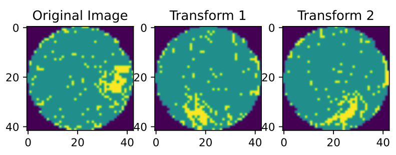
(c) Donut.



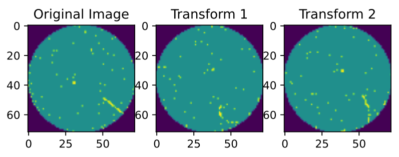
(d) Edge-Ring.



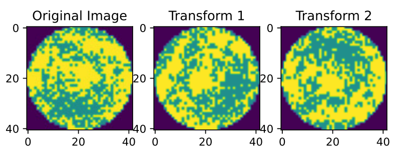
(e) Edge-Loc.



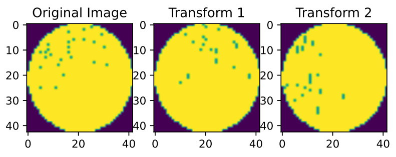
(f) Loc.



(g) Scratch.



(h) Random.



(i) Near-Full.

Figure 7.8: Rotation-Twist transformation examples for all the considered wafer map patterns with $m = 3$.

As the twisting effect of the *rotation-twist* transformation is determined by the generated random smooth angle function, we first check how different orders m affect the resulting twisted wafer maps. Fig. 7.7 gives a few examples of the resulting wafer maps after applying the *rotation-twist* transformation with different orders to the same wafer map with a single line pattern. For each order, the first plot is the generated random angle function, and the rest two images are the original wafer map and the twisted one. When $m = 1$, the proposed transformation is degraded into simple rotation, which provides no additional information to contrastive learning; when $m = 2$, the scratch in the wafer map starts to be twisted, but in a relatively small scale. As the order grows, more twisted detailed shapes can be observed in the wafer maps, which changes the relative positioning of close dies in wafer maps while maintaining the wafer map patterns when $m = 3$ or $m = 4$. However, if we keep increasing the order, we can see the wafer map pattern can be destroyed as higher frequency components are added into random angle function, resulting abrupt function changes like $m = 10$.

Furthermore, Fig. 7.8 gives several wafer map examples after applying the proposed rotation-twist transformation for all the patterns under consideration with an order of $m = 3$. For each pattern in Fig. 7.8, the first image is the original wafer map in the dataset, the second and the third images are two examples of the twisted views. In general, it can be observed that all the wafer map patterns have been kept after the transformation while the detailed shape in each wafer map is significantly changed. For example, in Fig. 7.8(g), although we can treat all the three wafer maps as "Scratch" pattern, the original scratch is perpendicular to the wafer edge, while the twisted scratches are not. In addition, the generated scratches are a bit curved compared to the original one. In addition, the localized die failure patterns are significantly changed after transformation, such as the "Edge-Loc" and "Loc" patterns in Fig. 7.8(e) and 7.8(f). These wafer map examples demonstrate the effectiveness of the proposed transformation to twist wafer maps while maintaining their patterns.

7.6 Experimental Results

7.6.1 Experimental Settings

Methods under Comparison

We compared our proposed semi-supervised contrastive learning method with two sets of baseline methods. The first set of baseline methods mainly adopts a supervised learning methodology, including the support vector machine (SVM) [95], and the convolutional neural network (CNN) architectures, which are widely adopted in [97, 98, 99, 100]. The second baseline method adopts the semi-supervised contrastive learning method with *random horizontal flipping* and *random resized cropping* chosen as transformations. These two transformations are the only transformations used in SimCLR [103] that are suitable for wafer pattern recognition. Hence, the second baseline corresponds to the existing state-of-the-art contrastive learning technique when applied to wafer map data.

We consider two variants of our proposed semi-supervised contrastive learning method. The first variant adopts four adapted transformations: *random horizontal flipping*, *random resized cropping*, *random wafer rotation*, and *random die noise*, which is denoted by "4TCLWMPR" (4 transformations in contrastive learning for wafer map pattern recognition). The second variant includes the *rotation-twist* transformation in addition to the four transformations included in the first variant, denoted by "5TCLWMPR" below.

Dataset and Metric for Comparison

We experimented on a public wafer map pattern dataset: WM-811K [107], containing wafer maps collected from real-world manufacturing process in our experimental studies. As shown in Table 7.1, we used 54,355 labeled wafer maps in the dataset, and split them into a training dataset and a testing dataset with a percentage of 90% and 10%, respectively. In order to perform the proposed semi-supervised contrastive learning framework, all the 48920 wafer maps in the training dataset are used to construct the unlabeled dataset $\mathcal{D}_U$. Only a small portion p_d% of the training dataset is collected to build the labeled dataset $\mathcal{D}_L$, containing both wafer maps and the corresponding patterns, which is applied for both the proposed semi-supervised learning method and the supervised learning methods for comparison. A wide range of labeled data percentage p_d% is experimented as shown in Table 7.3.

Table 7.1: WM-811K dataset statistics.

Wafer map patterns	Training	Testing
None	33051	3679
Center	3113	349
Donut	372	37
Edge-Loc	2150	267
Edge-Ring	7735	819
Loc	1458	162
Random	546	63
Scratch	446	54
Near-full	49	5
Total	48920	5435

Note that the number of samples for each wafer map pattern is highly imbalanced in Table 7.1. For example, the majority pattern, "None" pattern (wafers without any failure pattern), constitutes 67.6% of the whole dataset. Therefore, a traditional accuracy metric doesn't give much information about the wafer map pattern recognizer quality. Even a recognizer predicting all wafer maps as "None" patterns gives an accuracy of 67.6%, which obviously exaggerates the recognizer performance. Instead, we choose a balanced accuracy metric defined as follows.

$$BAC = \frac{\sum_{i=1}^{N} w_i \mathbb{1}\left[\hat{y}_i = y_i\right]}{\sum_{i=1}^{N} w_i}, \tag{7.9}$$

where y_i and $\hat{y}_i$ is the true data label and the prediction result for a wafer map $\mathbf{x}_i$, respectively. The sample weight w_i balances the imbalance among all the pattern types using $w_i = 1/N_{y_i}$, where N_{y_i} is the number of samples with the same pattern y_i in the dataset. Note that here a trivial recognizer (predicting "None" for all inputs) can only give a balanced accuracy of 11.1% (= 1/9), more fairly presenting the recognizer performance. Furthermore, for the wafer map pattern recognition, a higher balanced accuracy indicates the recognizer is more likely to capture systematic manufacturing failure patterns, which is more meaningful in the semiconductor fabrication.

Detailed Hyperparameter Settings

For the SVM, we followed [95] extracting features using Radon-based transform and geometric-based statistics, and trained the SVM recognizer with a radial basis function (RBF) kernel.

For the neural network architecture used in the CNN and the contrasive learning methods, in order to make a fair comparison, the same CNN architecture is used for both the supervised CNN training and the semi-supervised contrastive learning, although different neural network architectures are used in [97, 98, 99, 100]. In particular, the composition of the encoder $v\left(\cdot;\boldsymbol{\theta}_e\right)$ and the supervised head $h_s\left(\cdot;\boldsymbol{\theta}_s\right)$ used in the contrastive learning forms the CNN architecture $f\left(\cdot;\boldsymbol{\theta}_e,\boldsymbol{\theta}_s\right)$ used in a supervised setting. The detailed neural network architectures for the encoder, the projection head, and the supervised head are given in Fig. 7.9. We resize all the wafer maps to a size of 128×128, and use a vector space with a dimensionality of 256 for the internal representation space

$\mathcal{V}$. The entire architectures are implemented using PyTorch v1.8.0 [62] on a workstation with an AMD Ryzen Threadripper 3970X 32-Core processor and an NVIDIA GeForce RTX 3090 GPU.

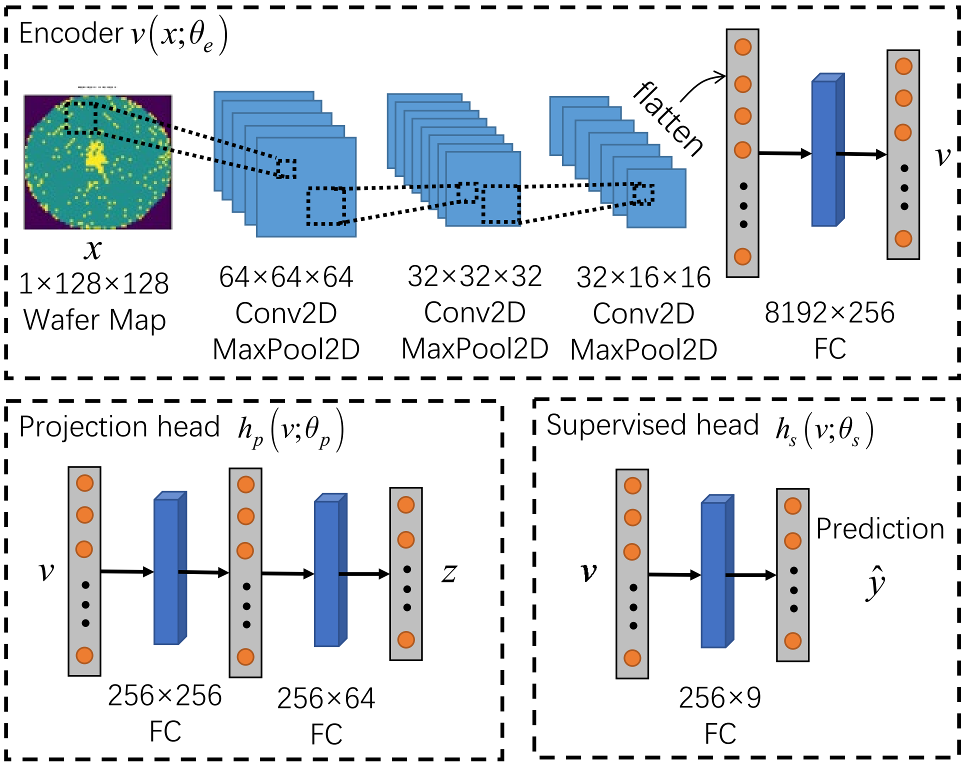


Figure 7.9: Neural networks architectures for semi-supervised contrastive learning framework. Note that the supervised CNN uses the same architecture as $f(x;\theta_e,\theta_s) = h_s(v(x;\theta_e);\theta_s)$.

We experimented two different batch sizes for the semi-supervised contrastive learning $B_U = 256$ and $B_U = 512$ (denoted as "SB" and "LB" in Table 7.3). The training of the encoder uses a maximum epochs of $T_U = 100$ with an early stopping mechanism [108]. The temperature coefficient is set to be 0.1 here. For the supervised head training, another $T_L = 20$ epochs are assigned for finetuning the supervised head with a batch size of $B_L = 64$. Note that in order to alleviate the imbalance issue as shown in Table 7.1, we adopt a weighted batch sampler to balance the occurrence probability of each wafer map pattern during the supervised head training. The same setting is applied to the supervised CNN training with longer epochs $T_L = 100$ to train the entire networks. Moreover, to give more favors to SVM [95] over BAC metric, similar consideration for balancing weighting among different patterns is applied (denoted as “Weighted SVM” in Table 7.3). All the neural network optimization is conducted via an Adam optimizer [109] with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-4}.

7.6.2 Performance Evaluation of Transformations in 4TCLWMPR

In order to justify the usage of the four domain-specific transformations used in 4TCLWMPR, a comprehensive study of the four adopted transformations is performed in Table 7.2. We consider all variants of the possible transformation combinations, with each transformation enabled or not. In total, 24 ($= 2 \times 2 \times 3 \times 2$) possible transformation combinations are considered (2 from random horizontal flip, 2 from random resized cropping, 3 from random wafer rotation, and 2 from random die noise,). Each row in Table 7.2 gives the averaged balanced accuracy of the transformation combinations with the enabled transformation category. For instance, the row of random “Discrete” wafer rotation averages the balanced accuracy from the corresponding 8 transformation combinations.

As shown in Table 7.2, three of selected transformations: random horizontal flipping, random resized cropping, and random die noise, always improve the resulting prediction accuracy for all the labeled data percentages considered. For the random wafer rotation, although the “Discrete” variant’s performance is slightly lower than the ”Continuous” one for a labeled data percentage of 20%, a relatively large drop 0.46% of ”Continuous” variant can be observed at a labeled data percentage of 10% compared to disabling wafer rotation. With that, we include the “Discrete” version of random wafer rotation in the final selected transformations.

Table 7.2: Averaged balanced accuracy performance from 24 variants of different transformation combinations in 4TCLWMPR.

Category	Enabled	Labeled data percentage		
		5%	10%	20%
Flip	No	80.71%	80.61%	82.53%
	Yes	**80.90%**	**81.42%**	**83.02%**
Crop	No	80.02%	79.93%	80.99%
	Yes	**81.59%**	**82.09%**	**84.56%**
Rotation	No	80.60%	81.05%	82.17%
	Continuous	80.68%	80.59%	**83.09%**
	Discrete	**81.14%**	**81.40%**	83.07%
Noise	No	80.26%	80.24%	81.73%
	Yes	**81.35%**	**81.79%**	**83.82%**

7.6.3 Semi-supervised Learning Performance

The wafer map pattern recognition performance over the WM-811K dataset for the semi-supervised learning method is given in Table 7.3. The first three rows are the balanced accuracy metric for the three supervised methods: SVM [95], Weighted SVM, and the CNN architecture. The next 6 rows are the performance for the baseline SimCLR and the proposed semi-supervised contrastive learning variants with different transformation combinations and batch sizes. As we can see from the table, the recognition balanced accuracy for the supervised methods is relatively low. For the traditional SVM classifier in [95], the recognition balanced accuracy is even lower than 50%. Even with balanced sample weights to train the SVM, the recognizer can be hardly to be used in a real application to identify wafer map failure patterns, although the averaged balanced accuracy of weighted SVM gets improved by 4.56% compared to the vanilla SVM. With the help of modern neural network architectures like CNN, the recognition power gets greatly

Table 7.3: Balanced accuracy comparison for WM-811K between supervised learning and semi-supervised learning methods.

Type	Method	Labeled data percentage							Avg.
		1%	5%	8%	10%	20%	30%	50%	
Supervised	SVM	47.27%	41.57%	47.79%	45.88%	48.66%	N/A	46.99%	46.36%
	Weighted SVM	48.07%	46.99%	51.70%	54.72%	55.40%	N/A	48.76%	50.94%
	CNN	64.87%	66.10%	71.41%	66.74%	77.90%	74.62%	74.20%	70.83%
CL (SB)	SimCLR	71.20%	71.11%	71.15%	72.81%	81.13%	79.22%	82.42%	75.58%
	Proposed 4TCLWMPR	**74.64%**	70.16%	73.38%	73.18%	**83.05%**	79.94%	83.32%	76.81%
	Proposed 5TCLWMPR	72.81%	**74.65%**	**76.48%**	**75.48%**	79.75%	**80.26%**	**83.47%**	**77.56%**
CL (LB)	SimCLR	72.35%	72.05%	75.24%	70.39%	78.30%	80.10%	79.91%	75.48%
	Proposed 4TCLWMPR	72.42%	73.90%	76.34%	**75.08%**	80.23%	81.75%	80.21%	77.13%
	Proposed 5TCLWMPR	**73.27%**	**75.54%**	**77.52%**	73.70%	**82.19%**	**82.39%**	**80.72%**	**77.90%**

enhanced by another increase of 19.89% in terms of the average balanced accuracy.

Thanks to the good representation learnt by the contrastive learning using an unlabeled dataset, the proposed semi-supervised learning framework significantly improves the recognition performance with a boost around 7% for the balanced accuracy on average. For instance, we can observe a recognition accuracy boost of 9.77% using the proposed contrastive learning with the selected transformations, compared to CNN, even for a small labeled data percentage like 1%. These significant wafer map pattern recognition performance improvements can be attributed to two aspects: 1) Good representation for wafer maps is well extracted by comparing different transformations of wafer maps to learn the similarity among samples; 2) The unlabeled data is fully utilized, which cannot be learnt by supervised methods, greatly reducing the manual labor to label all the wafer map patterns.

The effect of different transformation combinations is also studied for the semi-supervised contrastive learning framework. It is observed from Table 7.3 that there still exist certain performance gaps between different transformation combinations, although even the baseline SimCLR transformations' performance already surpasses the supervised CNN a lot. With the domain-specific transformations (4TCLWMPR), the average balanced accuracy over all labeled data percentages get improved by 1.23% for small batch size and 1.65% for large batch size. Another 0.76% boost can be observed when applying the proposed *rotation-twist* transformations (5TCLWMPR). With the additional proposed *rotation-twist* transformation (5TCLWMPR), there are 5 cases outperforming all other methods using a small batch size for all the 7 experimented labeled data percentages, and 6 out of 7 using a large batch size. Such domain-specific transformations suggest the similarity among wafer maps are well captured by these selected transformations, enhancing the representation learnability for the wafer map pattern recognition application. The proposed *rotation-twist* transformation further refines the representation learning by identifying similar samples with distinct detailed failure pattern shapes.

As suggested in [103], a large batch size is beneficial for the contrastive learning, as more dissimilar views of samples can be used in (7.4) to identify the dissimilarity among samples. As shown in Table 7.3, with a large batch size, the balanced recognition accuracy can be slightly improved on average, however, we still suggest a case-by-case study for the batch size to be used in contrastive learning when applying different labeled data percentages.

Note that we also experimented another variant of contrastive learning which also finetunes the encoder during the second phase of the semi-supervised learning. However, a similar performance is observed as the one for supervised CNN, implying that the finetuned encoder forgets the representation learnt during the unsupervised contrastive learning and degrades to a simple supervised learning.

7.6.4 Rotation-Twist as Data Augmentation

Table 7.4: Performance boost of *rotation-twist* transformation as data augmentation in a supervised learning setting.

Labeled data percentage	CNN	CNN+RoTwist	Improvement
1%	64.87%	68.19%	+3.32%
5%	66.10%	73.25%	+7.15%
8%	71.41%	73.10%	+1.69%
10%	66.74%	73.60%	+6.86%
20%	77.90%	81.39%	+3.49%
30%	74.62%	80.62%	+6.00%
50%	74.20%	78.06%	+3.86%
Avg.	70.83%	75.46%	+4.62%

To boost the recognition power for a supervised learning model, one common strategy is to augment a small dataset by adding more samples transformed from the original dataset with the corresponding data labels. To validate the effectiveness of our proposed *rotation-twist* transformation, we experiment the same CNN architecture, and train them in a supervised manner with each sample randomly transformed by the proposed *rotation-twist* transformation with an order of $m = 3$. Table 7.4 gives the resulting balanced accuracy performance with *rotation-twist* data augmentation at different labeled data percentages. Compared to the vanilla CNN, a great accuracy improvement can be observed with 4.62% on average, indicating the effectiveness of the proposed transformation, which well extracts the similarity among wafer maps.

Compared to the semi-supervised contrastive learning results as shown in Table 7.3, the proposed contrastive learning recognition still outperforms data-augmented supervised CNN for all the experimented labeled data percentages, demonstrating that the proposed semi-supervised contrastive learning can efficiently learn good representations for wafer maps to build a robust wafer map pattern recognizer.

7.7 Summary

In this chapter, we proposed a semi-supervised contrastive learning framework for wafer map pattern recognition. Contrastive learning is adopted to learn good representation in an unsupervised manner via comparing different views of wafer maps generated by a set of selected domain-specific transformations. In addition, a novel *rotation-twist* transformation is proposed to change the detailed shape of wafer maps while maintaining the original patterns. Our experimental results demonstrate the effectiveness of the semi-supervised contrastive learning over the supervised learning methods, and present the performance boost for the proposed domain-specific transformations.

Chapter 8

Conclusion

With verification and testing becoming key bottlenecks in the product development cycle, this dissertation focuses on the rare failure detection in a high-dimensional parameter space using minimal expensive simulation/measurement data.

On the verification side, this dissertation proposes to place machine learning models, mimicking the circuit behavior, under verification, which significantly relaxes the simulation/measurement requirements and improves the verification efficiency.

First, we present a new direction in AMS verification by proposing a hybrid formal/machine-learning verification technique (HFMV) to combine the best of the two worlds. HFMV adds formalism on the top of a probabilistic learning model while providing a sense of coverage for extremely rare failure detection. HFMV intelligently and iteratively reduces uncertainty of the learning model by a proposed formally-guided active learning strategy and discovers potential rare failure regions in complex high-dimensional parameter spaces. It leads to reliable failure prediction in the case of a failing circuit, or a high-confidence pass decision in the case of a good circuit. We demonstrate that HFMV is able to employ a modest amount of data to identify hard-to-find rare failures which are completely missed by state-of-the-art sampling methods even with high volume sampling data.

Later, we present a Bayesian optimization (BO) based approach to the challenging problem of verifying AMS circuits with stringent low failure requirements. At the heart of the proposed BO process is a delicate balancing between two competing needs: exploitation of the current statistical model for quick identification of highly-likely failures and exploration of undiscovered design space so as to detect hard-to-find failures within a large parametric space. To do so, we simultaneously leverage multiple optimized acquisition functions to explore varying degrees of balancing between exploitation and exploration. This makes it possible to not only detect rare failures which other techniques fail to identify, but also do so with significantly improved efficiency. We further build in a mechanism into the BO process to enable detection of multiple failure regions, hence providing a higher degree of coverage. Moreover, the proposed approach is readily parallelizable, further speeding up failure detection, particularly for large circuits for which acquisition of simulation/measurement data is very time-consuming. Our experimental study demonstrates that the proposed approach is very effective in finding very rare failures and multiple failure regions which existing statistical sampling techniques and other BO techniques can miss, thereby providing a more robust and cost-effective methodology for rare failure detection.

Furthermore, this dissertation proposes BO frameworks under high dimensional space to further improve the verification efficiency. Two techniques are explored here. On one hand, we utilize random embedding to effectively reduce the dimensionality of a given verification problem in a linear manner to improve both the quality of BO-based optimal sampling and computational efficiency. On the other hand, we combine a reversible network and a gating architecture to identify essential features from datasets and reduce feature dimension for fast failure detection. While reversible residual networks (RevNets) have been actively studied for its restoration ability from output to input without the loss of information, the gating network facilitates the RevNet to aim at effective dimension reduction. We incorporate the proposed reversible gating architecture into Bayesian optimization (BO) framework to reduce the dimensionality of BO embedding important features clarified by gating fusion weights so that the failure points can be efficiently located. Furthermore, we propose a conditional density estimation of important and non-important features to extract high-dimensional original input features from the low-dimension important features, improving the efficiency of the proposed methods.

On the subject of AMS testing, this dissertation proposes to utilize self-supervised learning methods to detect extremely rare customer failure, which addresses the lack of labels.

In the first effort of this direction, we propose to train a more robust unsupervised learning model by self-labeling the training data via a set of transformations. Using the labeled data we train a multi-class classifier through supervised training. The goodness of the multi-class classification decisions with respect to an unseen input data is used as a normality score to defect anomalies. Furthermore, we propose to use reversible information lossless transformations to retain the data information and boost the performance and robustness of the proposed self-labeling approach.

Finally, this dissertation suggests a contrastive learning framework for semi-supervised learning and prediction of wafer map patterns. Our framework incorporates an encoder to learn good representation for wafer maps in an unsupervised manner, and a supervised head to recognize wafer map patterns. In particular, contrastive learning is applied for the unsupervised encoder representation learning supported by augmented data generated by different transformations (views) of wafer maps. We identified a set of transformations to effectively generate similar variants of each original pattern. We further proposed a novel *rotation-twist* transformation to augment wafer map data by rotating each given wafer map for which the angle of rotation is a smooth function of the radius. Experimental results demonstrate that the proposed semi-supervised learning framework greatly improves recognition accuracy compared to traditional supervised methods, and the rotation-twist transformation further enhances the recognition accuracy in both semi-supervised and supervised tasks.

Bibliography

[1] S. Kondo, H. Kubota, H. Katagiri, Y. Ota, M. Hirono, T. T. Ta, H. Okuni, S. Ohtsuka, Y. Ojima, T. Sugimoto, H. Ishii, K. Yoshioka, K. Kimura, A. Sai, and N. Matsumoto, *An automotive lidar soc for 240 × 192-pixel 225-m-range imaging with a 40-channel 0.0036-mm¡sup¿2¡/sup¿ voltage/time dual-data-converter-based afe*, *IEEE Journal of Solid-State Circuits* **55** (2020), no. 11 2866–2877.

[2] E. Noorsal, K. Sooksood, H. Xu, R. Hornig, J. Becker, and M. Ortmanns, *A neural stimulator frontend with high-voltage compliance and programmable pulse shape for epiretinal implants*, *IEEE Journal of Solid-State Circuits* **47** (2012), no. 1 244–256.

[3] S. Carreon-Bautista, L. Huang, and E. Sanchez-Sinencio, *An autonomous energy harvesting power management unit with digital regulation for iot applications*, *IEEE Journal of Solid-State Circuits* **51** (2016), no. 6 1457–1474.

[4] *Ieee standard for systemverilog–unified hardware design, specification, and verification language*, *IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012)* (2018) 1–1315.

[5] *Ieee standard for universal verification methodology language reference manual*, *IEEE Std 1800.2-2020 (Revision of IEEE Std 1800.2-2017)* (2020) 1–458.

[6] T. Marchok, A. El-Maleh, W. Maly, and J. Rajski, *Complexity of sequential atpg*, in *Proceedings the European Design and Test Conference. ED TC 1995*, pp. 252–261, 1995.

[7] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, *Automatic test packet generation*, in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, p. 241–252, 2012.

[8] H. Chang and K. Kundert, *Verification of complex analog and rf ic designs*, *Proceedings of the IEEE* **95** (March, 2007) 622–639.

[9] M. Miller and F. Brewer, *Formal verification of analog circuit parameters across variation utilizing sat*, in *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1442–1447, March, 2013.

[10] L. Yin, Y. Deng, and P. Li, *Verifying dynamic properties of nonlinear mixed-signal circuits via efficient smt-based techniques*, in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 436–442, November, 2012.

[11] W. Denman, B. Akbarpour, S. Tahar, M. H. Zaki, and L. C. Paulson, *Formal verification of analog designs using metitarski*, in *Formal Methods in Computer-Aided Design (FMCAD)*, pp. 93–100, Nov, 2009.

[12] S. Sun, X. Li, H. Liu, K. Luo, and B. Gu, *Fast statistical analysis of rare circuit failure events via scaled-sigma sampling for high-dimensional variation space*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **34** (July, 2015) 1096–1109.

[13] S. Sun, X. Li, H. Liu, K. Luo, and B. Gu, *Fast statistical analysis of rare circuit failure events via scaled-sigma sampling for high-dimensional variation space*, in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 478–485, Nov, 2013.

[14] A. Singhee and R. A. Rutenbar, *Statistical blockade: Very fast statistical simulation and modeling of rare circuit events and its application to memory design*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **28** (Aug, 2009) 1176–1189.

[15] W. R. Mann, F. L. Taber, P. W. Seitzer, and J. J. Broz, *The leading edge of production wafer probe test technology*, in *Proceedings of International Conference on Test*, pp. 1168–1195, Oct, 2004.

[16] S. S. Sabade and D. M. Walker, *Ic outlier identification using multiple test metrics*, *IEEE Design & Test of Computers* **22** (Nov, 2005) 586–595.

[17] S. Illyes and D. A. G. Baglee, *Statistical bin limits-an approach to wafer disposition in ic fabrication*, *IEEE Transactions on Semiconductor Manufacturing* **5** (Feb, 1992) 59–61.

[18] M. J. Moreno Lizaranzu, *Computer Integrated Manufacturing in Semiconductor Industry. Automation, Electronic Wafer Mapping, Defect Reduction and Equipment Utilization Improvement in Probe and Final Test.* PhD thesis, University of Seville, November, 2012.

[19] *Guidelines for part average testing*, *Automotive Electronic Council* **AEC-Q001 Rev-D** (Dec., 2011).

[20] T. Sakamoto, K. Yofu, and T. Kyuho, *New method of screening out outlier; expanded part average testing during package level test*, *IEEE Transactions on Semiconductor Manufacturing* **30** (Nov, 2017) 351–356.

[21] W. Dobbelaere, R. Vanhooren, W. De Man, K. Matthijs, A. Coyette, B. Esen, and G. Gielen, *Analog fault coverage improvement using final-test dynamic part average testing*, in *Proceedings of IEEE International Test Conference (ITC)*, pp. 1–9, Nov, 2016.

[22] M. J. Moreno-Lizaranzu and F. Cuesta, *Improving electronic sensor reliability by robust outlier screening*, *Sensors (Basel, Switzerland)* **13** (2013), no. 10 13521–13542.

[23] K. M. Butler, S. Subramaniam, A. Nahar, J. M. Carulli, T. J. Anderson, and W. R. Daasch, *Successful development and implementation of statistical outlier techniques on 90nm and 65nm process driver devices*, in *IEEE Intl. Reliability Physics Symposium*, pp. 552–559, 2006.

[24] W. R. Daasch, J. McNames, D. Bockelman, and K. Cota, *Variance reduction using wafer patterns in i/sub ddq/ data*, in *International Test Conference*, pp. 189–198, 2000.

[25] S. S. Sabade and D. M. H. Walker, *Comparison of wafer-level spatial i/sub ddq/ estimation methods: Nnr versus ncr*, in *IEEE Intl. Workshop on Current and Defect Based Testing*, pp. 17–22, Apr., 2004.

[26] W. R. Daasch, K. Cota, J. McNames, and R. Madge, *Neighbor selection for variance reduction in iddq and other parametric data*, in *IEEE International Test Conference*, (USA), pp. 92–100, 2001.

[27] P. M. O'Neill, *Production multivariate outlier detection using principal components*, in *IEEE International Test Conference*, pp. 1–10, 2008.

[28] N. Sumikawa, J. Tikkanen, L. Wang, L. Winemberg, and M. S. Abadir, *Screening customer returns with multivariate test analysis*, in *IEEE International Test Conference*, pp. 1–10, 2012.

[29] I. Golan and R. El-Yaniv, *Deep Anomaly Detection Using Geometric Transformations*, in *Proceedings of Advances in Neural Information Processing Systems*, pp. 9758–9769, 2018.

[30] P. H. Le-Khac, G. Healy, and A. F. Smeaton, *Contrastive Representation Learning: A Framework and Review*, *IEEE Access* **8** (2020) 193907–193934.

[31] K. Scheibler, F. Neubauer, A. Mahdi, M. Fränzle, T. Teige, T. Bienmüller, D. Fehrer, and B. Becker, *Accurate icp-based floating-point reasoning*, in *Formal Methods in Computer-Aided Design (FMCAD)*, pp. 177–184, Oct, 2016.

[32] L. D. Moura and N. Bjørner, *Z3: An efficient smt solver*, in *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 337–340, April, 2008.

[33] H. A. Chipman, E. I. George, and R. E. Mcculloch, *Bart: Bayesian additive regression trees*, *Annals of Applied Statistics* **4** (2010), no. 1 266–298.

[34] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning.* MIT Press, 2006.

[35] M. E. Tipping, *Sparse bayesian learning and the relevance vector machine*, *Journal of Machine Learning Research* **1** (June, 2001) 211–244.

[36] H. Lin and P. Li, *Relevance vector and feature machine for statistical analog circuit characterization and built-in self-test optimization*, in *ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, June, 2016.

[37] H. Lin, A. M. Khan, and P. Li, *Statistical circuit performance dependency analysis via sparse relevance kernel machine*, in *IEEE International Conference on IC Design and Technology (ICICDT)*, pp. 1–4, May, 2017.

[38] L. D. Moura, B. Dutertre, and N. Shankar, *A tutorial on satisfiability modulo theories*, in *Proceedings of the International Conference on Computer Aided Verification (CAV)*, pp. 20–36, July, 2007.

[39] S. Lai and P. Li, *A fully on-chip area-efficient cmos low-dropout regulator with fast load regulation*, *Analog Integrated Circuits and Signal Processing* **72** (August, 2012) 433–450.

[40] Y. Wang, P. Li, and S. Lai, *A unifying and robust method for efficient envelope-following simulation of pwm/pfm dc-dc converters*, in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 618–625, November, 2014.

[41] J. Snoek, H. Larochelle, and R. P. Adams, *Practical bayesian optimization of machine learning algorithms*, in *International Conference on Neural Information Processing Systems (NIPS)*, pp. 2951–2959, Dec, 2012.

[42] K. Swersky, J. Snoek, and R. P. Adams, *Multi-task bayesian optimization*, in *Advances in Neural Information Processing Systems 26*, pp. 2004–2012. Curran Associates, Inc., 2013.

[43] E. Brochu, V. M. Cora, and N. D. Freitas, *A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning*, *arXiv preprint arXiv:1012.2599* (Dec, 2010).

[44] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, *Taking the human out of the loop: A review of bayesian optimization*, *Proceedings of the IEEE* **104** (Jan, 2016) 148–175.

[45] B. Reagen, J. M. Hernández-Lobato, R. Adolf, M. Gelbart, P. Whatmough, G.-Y. Wei, and D. Brooks, *A case for efficient accelerator design space exploration via bayesian optimization*, in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pp. 1–6, July, 2017.

[46] W. Lyu, P. Xue, F. Yang, C. Yan, Z. Hong, X. Zeng, and D. Zhou, *An efficient bayesian optimization approach for automated optimization of analog circuits*, *IEEE Transactions on Circuits and Systems I: Regular Papers* (2017) 1–14.

[47] D. R. Jones, *A taxonomy of global optimization methods based on response surfaces*, *Journal of Global Optimization* **21** (Dec, 2001) 345–383.

[48] R. Martinez-Cantin, *Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits*, *Journal of Machine Learning Research* **15** (2014) 3915–3919.

[49] J. M. Gablonsky and C. T. Kelley, *A locally-biased form of the direct algorithm*, *Journal of Global Optimization* **21** (Sep, 2001) 27–37.

[50] M. J. D. Powell, *The bobyqa algorithm for bound constrained optimization without derivatives*, *Cambridge NA Report* (2009) 26–46.

[51] S. G. Johnson, *The nlopt nonlinear-optimization package [software]*, 2014.

[52] M. R. Hoque and S. S. Ang, *A cmos under-voltage lockout circuit*, in *World Congress on Engineering and Computer Science (WCECS)*, pp. 1–4, Oct, 2008.

[53] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, *Gaussian process optimization in the bandit setting: No regret and experimental design*, in *Proceedings of the International Conference on International Conference on Machine Learning (ICML)*, pp. 1015–1022, June, 2010.

[54] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. D. Freitas, *Bayesian optimization in high dimensions via random embeddings*, in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1778–1784, August, 2013.

[55] M. J. D. Powell, *Direct search algorithms for optimization calculations*, *Acta Numerica* **7** (1998) 287–336.

[56] M. S. Shim, C. Zhao, Y. Li, X. Zhang, and P. Li, *Robust deep multi-modal sensor fusion using fusion weight regularization and target learning*, *CoRR* **abs/1901.10610** (2019).

[57] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, *Reversible architectures for arbitrarily deep residual neural networks*, *arXiv preprint arXiv:1709.03698* (2017).

[58] E. Haber and L. Ruthotto, *Stable architectures for deep neural networks*, *Inverse Problems* **34** (2017), no. 1 014004.

[59] G. Zhang, J. Zhang, and J. Hinkle, *Learning nonlinear level sets for dimensionality reduction in function approximation*, in *Advances in Neural Information Processing Systems*, pp. 13220–13229, 2019.

[60] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, *i-revnet: Deep invertible networks*, *arXiv preprint arXiv:1802.07088* (2018).

[61] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, *The reversible residual network: Backpropagation without storing activations*, in *Advances in neural information processing systems*, pp. 2214–2224, 2017.

[62] 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*, pp. 8024–8035, 2019.

[63] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, *Deep learning in the automotive industry: applications and tools*, in *IEEE International Conference on Big Data*, 2016.

[64] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy, *A review of deep learning methods and applications for unmanned aerial vehicles*, *Journal of Sensors* (2017).

[65] I. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, in *International Conference on Learning Representations*, 2015.

[66] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, *Analysis of universal adversarial perturbations*, *arXiv preprint arXiv:1705.09554* (2017).

[67] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, *The limitations of deep learning in adversarial settings*, in *IEEE European Symposium on Security and Privacy*, 2016.

[68] N. Carlini and D. Wagner, *Towards evaluating the robustness of neural networks*, *arXiv preprint arXiv:1608.04644* (2016).

[69] A. Kurakin, I. Goodfellow, and S. Bengio, *Adversarial machine learning at scale*, *arXiv preprint arXiv:1611.01236* (2016).

[70] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, in *International Conference on Learning Representations*, 2018.

[71] T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, and L. Daniel, *Evaluating the robustness of neural networks: An extreme value theory approach*, in *International Conference on Learning Representations*, 2018.

[72] Y. Tsuzuku, I. Sato, and M. Sugiyama, *Lipschitz-margin training: scalable certification of perturbation invariance for deep neural networks*, in *International Conference on Neural Information Processing Systems*, 2018.

[73] I. J. Goodfellow, *Gradient masking causes CLEVER to overestimate adversarial perturbation size*, *arXiv preprint arXiv:1804.07870* (2018).

[74] T. Huster, C. J. Chiang, and R. Chadha, *Limitations of the lipschitz constant as a defense against adversarial examples*, *arXiv preprint arXiv:1807.09705* (2018).

[75] Y. Song, R. Shu, N. Kushman, and S. Ermon, *Constructing unrestricted adversarial examples with generative models*, in *International Conference on Neural Information Processing Systems*, 2018.

[76] I. Dunn, T. Melham, and D. Kroening, *Generating realistic unrestricted adversarial inputs using dual-objective GAN training*, *arXiv preprint arXiv:1905.02463* (2019).

[77] M. I. Gomes and A. Guillou, *Extreme value theory and statistics of univariate extremes: a review*, *International Statistical Review* (2015).

[78] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, *An introduction to mcmc for machine learning*, *Machine Learning* (2003).

[79] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, *Proceedings of the IEEE* (1998).

[80] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, tech. rep., Citeseer, 2009.

[81] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, *arXiv preprint arXiv:1409.1556* (2014).

[82] V. Chandola, A. Banerjee, and V. Kumar, *Anomaly detection: A survey*, *ACM Computing Surveys* **41** (July, 2009) 1–58.

[83] R. Chalapathy and S. Chawla, *Deep Learning for Anomaly Detection: A Survey*, *arXiv:1901.03407* (Jan., 2019).

[84] F. T. Liu, K. M. Ting, and Z. Zhou, *Isolation forest*, in *Proceedings of IEEE International Conference on Data Mining*, pp. 413–422, Dec, 2008.

[85] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, *Estimating the support of a high-dimensional distribution*, *Neural Computation* **13** (2001), no. 7 1443–1471.

[86] D. M. Tax and R. P. Duin, *Support vector data description*, *Machine Learning* **54** (Jan, 2004) 45–66.

[87] L. Ruff *et. al.*, *Deep One-Class Classification*, in *International Conference on Machine Learning*, pp. 4393–4402, July, 2018.

[88] B. Zong *et. al.*, *Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection*, in *International Conference on Learning Representations*, Feb., 2018.

[89] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, *Deep Structured Energy Based Models for Anomaly Detection*, in *Proceedings of International Conference on Machine Learning*, pp. 1100–1109, June, 2016.

[90] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, *A Survey on GANs for Anomaly Detection*, *arXiv:1906.11632* (June, 2019).

[91] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, *Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery*, in *Information Processing in Medical Imaging*, vol. 10265, (Cham), pp. 146–157, 2017.

[92] P. J. Rousseeuw and K. V. Driessen, *A fast algorithm for the minimum covariance determinant estimator*, *Technometrics* **41** (1999), no. 3 212–223.

[93] S. Rayana, *ODDS library*, 2016.

[94] D. Dua and C. Graff, *UCI machine learning repository*, 2017.

[95] M.-J. Wu, J.-S. R. Jang, and Jui-Long Chen, *Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets*, *IEEE Transactions on Semiconductor Manufacturing* **28** (Feb., 2015) 1–12.

[96] M. Fan, Q. Wang, and B. van der Waal, *Wafer defect patterns recognition based on OPTICS and multi-label classification*, in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 912–915, Oct., 2016.

[97] M. B. Alawieh, D. Boning, and D. Z. Pan, *Wafer Map Defect Patterns Classification using Deep Selective Learning*, in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, July, 2020.

[98] R. di Bella, D. Carrera, B. Rossi, P. Fragneto, and G. Boracchi, *Wafer Defect Map Classification Using Sparse Convolutional Networks*, in *Image Analysis and Processing – ICIAP 2019*, (Cham), pp. 125–136, 2019.

[99] D.-Y. Du and Z. Shi, *A Wafer Map Defect Pattern Classification Model Based on Deep Convolutional Neural Network*, in *2020 IEEE 15th International Conference on Solid-State Integrated Circuit Technology (ICSICT)*, pp. 1–3, Nov., 2020.

[100] T.-H. Tsai and Y.-C. Lee, *A Light-Weight Neural Network for Wafer Map Classification Based on Data Augmentation*, *IEEE Transactions on Semiconductor Manufacturing* **33** (Nov., 2020) 663–672.

[101] M. Nero, C. Shan, L.-C. Wang, and N. Sumikawa, *Concept Recognition in Production Yield Data Analytics*, in *2018 IEEE International Test Conference (ITC)*, pp. 1–10, Oct., 2018.

[102] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, *Big Self-Supervised Models are Strong Semi-Supervised Learners*, *arXiv:2006.10029 [cs, stat]* (Oct., 2020). arXiv: 2006.10029.

[103] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A Simple Framework for Contrastive Learning of Visual Representations*, *arXiv:2002.05709 [cs, stat]* (June, 2020). arXiv: 2002.05709.

[104] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[105] H. Kahng and S. B. Kim, *Self-supervised representation learning for wafer bin map defect pattern classification*, *IEEE Transactions on Semiconductor Manufacturing* **34** (2021), no. 1 74–86.

[106] S. Filip, A. Javeed, and L. N. Trefethen, *Smooth random functions, random odes, and gaussian processes*, *SIAM Rev.* **61** (Jan., 2019) 185–205.

[107] Q. Yi, *WM-811K wafer map*, 2018.

[108] A. Lodwich, Y. Rangoni, and T. Breuel, *Evaluation of robustness and performance of early stopping rules with multi layer perceptrons*, in *2009 International Joint Conference on Neural Networks*, pp. 1877–1884, 2009.

[109] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017.