UNIVERSITY OF CALIFORNIA
RIVERSIDE

Machine Learning Approaches for VLSI Reliability Analysis

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

Doctor of Philosophy

in

Electrical Engineering

by

Wentian Jin

March 2023

Dissertation Committee:

Dr. Sheldon X.-D. Tan, Chairperson
Dr. Shaolei Ren
Dr. Daniel Wong

The Dissertation of Wentian Jin is approved:

____

____

____
Committee Chairperson

University of California, Riverside

Acknowledgments

This dissertation would not have been possible without the help and support from numerous individuals who have contributed in various ways.

First and foremost, I would like to thank my advisor, Dr. Sheldon Tan, for his invaluable guidance, support, and mentorship throughout my PhD journey. His expertise, knowledge, and dedication have been instrumental in shaping my research work and academic career.

I am also grateful to my committee members, Dr. Shaolei Ren and Dr. Daniel Wong, for their constructive feedback, insightful suggestions, and valuable contributions to my research. Their guidance and mentorship have been instrumental in shaping the direction of my research and ensuring its quality.

Furthermore, I would like to express my sincere appreciation to my fellow researchers at the VLSI Systems and Computation lab for their help and support. Especially, I want to thank Chase Cook, Zeyu Sun, Han Zhou, Shaoyi Peng, Sheriff Sadiqbatcha, Liang Chen, Jinwei Zhang, Shuyuan Yu, Yibo Liu, Maliha Tasnim, Mohammadamir Kavousi, Subed Lamichhane, Jincong Lu, Sachin Sachdeva and Chinmay Raje. Their insights, feedback, and discussions have been critical in shaping my research and broadening my perspective. I am grateful for the opportunity to work with such a talented and inspiring group of researchers.

Last and most importantly, I would like to thank my family, especially my wife Yirong, my father Yaping, and my mother Aihua, for their love, support, and encouragement. Their unwavering belief in me and my abilities has been a constant source of

motivation throughout my PhD journey. I am grateful for their sacrifices, understanding, and patience during this challenging time.

Once again, I extend my heartfelt gratitude to all those who have helped me in achieving this milestone. Thank you.

The content of this thesis is reprinted or rewritten from these published materials:

- Wentian Jin, Sheriff Sadiqbatcha, Zeyu Sun, Han Zhou, and Sheldon X-D Tan. "Emgan: Data-driven fast stress analysis for multi-segment interconnects". Proceedings of the 2020 IEEE 38th International Conference on Computer Design (Chapter 2)
- Wentian Jin, Liang Chen, Sheriff Sadiqbatcha, Shaoyi Peng, and Sheldon X-D Tan. "Emgraph: Fast learning-based electromigration analysis for multi-segment interconnect using graph convolution networks". Proceedings of the 2021 58th ACM/IEEE Design Automation Conference (Chapter 2)
- Wentian Jin, Liang Chen, Subed Lamichhane, Mohammadamir Kavousi, and Sheldon X-D Tan. "HierPINN-EM: Fast Learning-Based Electromigration Analysis for Multi-Segment Interconnects Using Hierarchical Physics-Informed Neural Network". Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design (Chapter 3)
- Wentian Jin, Sheriff Sadiqbatcha, Jinwei Zhang, and Sheldon X-D Tan. "Full-Chip Thermal Map Estimation for Commercial Multi-Core CPUs with Generative Adversarial Learning". Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (Chapter 4)

- Wentian Jin, Shaoyi Peng, and Sheldon X-D Tan. “Data-driven electrostatics analysis based on physics-constrained deep learning”. Proceedings of the 2021 Design, Automation and Test in Europe Conference (Chapter 5)

To my mother Aihua, my father Yaping, and my wife Yirong for all the love and support.

ABSTRACT OF THE DISSERTATION

Machine Learning Approaches for VLSI Reliability Analysis

by

Wentian Jin

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, March 2023
Dr. Sheldon X.-D. Tan, Chairperson

The reliability of Very Large Scale Integration (VLSI) circuits is crucial in modern electronic devices. VLSI circuits, which contain millions of transistors, are vulnerable to a variety of reliability issues such as electromigration (EM), time-dependent dielectric breakdown (TDDB), and temperature variation. These issues can lead to circuit failure and reduce the lifetime of electronic devices. Traditionally, VLSI reliability analysis and prediction have been performed using physics-based models and simulators. These models, however, are computationally intensive and can be time-consuming to run. In recent years, machine learning (ML) techniques have been used to predict and diagnose reliability issues in VLSI circuits.

This thesis presents an in-depth study of machine learning techniques applied to EM analysis, post-silicon thermal map estimation, and electrostatics analysis. Specifically, the first segment proposes two data-driven ML methods for the fast prediction of transient EM stress of general interconnects in VLSI circuits. The traditional numerical partial differential equation (PDE) problem is treated as an image processing or graph aggregation problem which yields considerable speedup with acceptable accuracy. However, these methods are still supervised learning approaches, requiring extensive training data generated from numerical solvers. Therefore, the second segment proposes a hierarchical physics-informed neural networks (PINN) based method for EM analysis. This approach leverages PINN, which is trained mainly by physics laws with minimal labeled data requirements. The hierarchical nature of interconnects is leveraged, and the entire interconnect tree is solved step by step. Temperature variation has always been problematic in VLSI circuits, as reliability degrades drastically as temperature varies. The third segment presents a real-time thermal map estimation method for commercial VLSI circuits. This approach treats thermal modeling as an image-generation task using generative neural networks (GANs), producing tool-accurate thermal map estimations. Electrostatics analysis is an essential step for analyzing TDDB, an important failure mechanism for interconnects. Lastly, the fourth segment presents a PINN-based 2D electric field analysis method. This approach eliminates the heavy dependence of finite element methods (FEM) used in traditional TDDB analysis and leads to orders of magnitude of speedup.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Electromigration Analysis for VLSI Interconnects

Electromigration (EM) is a primary long-term reliability concern for copper-based back-end-of-the-line interconnects used in modern semiconductor chips. As predicted by the International Technology Roadmap for Semiconductors (ITRS), EM is projected to only get worse in future technology nodes [2]. This, as with many other reliability effects, is due to the continued trend of feature-size reduction and rapid integration, which ultimately affects the critical sizes for the EM failure process. EM-related aging and reliability will become worse for current 7nm and below technologies. As a result, it is crucial to ensure the reliability of the very large scale integration (VLSI) chips during their projected lifetimes.

For EM analysis, it is well known that existing Black and Blech-based EM models [11, 12] are overly conservative and can only work for single wire segment [34, 79]. To mitigate these problems, recently many physics-based EM models and simulation techniques have been proposed [24, 37, 81, 14, 62, 15, 13, 21, 92, 106, 4, 16, 83, 77, 85]. These computational techniques primarily focus on finding a solution of the Korhonen equations [46], which are the partial differential equations (PDEs) describing the hydrostatic stress evolution in confined multi-segment interconnects subject to blocking material boundary conditions. A number of conventional numerical and analytical methods have been proposed to attempt to solve the PDEs efficiently and accurately [22, 80, 106, 93, 16]. Although numerical methods, such as finite difference method [22, 80] and finite element method [106], can work for complex interconnect structures and obtain EM stress accurately, they impose high computational costs due to the discretization of space and time. Recently, a semi-analytical method based on the separation of variables method has been proposed [93, 16], which shows promising performance in both accuracy and efficiency for general multi-segment interconnects. However, solving the Korhonen equation in particular and PDEs in general by traditional numerical methods still remains a significant challenge due to the inherent limitations of those methods.

Recently, breakthroughs in deep learning for cognitive tasks based on deep neural networks (DNNs) have brought new opportunities for solving differential equations in many applications in the electric design automation (EDA) field [49]. However, how to apply deep learning techniques to solve nonlinear partial differential equations still remains in its infancy.

On the other hand, scientific machine learning (SciML) has emerged as a promising and alternative solution to traditional numerical analysis techniques for solving PDEs. The main concept is to replace traditional numerical discretization with a DNN that approximates the solution of the PDE. One important framework is the so-called physics-informed neural networks (PINN), which use differentiable DNNs to regularize the loss functions via backpropagation-based training to obtain physics-informed/constrained surrogate models [68, 97]. The resulting models can quickly infer the solutions of the PDE for all input coordinates and parameters. Recently, a PINN-based EM analysis approach has been proposed in [35]. The authors show that PINN can be applied to solving the PDE for stress evolution in confined metal. However, it only demonstrated the solution on simple interconnect straight wires. Additionally, our study shows that the plain PINN method does not work well for large interconnect trees.

To address these challenges, we propose two data-driven deep learning techniques and a hierarchical PINN-based model for fast transient hydrostatic stress analysis. The specific contributions of this segment are as follows:

- First, we show that EM analysis, modeled by partial differential equations, can be viewed as a 2D image-to-image transforming process. Then, we propose to explore the conditional generative adversarial networks (GAN) [28] structure in which the input images, which are the multi-segment interconnects topology with current densities, are treated as conditions.
- To learn the temporal dynamics in the transient EM analysis, we further explore the conditional GAN structures to use the *time* variable as the continuous condition for generator and discriminator. We show such a time-conditional GAN works well for time-varying stress modeling.
- Different hyperparameters of GAN were studied and compared. We use current densities of wire segments and aging time as the conditions for the conditional GAN.

The resulting *EM-GAN* can quickly give an accurate stress distribution of any multi-segment wires for a given aging time.

- Second, we apply graph neural network (GNN) to perform transient EM stress on multi-segment interconnect for the first time to the best of our knowledge. A graph dataset on EM assessment is created using COMSOL multiphysics. The input of the GNN model is edge features, such as length, width, current density, a graph structure, and time. Its output is the stress on the edges. Then, we can estimate the hydrostatic stress in each segment wire at the given time.
- We design our own graph neural network (called *EMGraph*) to perform the node-edge regression task based on the popular GraphSage network. Compared with the GAN-based method, the proposed *EMGraph* model can learn more transferable knowledge to predict stress distributions on new graphs without retraining via inductive learning. We use *EMGraph* to predict EM stress on large and unseen designs with good accuracy and fast speed, which cannot be achieved by *EM-GAN* method [42] because of its size-fixed image limitation. In addition, the size of the *EMGraph* model is much smaller than that of the *EM-GAN* model.
- A novel graph convolution-decoder structure is employed in the *EMGraph* model. Our model first processes the input graph using graph convolution. The resulting graph embedding features are then fed into node and edge decoders, which convert latent features to stress outputs.
- Lastly, we show that the plain PINN-based unsupervised learning does not work very well for interconnects with a large number of segments in terms of accuracy and

training speed. To mitigate this problem, we propose a new *hierarchical PINN* solving strategy to reduce the number of variables, which can lead to faster training and more accurate models. The resulting PINN training framework is called *HierPINN-EM* for fast EM-induced stress analysis for multi-segment interconnects.

- In the *HierPINN-EM* framework, the solving process consists of two steps (levels). The first step is to find a parameterized solution for single-segment wires under different boundary conditions, geometrical parameters, and stressing current densities. This step can be solved by different approaches. In this work, we apply the supervised learning method to build the DNN model with a parameterized input layer as a universal solution to different single-segment wires under various boundary conditions.
- In the second step of *HierPINN-EM*, we apply the existing unsupervised PINN concept to solve the stress and atom flux continuity problem in the interconnects by enforcing the physics laws at the boundaries of all wire segments. In this way, *HierPINN-EM* can significantly reduce the number of variables at the PINN solver, which leads to faster training speed and better accuracy than the plain PINN method.
- The proposed *HierPINN-EM* framework consists of two steps. The first step is to find a parameterized solution for single-segment wires under different boundary conditions, geometrical parameters, and stressing current densities. This step can be solved by different approaches. In this work, we apply the supervised learning method to build the DNN model with a parameterized input layer as a universal solution to different single-segment wires under various boundary conditions. In the second step, we apply the existing unsupervised PINN concept to solve the stress and atom flux continuity

problem in the interconnects by enforcing the physics laws at the boundaries of all wire segments. In this way, *HierPINN-EM* can significantly reduce the number of variables at the PINN solver, which leads to faster training speed and better accuracy than the plain PINN method.

Our experimental results demonstrate that:

EM-GAN is capable of quickly and accurately providing stress distribution of any multi-segment wires for a given aging time. Compared to COMSOL [1] simulation results, our experimental results show that the *EM-GAN* method has an average error of 6.6% and orders of magnitude speedup. It also provides an 8.3× speedup over the previous state-of-the-art analytic-based EM analysis solver [16].

EMGraph yields an average error of 1.5% compared to the ground truth results, and it is orders of magnitude faster than both COMSOL and the state-of-the-art method. It also achieves better accuracy and a 14× speedup over the *EM-GAN* method on several interconnect tree benchmarks.

HierPINN-EM outperforms the plain PINN method in both accuracy and performance with over 79× error reduction and orders of magnitude speedup, indicating much better scalability. *HierPINN-EM* also yields 19% better accuracy with 99% reduction in training cost over *EMGraph*.

1.2 Real-Time Full-Chip Thermal Map Estimation

As technology advances, high-performance microprocessors are becoming increasingly thermally constrained due to steadily increasing power densities [86]. To enhance reliability, many system-level thermal/power regulation techniques such as clock gating, power gating, dynamic voltage and frequency scaling (DVFS), and task migration have been proposed in the past [33, 55, 90]. One critical aspect of the algorithms mentioned above is correctly estimating the full-chip temperature profile to guide the online thermal management schemes properly [47, 45]. However, accurate thermal estimation is a difficult task, especially for commercial off-the-shelf multi-core processors.

Some of the existing methods depend on on-chip temperature sensors. However, typically, only a few physical sensors are available, and they may not be located in close proximity to the true hotspots on the chip, misleading the temperature regulation decision [7]. Hence, the more popular solution is to supplement the data from the few on-chip sensors with estimated temperatures of all the prominent hotspots on the chip via thermal models based on estimated power-traces [9]. These methods offer higher spatial resolution as they allow for the temperature of all the hotspots on the chip to be monitored in real-time [64, 36, 96, 88].

However, existing thermal modeling methods still suffer a few drawbacks. Firstly, they require accurate power-traces as inputs, but estimating the power of each functional unit (FU) of a real processor under varying workloads is not a trivial task, if not infeasible [94, 25]. On the other hand, from the system-level thermal or power management perspective, the parameters that can be easily accessed are the core frequency, voltage, and many utilization or performance metrics natively supported by most commercial processors [103]. Examples include Intel's Performance Counter Monitor (PCM) [38] and AMD's uProf [6]. Thermal models parameterized by these parameters will be more desirable and practical. Secondly, it is difficult to calibrate these models for practical use due to simplified modeling, boundary conditions, and the lack of sufficient accuracy. Lastly, most models such as HotSpot [36] still employ expensive numerical methods to find temperature solutions, which may not be fast enough for real-time use.

On the other hand, estimating the full-chip 2D thermal map of multi-core CPUs from given performance monitor parameters can be viewed as an imaging synthesis problem. We can treat the performance monitor parameters as extracted latent features for power information of the chip. Then we can synthesize the 2D thermal maps once the neural network is trained for the utilization to temperature transformation. Such training and image generation process can be carried out using GAN, a popular generative deep neural network for imaging synthesis, semantic imaging editing, style transfer, image superresolution, etc [28, 23].

Inspired by this observation, in this work, we propose a novel data-driven fast transient full-chip thermal map estimation method for multi-core commercial CPUs by exploiting conditional generative adversarial learning. The specific contributions of this segment are as follows:

- First, *ThermGAN* can be implemented on most, if not all, existing commercial multi-core microprocessors, as it only uses the existing temperature sensors and workload-independent utilization information. In other words, our strictly post-silicon approach does not require any modifications to the chip's design.
- We propose to treat this existing thermal modeling problem as an image generation problem conditioned on high-level performance monitors, which are available in most,

if not all, commercial microprocessors. We then propose to explore the conditional generative neural network structure in which the input high-level performance data is treated as categorical conditions.

- In our work, we use a simple memory-less convolutional neural network for both generator and discriminator, with Wasserstein distance as the loss function. We demonstrate that the proposed *ThermGAN* can estimate transient and real-time thermal maps without using any historical data for training and inferences, which is in contrast with a recent LSTM-based thermal map estimation method in which historical data is needed [74].
- We use an advanced infrared thermography setup system that enables clear heatmaps to be recorded directly from commercial microprocessors while they are under load. A total of 257,400 pairs of PCM data and thermal maps were collected, and 75% were used for training.
- The resulting *ThermGAN* can provide tool-accurate full-chip *transient* thermal maps from the given performance monitor traces of commercial off-the-shelf multi-core processors.

Experimental results show that the trained model is very accurate in thermal estimation with an average RMSE of 0.47°C, namely, 0.63% of the full-scale error. Our data further show that the speed of the model is less than *7.5ms* per inference, which is two orders of magnitude faster than the traditional finite element-based thermal analysis and is suitable for real-time thermal estimation. Furthermore, the new method is ~4x more accurate than the recently proposed LSTM-based thermal estimation method [74] and has a faster inference speed. It also achieves $\sim$2x accuracy with much less computational cost than the EigenMaps method [70], which is a state-of-the-art pre-silicon method.

1.3 Electrostatics Analysis for VLSI Interconnects

Electrostatics is an important subject of study as it is pivotal in many VLSI modeling applications. The goal is to compute voltage potential and electric fields with some voltage and current boundary conditions for dielectrics and metal interconnects or planes. In the back-end of VLSI manufacturing, a strong electric field can induce failure of the dielectrics, which is known as Time-dependent dielectric breakdown (TDDB) [60]. Simulation of this aging effect requires electrostatics. Also, several methods of parasitic extraction involve electrostatics simulations in the chip layouts [18]. Recently, global placement is also modeled as electrostatic problem [57].

Traditionally, this problem is primarily solved by numerical methods with spatial discretization of the governing equation using polynomials into a finite-dimensional algebraic system (as is done in finite element method (FEM) or finite difference method (FDM)). Such numerical methods typically require meshing of a complicated layout or geometry, which can be very computationally prohibitive for large problems.

On the other hand, DNNs have propelled an evolution in machine learning fields and redefined many existing applications with new human-level AI capabilities. DNNs such as convolutional neural networks (CNNs) have recently been applied to many cognitive applications such as visual object recognition, object detection, speech recognition, natural language understanding, etc. due to dramatic accuracy improvements in those tasks [49].

However, how to apply deep learning techniques to learn and encode laws of physics and help to solve nonlinear partial differential equations still remains in its infancy.

Recently, the so-called PINN or physics-constrained neural networks (PCNNs) have been proposed to learn and encode physics laws expressed by nonlinear PDEs for complex physical, biological or engineering systems [66, 67]. The idea is to use DNNs, which are differentiable, to regularize the loss functions via back-propagation based training to obtain so-called physics-informed/constrained surrogate models, which can quickly infer the solutions of the PDEs to all input coordinates and parameters. The promise of PINN/PCNN is that we only need a small number of training examples and the resulting DNN models will quickly respect the underlying principled physics laws for all the input coordinates and parameters so that fast and accurate numerical solutions can be obtained. Another significant benefit of the PINN/PCNN idea is that they are mesh-free compared to traditional FEM or FDM based methods. However, only very simple PDE problems were demonstrated in [68, 97, 78, 10, 67], although some progress was made for more complicated aerodynamics simulation recently [19].

Inspired by recent progress with PINN/PCNN for solving partial differential equations, in this work, we propose a new data-driven 2D analysis of electric potential and electric fields based on a physics-constrained deep learning scheme, called *PCEsolve*. The specific contributions of this segment are as follows:

- We first show how to formulate the differential loss functions to consider the Laplace differential equations with voltage boundary conditions for a typical electrostatic analysis problem. The resulting solving process becomes a nonlinear optimization process, which is solved by the back-propagation method in existing DNN networks. We apply the resulting *PCEsolve* solver to calculate electric potential and electric field for VLSI interconnects with complicated boundaries.

- Our study for purely label-free training (in which no information from FEM solver is provided) shows that *PCEsolve* can get accurate results around the boundaries, but the accuracy degenerates in regions far away from the boundaries. To mitigate this problem, we explore adding some simulation data or labels at collocation points derived from FEM analysis. We explore both voltage and electric field (1st order derivative) label information, and the resulting *PCEsolve* can be much more accurate across all the solution domain.
- We also studied the impacts of weights on different components of loss functions to improve the model accuracy for both voltage and electric field.

Experimental results demonstrate the *PCEsolve* achieves an average error rate of 3.6% on 64 cases with random boundary conditions, and it is 27.5× faster than COMSOL on test cases. The speedup can be further boosted to $\sim 38,000\times$ in single-point estimations. Our study shows that the current PINN/PCNN frameworks have some potentials for solving practical electrostatics analysis but with limited accuracy.

1.4 Organization

The rest of this thesis is organized as follows. Chapter 2 presents two data-driven learning-based EM analysis methods for multi-segment interconnects in VLSI circuits, namely GAN-based model *EM-GAN* and GNN-based model *EMGraph*. To eliminate the heavy dependency on simulated labeled data, Chapter 3 presents a hierarchical PINN-based method, called *HierPINN-EM*, to solve the Korhonen equations for multi-segment interconnects for fast EM failure analysis. Chapter 4 presents a data-driven full-chip transient thermal map estimation method for commercial multi-core microprocessors using the generative adversarial learning technique. Chapter 5 presents a 2D electric field analysis approach based on the concept of physics-constrained deep learning. Finally, Chapter 6 concludes the thesis.

Chapter 2

Data-Driven Learning-Based Electromigration Analysis

2.1 Related Work and Motivation

2.1.1 Review of EM and EM modeling

EM is a diffusion phenomenon of metal atoms migrating from cathode to anode of confined metal interconnect wires due to the momentum exchange between the conducting electrons and metal atoms [11]. With the EM driving force, the hydrostatic stress increases over time. When the stress reaches a critical value, a void is nucleated at the cathode and ahillock is created at the anode of the interconnects. This eventually leads to an open or short circuit, which is an EM-induced reliability problem in modern VLSI circuits.

Black's equation predicts the EM-induced time-to-failure (TTF) based on empirical or statistical data fitting, which only works for a specific single wire [11]. Blech's limit, which is an immortality check method, cannot estimate transient hydrostatic stress and is subject to growing criticism due to unnecessary overdesign [12]. To mitigate this problem, the physics-based EM model, Korhonen equations [46], is employed to describe the hydrostatic stress evolution for general multi-segment interconnects.

The general multi-segment interconnect consists of n nodes, including p interior junction nodes $x_r \in \{x_{r1}, x_{r2}, \cdots, x_{rp}\}$ and q block terminals $x_b \in \{x_{b1}, x_{b2}, \cdots, x_{bq}\}$, and several branches. The physics-based Korhonen's PDE for this general structure in nucleation phase can be formulated as follows [93, 16].

$$
\begin{aligned}
&\frac{\partial \sigma_{ij}(x,t)}{\partial t} = \frac{\partial}{\partial x}\left[\kappa_{ij}\left(\frac{\partial \sigma_{ij}(x,t)}{\partial x} + G_{ij}\right)\right], t > 0 \\
&BC : \sigma_{ij_1}(x_i,t) = \sigma_{ij_2}(x_i,t), t > 0 \\
&BC : \sum_{ij} \kappa_{ij}\left(\left.\frac{\partial \sigma_{ij}(x,t)}{\partial x}\right|_{x=x_r} + G_{ij}\right) \cdot n_r = 0, t > 0 \\
&BC : \kappa_{ij}\left(\left.\frac{\partial \sigma_{ij}(x,t)}{\partial x}\right|_{x=x_b} + G_{ij}\right) = 0, t > 0 \\
&IC : \sigma_{ij}(x,0) = \sigma_{ij,T}
\end{aligned}
\tag{2.1}
$$

where BC and IC are boundary and initial conditions respectively, ij denotes a branch connected to nodes i and j, n_r represents the unit inward normal direction of the interior junction node r on branch ij. $\sigma(x,t)$ is the hydrostatic stress, $G = \frac{Eq*}{\Omega}$ is the EM driving force, and $\kappa = D_a B\Omega / k_B T$ is the diffusivity of stress. E is the electric field, $q*$ is the effective charge. $D_a = D_0 \exp(\frac{-E_a}{k_B T})$ is the effective atomic diffusion coefficient. D_0 is the pre-exponential factor, B is the effective bulk elasticity modulus, Ω is the atomic lattice volume, k_B is the Boltzmann's constant, T is the absolute temperature, E_a is the EM activation energy. σ_T is the initial thermal-induced residual stress.

2.1.2 DNN based approaches for solving PDEs

In order to solve the PDE (2.1) accurately, numerical methods [22, 80, 106], such as finite difference and finite element methods, are applied for EM assessment. However, these methods require huge computational costs and are not scalable for modern chips. Therefore, an analytic-based method, called the separation of variables method, is employed to estimate the transient hydrostatic stress with eigenvalues. This method suffers from computing eigenvalues slowly and determining the number of eigenvalues [93, 16].

Deep learning has revolutionized the machine learning field with breakthroughs in many cognitive applications such as image, text, speech, and graph recognition [49, 28]. Inspired by these observations, neural networks are modified to solve PDEs.

There are several strategies for solving PDEs using DNN-based methods. One approach is to frame the PDE-solving process into a nonlinear optimization process coded by DNN with loss functions to enforce the physics laws represented by the PDE and boundary conditions. Recently proposed physics-informed neural networks [68, 67] or physics-constrained neural networks [78, 61] represent this strategy. However, most of the reported works only solve very small PDEs with simple boundary conditions. Furthermore, it is unclear whether such methods can deliver sufficient accuracy without any labels (unsupervised learning). On the other hand, the second approach uses supervised learning to build DNN models based on the measured or simulated label data. Despite the abundance of successful models in cognitive applications, identifying an appropriate representation for VLSI interconnects and formulating an effective DNN model for EM analysis remain significant challenges that demand investigation.

To harness the tremendous success of deep learning models in image processing applications, it is practical to represent VLSI interconnects as a 2-D image, and the EM analysis problem can be addressed as an image-to-image problem. The topology of interconnects can be projected onto a pixel grid while other relevant electrical or geometrical information can be stored in multiple channels of the image. This formulation makes it possible to model the EM-induced stress evolution using existing image processing models with necessary adaptations.

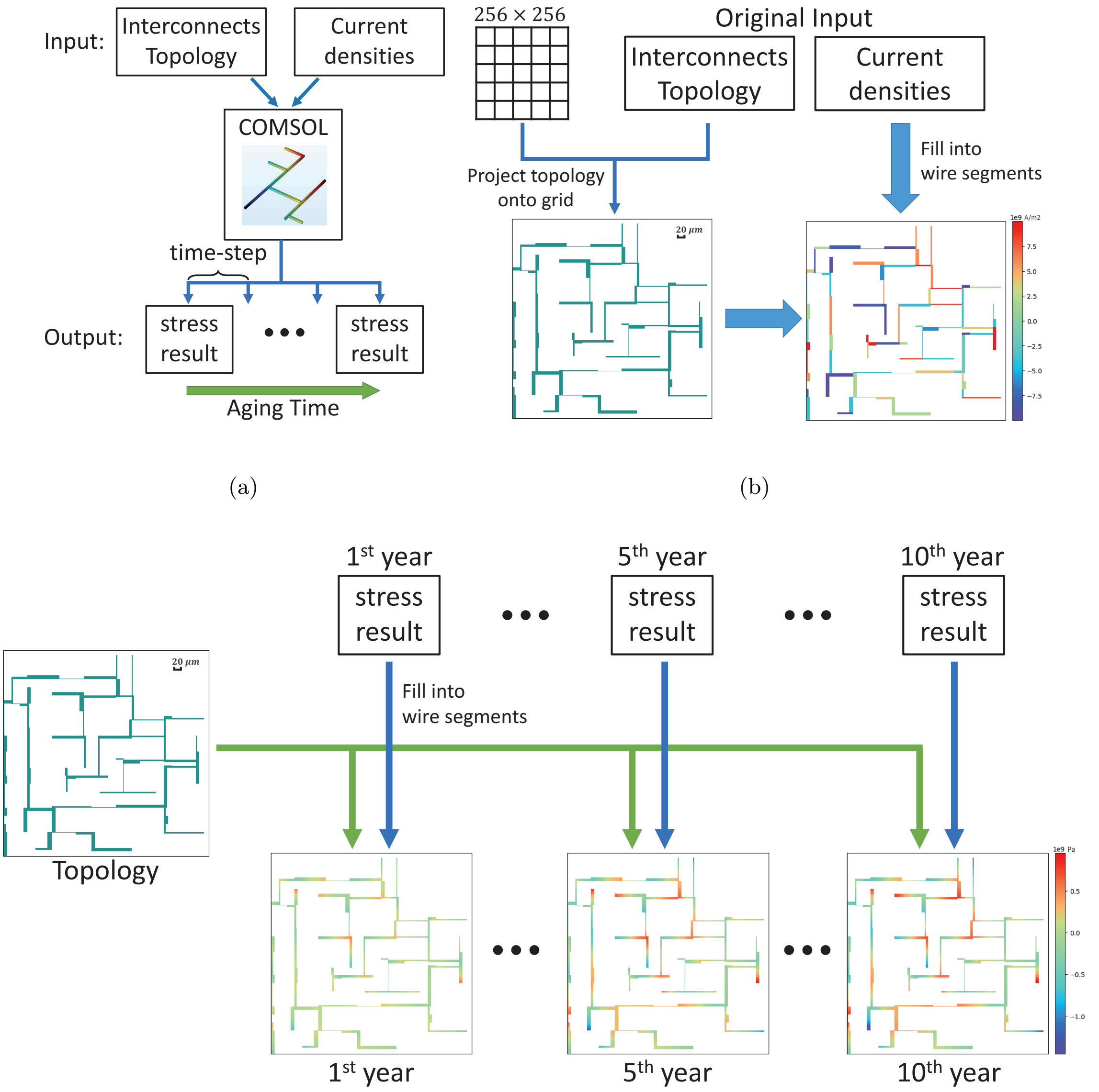
To efficiently represent the multi-segment interconnects structure, a graph is more suitable to store the node and edge information of the interconnect trees. Kipf and Welling introduced a definition of the convolution operation on a graph, which aggregates information into the node from its neighborhood nodes [44]. However, this method only works on a fixed graph because the input needs an adjacency matrix representing a graph. Once the graph is changed, the model needs to be trained again. To mitigate this problem, the GraphSAGE network is proposed for inductive learning on graphs [32]. Unlike the matrix factorization method proposed by Kipf and Welling, GraphSAGE only learns the local node features by aggregating the information from its neighborhood and can predict the features at unseen nodes. This means that the model can predict the embedding features on new graphs without retraining. Also, several works leveraging GNN have been proposed recently for solving various problems in EDA, such as analog circuit clustering [76], estimation of device parameters in [72], chip power estimation in [104], 3D circuit partitioning [58], transistor sizing [89], analog IC placement [53]. Since GNN represents more general and natural relationship among different design objectives, the knowledge learned by GNN models tends to be more transferable for different designs, which is highly desirable in this work.

2.2 EM-GAN: CGAN-Based Current Density to EM Stress Transformation

2.2.1 Data preparation

For machine learning-based approaches, one crucial aspect is sufficient training data. For our GAN-based EM stress estimation, the input data are interconnect topologies with various current densities in different wire segments, while the output is the evolution of the EM-induced stress distribution across all wire segments. The proposed *EM-GAN* is trained to model the transformation scheme between these two datasets. In what follows, we present the details of the training data and how we preprocess and map them into the domain that can be leveraged by GAN-based model.

To achieve the abundance of training data, we randomly generated 2500 different topologies of multi-segment interconnects. Each of them has a different number of wire segments with random widths, lengths, and current densities. These generated topologies with current densities are then fed into COMSOL, an off-the-shelf finite element method (FEM) solver. For each input, COMSOL produces a series of stress distributions that reflect the stress evolution along the aging time. The data acquisition process is illustrated in Fig. 2.1(a). The time-step between two adjacent results can be adjusted to obtain the best trade-off between accuracy and performance.



(a) (b)

(c)

Figure 2.1: Illustration of training data: (a) Raw data acquisition for training dataset (b) Input: A *design* with wire segments filled with current densities (c) Output: Evolution of EM-induced stress distribution along 10 aging years.

These data are all saved in numerical format. As stated in Section 2.1.2, to leverage the GAN model, we have to transform them into the image domain so that the problem is simplified as an image-to-image task.

- **Input:** Every interconnects topology is composed of rectangular wire segments with random sizes. When generating the topologies, we set the bounds for both x- and y-dimensions to 256 μm, and the resolution of the wire segments is set to 1 μm. With this configuration, we can easily project the topology onto a 256 × 256 grid, which can also be seen as a single-channel image, as shown in Fig. 2.1(b). We note that this configuration does not restrict our work to only small-size interconnects, as in real cases, bulk interconnect system may be divided into small pieces with partitioning algorithms for parallel calculation. The proposed *EM-GAN* is used as the solver for small partitions, and the results can be synthesized back to form the final results for the original bulk interconnect system. Another input is the current density, which is generated by applying random current sources to the interconnects. In each topology, current density varies drastically among different wire segments but is equally and uniformly distributed within the same segment. To combine these two inputs into a single image, we fill every wire segment with its current density, and the resulting single-channel image is shown in Fig. 2.1(b). In this work, we refer to every combined input of topology and current density as a *design*.

- **Output:** The results we obtain from COMSOL for each *design* are a time-series of gradually changing stress distributions. In this work, the maximum aging time is set to 10 years, and we reserve 10 results from the 1st to the 10th year for training purposes. Similar to the raw data of current densities, the stress distributions are also saved in numerical format, so they can also be combined with topologies. The combined result is referred to as a *stress map* in this work. The combination process together with the resulting *stress maps* at the 1st, 5th, and 10th aging years are illustrated in Fig. 2.1(c). Each *stress map* can also be seen as a single-channel image with the same 256×256 size as the input *design*. The difference is that each pixel in *stress maps* represents the stress value in the corresponding 1 μm^2 area, and for each input *design*, there are 10 resulting *stress maps*.

Feeding all 2500 randomly generated *designs* into COMSOL results in 25000 stress distributions, which are then organized into a training dataset with 2500 pairs of (**Input** : 1 *design*, **Output** : 10 *stress maps*) samples. The 1-to-10 relationship within each data pair implies that a single-input multiple-output (SIMO) model is required, while traditional GANs are only capable of single-input single-output (SISO) modeling. The technique we use to overcome this barrier will be detailed in Section 2.2.2.

Now that both the input and output have been transformed into the image domain, a GAN-based model can be leveraged to solve the proposed problem as an image-to-image task, as illustrated in Fig. 2.2. However, some preprocessing is still required before the data can be fed into the model. Since there is only one channel in the image, the figures shown in Fig. 2.1 are depicted as heat maps in which the colors are only for visualization purpose. In a typical color image, pixels usually have red-green-blue channels, and the values are limited to the range of 0 to 255, which is not the case in our dataset. Pixels in *design* and *stress map* are filled with real values of current density and stress, respectively. In this work,

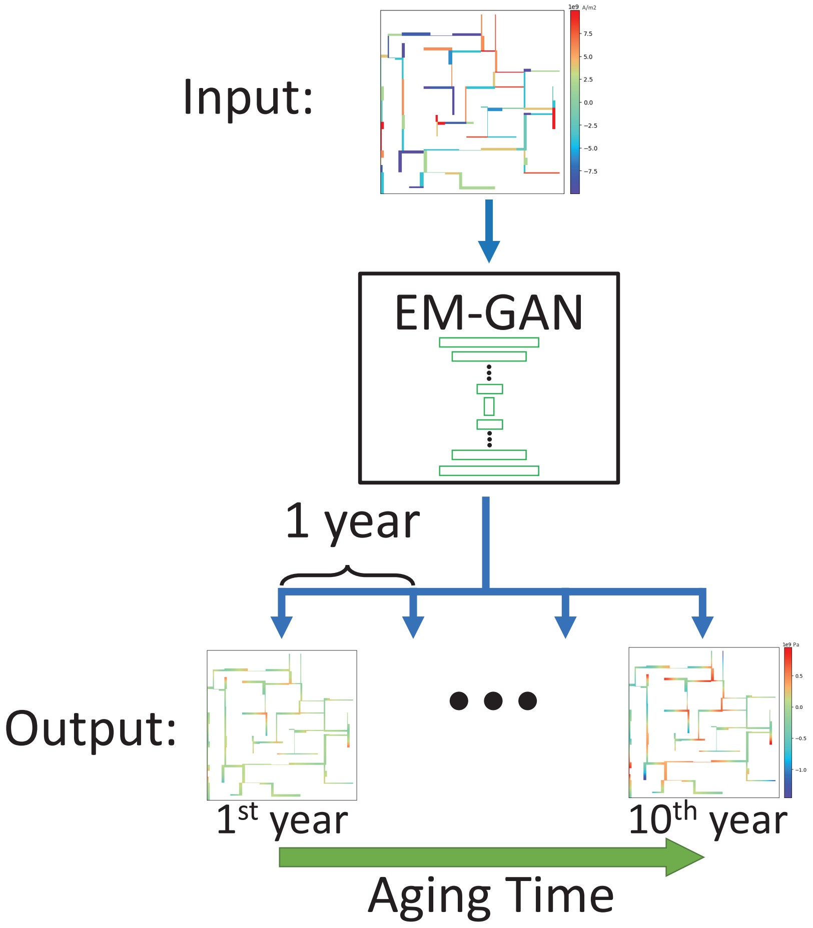


Figure 2.2: EM-GAN models the stress estimation as an image-to-image task

both current density and stress can range drastically from magnitudes of -10^9 to 10^9. The positive sign here denotes the direction toward right and up, and vice versa. It is commonly accepted that values around zero are more numerically stable for neural networks. Thus, we have to scale our dataset down to such a range. In this work, we rescale all samples to zero mean and unit standard deviation using data standardization method. All values are squeezed into the range of -7 to 7 with the majority of values around zero.

2.2.2 Problem formulation

The current density image to EM stress image transformation

We first show that we can view the PDE solving process for a multi-segment wire as image synthesis process, in which the DNN can automatically extract features reflecting the physics-law of stress evolution in the confined metal wire. Then we can use the DNN network to map the input images of interconnect wires with stressing current to the stress distributions of wire segment for any given aging time.

Review of GANs

Generative Adversarial Networks (GANs) are widely used generative models that consist of two neural networks: (1) a Generator G, which is trained to produce real-like data that mimics the samples in the training set, and (2) a discriminator D, which takes either real or fake data as input and aims to discriminate between them. The input of G is usually random noise z, which follows a certain distribution. Thus, the generated output is also a random sample extracted from the distribution of fake data. The training of GAN requires both G and D to be trained simultaneously in an alternative fashion, and the final goal is to let the distribution of fake data overlap with that of the training set. The output of D measures the similarity between these two distributions, and usually, the Jensen-Shannon Divergence is employed as the measurement. To reduce the randomness in the generated data, Conditional GAN (CGAN) was created to provide a certain extent of control on the output of G. CGAN is a variant of GAN that introduces additional condition input so that the fake data distribution is conditioned on it. CGANs have been widely used as a conditional generation method and are at the forefront of a wide range of applications.

Time dependent architecture

GANs are designed for static applications where a single input always leads to a single output. However, our dataset consists of 2500 pairs of (1 *design* → 10 *stress maps*) samples, which requires the model to be able to generate a sequence of stress distributions across all the aging years using only one *design* as input. To overcome the barrier between traditional GAN and the time-dependent data, we propose the *EM-GAN*, which is a CGAN-based model with the capability of time-variant output synthesis.

There are some recent studies trying to preserve the temporal dynamics through modifications of GAN architecture. In TimeGAN [100], additional auxiliary networks called embedding and recovery are added to learn the temporal information of data. Other researchers employ recurrent neural networks (RNN), which is a natural architecture for time series modeling, in both generator and discriminator for time series data augmentation [69] and missing value imputation for multivariate time series data [59]. These existing works mostly deal with simulated or size-limited synthetic data, in which employing RNNs will not cause too much overhead. However, in this work, we are dealing with a time-dependent image synthesis problem where both input and output are of quite large sizes (256 × 256 pixels). Such large data throughput results in a heavy model, and integrating it recursively in an RNN-like architecture will lead to a bulky network that can be expressed as

$$\begin{aligned} &p\left(z,0\right) \overset{G}{\rightarrow} p\left(\hat{y}_1 \mid z\right) \\ &p\left(z,\hat{y}_{1:t-1}\right) \overset{G}{\rightarrow} p\left(\hat{y}_t \mid z,\hat{y}_{1:t-1}\right) \end{aligned} \tag{2.2}$$

where $\mathbf{z}$ is a random *design*, $\mathbf{G}$ the generator model, and $\boldsymbol{\hat{y}_t}$ the estimated stress distribution produced at the $\boldsymbol{t^{th}}$ time-step, which is conditioned on both *design* and history results. This is not a practical architecture due to the significant computational overhead it would introduce in both training and inference. Additionally, considering the fact that EM-induced stress continuously evolves over 10 years, such a large time range further impedes the employment of RNN, which otherwise would produce numerous intermediate results at each time-step before the final aging year is reached. In real cases, designers only care about whether the interconnects are able to last before the chip lifetime is reached, which implies that getting only the stress result at the specified aging year is enough. The intermediate results are only needed when a wire failure is spotted, and further investigation into the stress evolution is required.

Based on these observations, we propose the *EM-GAN* model, which employs a CGAN as the backbone and is illustrated in Fig. 2.3. The *design* $\mathbf{z}$ is taken as one input, and another input is the explicitly specified aging year $\mathbf{t}$, which serves as the time condition. Compared with the sequential network in (2.2), *EM-GAN* is simplified to directly map the *design* to the *stress map* at the conditioned aging year with no intermediate result generated. Additionally, if stress-induced failure is found in the *stress map*, a backward investigation can be conducted by changing the input aging year to previous time-steps, such that the detailed evolution of the *stress map* can be gathered and analyzed.

With such a time-conditioned architecture, a single *design* can be projected onto multiple *stress maps* by varying the time condition input. The proposed *EM-GAN* model can be expressed as

$$p(z,t) \overset{G}{\rightarrow} p(\hat{y} \mid z,t) \tag{2.3}$$

where **z** is a random *design*, **t** the specified aging year, **G** the generator network, and $\boldsymbol{\hat{y}}$ the *stress map* estimated by the generator conditioned on time **t**.

2.2.3 EM-GAN architecture

As shown in Fig. 2.3, the generator **G** of *EM-GAN* takes the *design* image $\boldsymbol{img_{des}} \in \mathbb{R}^{256\times256\times1}$ and the aging year $\boldsymbol{t} \in \mathbb{R}$ as input. The scalar value $\boldsymbol{t}$ is expanded into $\mathbb{R}^{256\times256\times1}$ through channel-wise duplication, so that $\boldsymbol{img_{des}}$ and $\boldsymbol{t}$ can be concatenated element-wise into a two-channel image **x** with the size of $256 \times 256 \times 2$. This two-channel image is then taken as the real input of the generator **G**. The architecture employed for **G** is an encoder-decoder network, which is widely used in image-to-image applications. In this network, the input **x** is first downsampled through a series of convolutional layers until a bottleneck layer is reached, where the extracted latent features are saved. These features may contain various abstract information, such as physics-law and temporal dependency. The rest of **G** leverages the extracted features and generates the *stress map* by upsampling them through transposed convolutional layers. However, a drawback of this encoder-decoder network is that all information passes through the narrow bottleneck layer in the middle, which is not

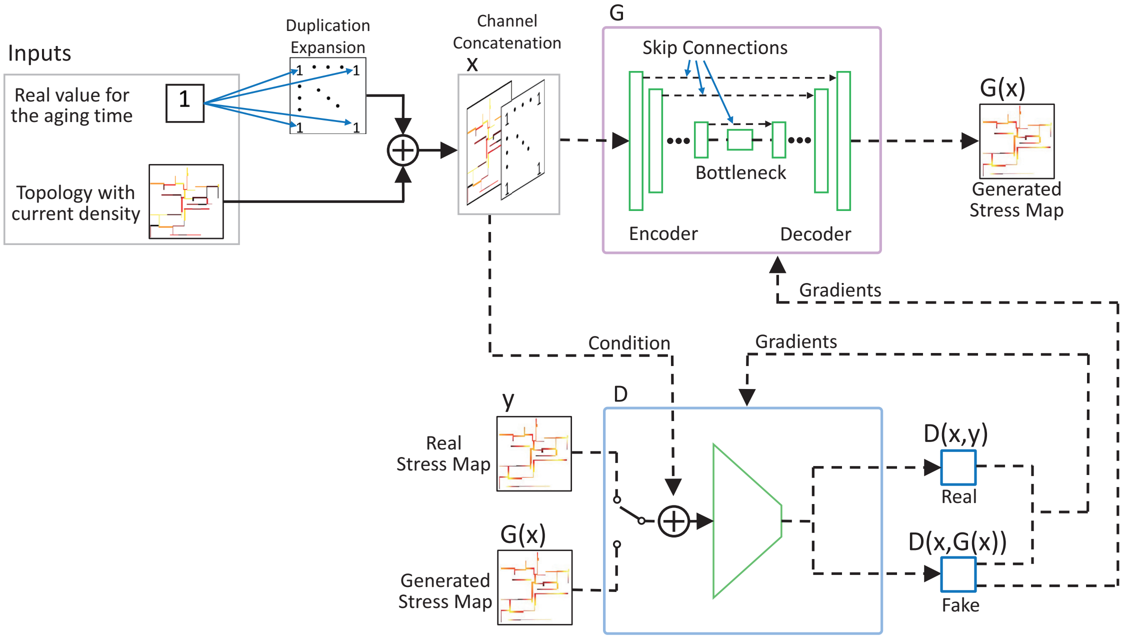


Figure 2.3: EM-GAN framework for stress estimation

necessary. In this work, both input *design* and output *stress map* share the same topology of interconnects. The extraction and reconstruction of such geometric information leads to excessive overhead in both computation and bandwidth. To make the model focus solely on the processing of temporal and physical features, we add skip connections between the encoder and the decoder, as shown in Fig. 2.3. With this configuration, the bottleneck layer is bypassed, and the geometric information is passed through the shortcuts directly from the encoder to the decoder. Skip connections can greatly improve the output accuracy, which will be discussed in detail in Section 2.4.1.

The output *stress map* of the generator is denoted as $\mathbf{G}(\mathbf{x})$ and referred to as fake *stress map* in this work. In the training process, either a fake $\mathbf{G}(\mathbf{x})$ or a real *stress map* $\mathbf{y}$ from the training set is fed into the discriminator $\mathbf{D}$ together with its corresponding *design* and aging time $\mathbf{x}$. The discriminator then judges whether the *stress map* is real or fake, based on the given $\mathbf{x}$. The output of the discriminator is a scalar score, which is denoted as $\mathbf{D}(\mathbf{G}(\mathbf{x}), \mathbf{x})$ or $\mathbf{D}(\mathbf{y}, \mathbf{x})$, depending on which *stress map*, fake or real, was inputted. It reflects how confident the discriminator is that the *stress map* it is being fed is a real one.

The key idea of the *EM-GAN* model is to let the generator learn the mapping method from the distribution of *designs* with aging year to that of the real *stress maps*. Such transformation is achieved by progressively training the generator according to the gradients backpropagated from the loss, which is based on the output of the discriminator. The generator and the discriminator are trained simultaneously but with opposite training objectives. The training goal of the discriminator is to minimize $\mathbf{D}(\mathbf{G}(\mathbf{x}), \mathbf{x})$ and maximize $\mathbf{D}(\mathbf{y}, \mathbf{x})$, which can be expressed as

$$\begin{gathered}\max_{D}\{\mathbb{E}_{\mathbf{x},\mathbf{y}}[D(\mathbf{y}, \mathbf{x})] - \mathbb{E}_{\mathbf{x}}[D(G(\mathbf{x}), \mathbf{x})] - \\ \lambda_{gp}\mathbb{E}_{\hat{\mathbf{x}}}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}}, \mathbf{x}))\|_2 - 1)^2]\}\end{gathered} \tag{2.4}$$

where $\mathbb{E}_{\mathbf{x},\mathbf{y}}[D(\mathbf{y}, \mathbf{x})]$ is the average score given by the discriminator to real *stress maps*, while $\mathbb{E}_{\mathbf{x}}[D(G(\mathbf{x}), \mathbf{x})]$ is the average score given to the fake ones. These two terms together confine the discriminator to be more confident in telling apart the real input from the fake ones. The last term in (2.4) is the gradient penalty adopted from WGAN-GP [8], which maintains the 1-Lipschitz continuity of the discriminator. $\hat{\mathbf{x}}$ is interpolation between the generated EM stress image and its ground truth, and λ_{gp} is the hyperparameter which controls the weight of gradient penalty.

On the contrary, the training objective of the generator is to produce real-like *stress maps* so that the discriminator is deceived to give high scores to the fake inputs.

Since the generator has no influence on the scores given to the real samples, term $\mathbf{D}(\mathbf{y}, \mathbf{x})$ is discarded in its objective function, which can be shown as

$$\min_{G} \{\mathbb{E}_{\mathbf{x}}[-D(G(\mathbf{x}), \mathbf{x})] + \lambda_{L2} \cdot \mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - G(\mathbf{x})\|_2]\} \tag{2.5}$$

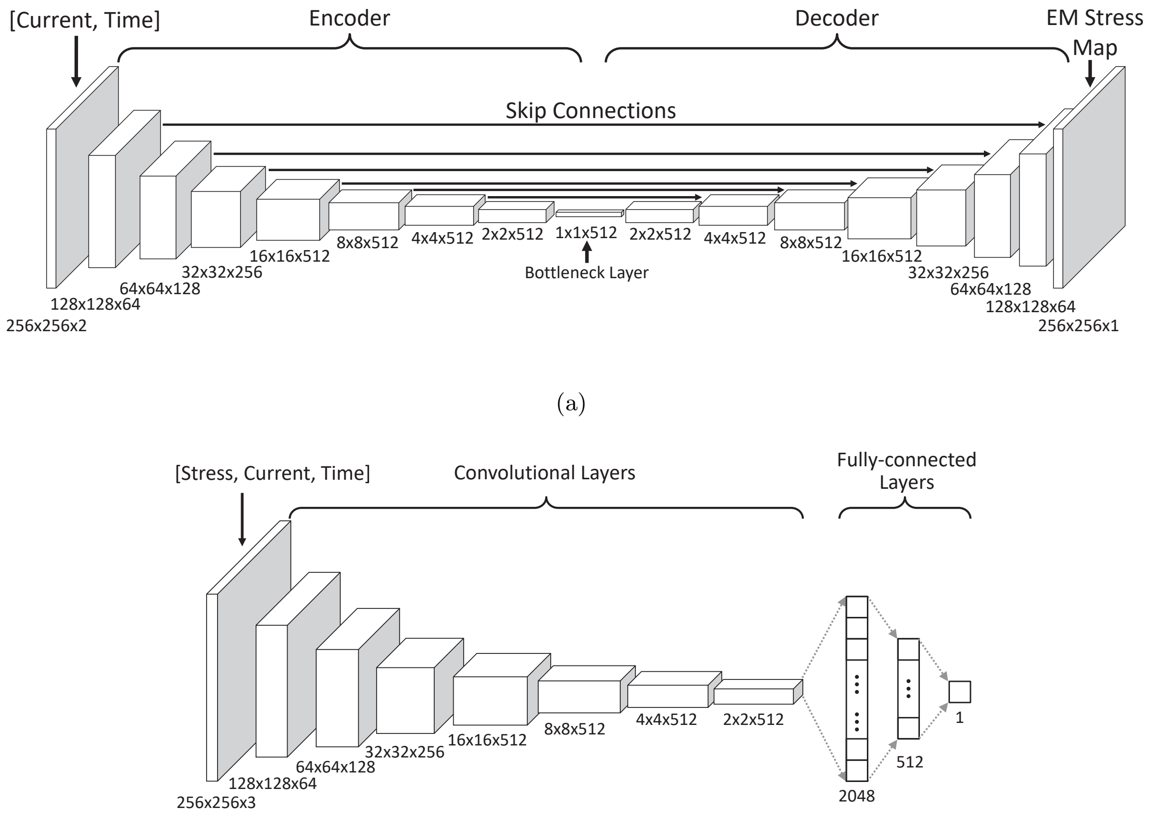
where only term $\mathbb{E}_{\mathbf{x}}[D(G(\mathbf{x}), \mathbf{x})]$ is reserved. We also add the average L2-norm $\mathbb{E}_{\mathbf{x},\mathbf{y}}[\|\mathbf{y} - G(\mathbf{x})\|_2]$ here to further improve the objective function according to [39] in which λ_{L2} controls its weight. Introducing L2-norm into the loss function Skip connections improves the quality of generated *stress maps* which will also be discussed in detail in Section 2.4.1.

In both (2.4) and (2.5), we adopted the Wasserstein distance as the measure of difference between distributions of real and fake *stress maps.* Compared to the conventional Jensen–Shannon Divergence, Wasserstein distance provides higher convergence possibility and stability in the training process. The detailed architectures of both generator and discriminator in the proposed *EM-GAN* are illustrated in Fig. 2.4.

2.3 EMGraph: Graph Neural Network-Based EM Stress Solver

2.3.1 Problem formulation

This work aims to predict the transient hydrostatic stress over time on general multi-segment interconnects using GCN. The current densities for each branch can be calculated using an IR drop solver, such as SPICE circuit simulator. Besides current density, the width and length of each branch also impact the EM stress. Thus, the input features



(a)

(b)

Figure 2.4: The architecture of the neural networks in the proposed EM-GAN: (a) generator (b) discriminator.

Table 2.1: Input and output for GCN model

	Features	Type	Definition
input	J	edge	current density (A/m^2)
	L	edge	length (μm)
	W	edge	width (μm)
	t	edge/node	time (s)
output	σ	edge/node	stress (Pa)

include current density, width, length, and time. The output feature is the hydrostatic stress. Table 2.1 lists all the inputs and outputs of the GCN model.

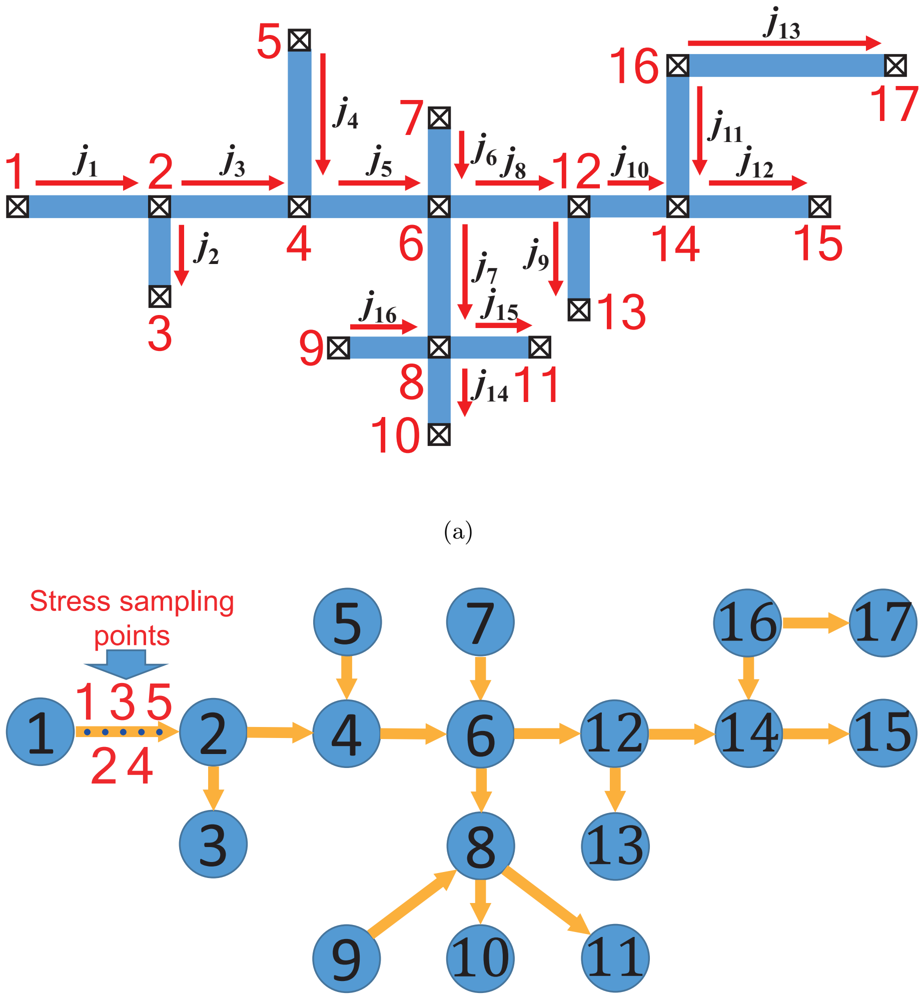
The general multi-segment interconnect can be naturally viewed as a graph, as shown in Fig. 2.5. Fig. 2.5(a) shows an interconnect tree extracted from a real power delivery network (PDN) where the current has direction. Each junction and branch can be represented by a node and an edge in a graph, respectively. To describe the direction of current, a directed graph is employed to represent the tree structure, as shown in Fig. 2.5(b). Then, the embedding features can be mapped into nodes and edges. Therefore, we can obtain a directed graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ that consists of a node set $\mathbf{V}$ and a directed edge set $\mathbf{E}$. The node embedding feature of input is time ($\mathbf{x}_v = [t], v \in \mathbf{V}$). The edge embedding features of input are current density, length, width and time ($\mathbf{x}_{v,u} = [J, L, W, t]^T, (v, u) \in \mathbf{E}$), where J is positive if current flows from v to u and vice versa. The node embedding feature of output is stress ($\mathbf{z}_v = [\sigma], v \in \mathbf{V}$) at node v. The edge embedding features of output are stress at five sampling points ($\mathbf{z}_{v,u} = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]^T, (v, u) \in \mathbf{E}$), as shown in Fig. 2.5(b).

Based on the embedding features of input and output, the graph learning task is a node-edge regression.

To obtain the training set, we implemented an interconnect tree generation algorithm that randomly generates branches with various widths, lengths, and current densities in a fixed area of 256×256 μm^2. The resulting dataset contains 2500 unique designs, and the number of branches ranges from 5 to 110. To obtain the ground truth stress results, the designs are simulated using a finite element-based commercial software COMSOL, and for each design, 10 results at 1st to 10th discrete aging years are produced.

2.3.2 EMGraph architecture

In this section, we focus on developing a GCN model that takes node and edge features as input and output as described in Section 2.3.1. However, there is no GNN model for node-edge regression task. Therefore, we proposed our own GCN model, which is called *EMGraph*, for EM stress assessment. The primary challenges are as follows: first, the stress ranges from -2×10^9 to 2×10^9 Pa. It is difficult for a neural network to predict the entire range, which spans 10 orders of magnitude; second, the edge is directed, and the output has both node and edge features. The GCN model should be complex enough to deal with this graph; third, the accuracy of the stress prediction should be high. However, the regression using GCN has low accuracy.



(a)

(b)

Figure 2.5: (a) Schematic and (b) directed graph of a multi-segment interconnect.

Data rescaling

It is commonly accepted that values around zero are numerically more stable for neural networks. Thus, we rescale all input and output features to -1 to 1 using the min-max normalization method. However, considering the large range of the output stress values (4×10^9 Pa), such normalization squeezes values with fewer orders of magnitude into a small range around zero. This may lead to more accurate predictions at large stress points but may impact the accuracy at the smaller ones as they may be considered noise by the model. This concern is verified by our experimental results in Sec. 2.4.2. However, such configuration is actually in favor of our goal since the large stress points are the ones that may lead to reliability issues and require higher accuracy, while the smaller ones are less important and can be ignored. This is further justified as hydrostatic stress typically will exceed the critical stress before a void is nucleated [84].

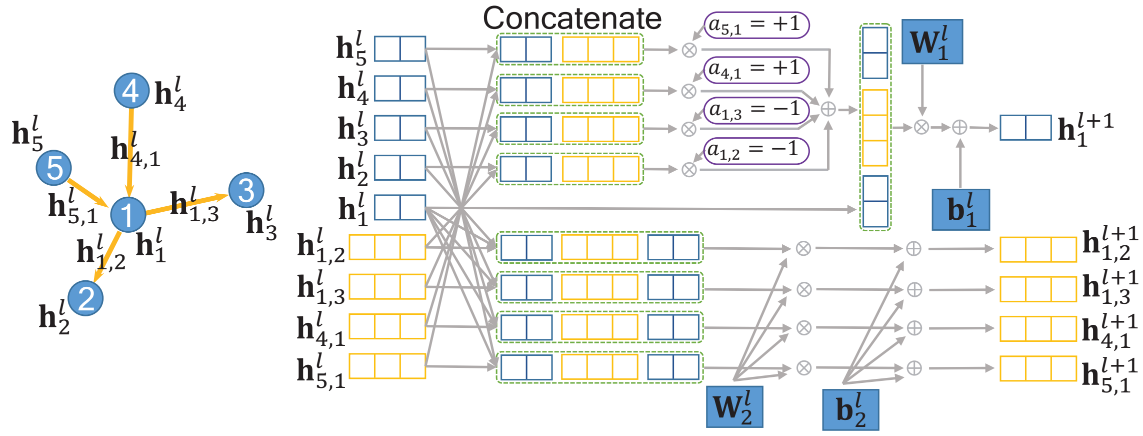
Graph convolution network

We propose an *EMGraph* architecture based on the GraphSAGE network [32] since the GraphSAGE only works for the node classification task. The input layer of *EMGraph* is represented by

$$\mathbf{h}_v^0 = \mathbf{x}_v \text{ and } \mathbf{h}_{v,u}^0 = \mathbf{x}_{v,u} \tag{2.6}$$

where $\mathbf{h}_v^0$ and $\mathbf{h}_{v,u}^0$ are node and edge hidden embedding features of the 0th layer, respectively. The lth hidden layer of *EMGraph* is given by

$$\mathbf{h}_v^{l+1} = \mathrm{ReLU}(\mathbf{b}_1^l + \mathbf{W}_1^l(\mathbf{h}_v^l || \sum_{u \in N(v)} a_{v,u}(\mathbf{h}_u^l || \mathbf{h}_{v,u}^l))) \tag{2.7}$$



(a)

(b)

Figure 2.6: (a) One node and four neighborhood nodes. Both nodes and edges are embedded with features. Each edge has the direction. (b) One hidden layer node-edge embedding update of EMGraph. The convolution operation is to aggregate the information of its neighborhood nodes and connected edges into one node. In the meantime, a convolution also aggregates the information of two end nodes into one edge.

$$\mathbf{h}_{v,u}^{l+1} = \text{ReLU}(\mathbf{b}_2^l + \mathbf{W}_2^l(\mathbf{h}_v^l||\mathbf{h}_u^l||\mathbf{h}_{v,u}^l)) \tag{2.8}$$

where ReLU$(\cdot)$ is an activation function, $a_{v,u}$ is a known parameter representing the direction of edge, $N(v)$ is the set of neighborhood nodes of the node v, $||$ denotes concatenation, $\mathbf{h}_v^l$ and $\mathbf{h}_{v,u}^l$ are node and edge hidden embedding features of the lth layer, $\mathbf{W}^l$ and $\mathbf{b}^l$ are learnable weights and biases, respectively. Fig. 2.6 provides an example of one hidden layer node-edge embedding update for *EMGraph*. The edge features can impact the node features. In turn, the node features can also influence the edge features. The convolution of *EMGraph* consists of two parts: one is to aggregate the information of its neighborhood nodes and connected edges into one node, and another is to aggregate the information of two end nodes into one edge. Concatenation is similar to the "skip connections" in different layers and is also employed to consider both node and edge features. Therefore, *EMGraph*

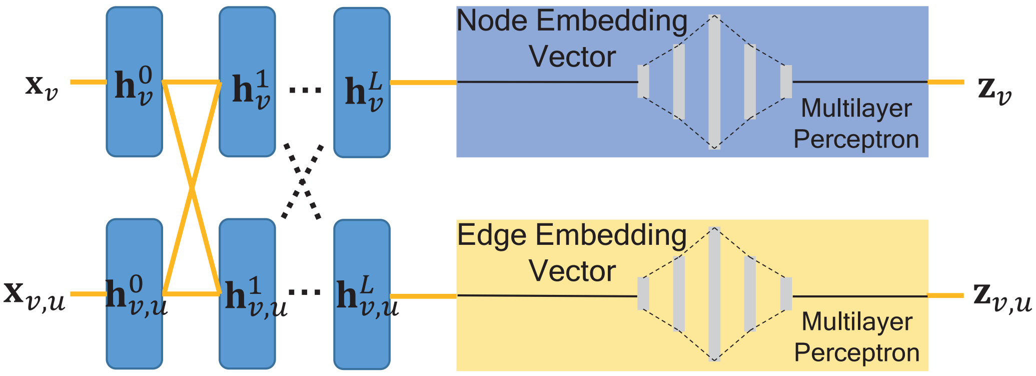


Figure 2.7: Framework of EMGraph with Multilayer perceptron network.

can do the node-edge regression task. To represent directed edge, we introduce a parameter $a_{v,u}$, which is 1 for the inward direction and -1 for the outward direction at the node, as shown in Fig. 2.6. The output layer of *EMGraph* is expressed as

$$\mathbf{z}_v = \mathbf{b}_1^L + \mathbf{W}_1^L(\mathbf{h}_v^L || \sum_{u \in N(v)} a_{v,u}(\mathbf{h}_u^L || \mathbf{h}_{v,u}^L)) \tag{2.9}$$

$$\mathbf{z}_{v,u} = \mathbf{b}_2^L + \mathbf{W}_2^L(\mathbf{h}_v^L || \mathbf{h}_u^L || \mathbf{h}_{v,u}^L)) \tag{2.10}$$

Node and edge decoder

To improve the modeling capacity of the *EMGraph*, we feed node and edge features of the output in Section 2.3.2 to node and edge decoders which are two separate multilayer perceptron (MLP) networks, as shown in Fig. 2.7. The GCN model, which is the first part of *EMGraph*, is only responsible for graph embedding which converts the input graph into latent edge and node features. These features are trained to extract and contain important neighboring information for stress prediction, which are then utilized by node and edge decoders to infer the stress values on each branch.

We propose such an architecture based on the observation that after a certain point, increasing the number of hidden units and hidden layers in the GCN model does not help much in improving the output stress accuracy. Due to the small size of GCN, modeling capacity needs to be increased to further improve accuracy. Therefore, we employ another way to increase the number of learnable parameters by combining GCN and MLP networks. The MLP network can further process the information as it learns node and edge features separately.

2.4 Experimental Results and Discussions

2.4.1 EM-GAN accuracy and performance

In this section, we present the experimental results demonstrating both the accuracy and speed of our proposed *EM-GAN* model for time-dependent EM stress estimation.

Our model is implemented in Python using the TensorFlow (1.14.0) library [3], which is an open-source machine learning platform. As detailed in Section 2.4.1, our dataset consists of 2500 pairs of (1 *design* → 10 *stress maps*). To train *EM-GAN*, a random selection of 15% of the dataset is set aside for testing purposes, while the remaining 85% forms the training set. To fit the dataset to the input layer of *EM-GAN*, we reorganize each 1-to-10 data pair into 10 samples of (*design* with aging year → *stress map*). During the training process, all samples are randomly permuted at the beginning of every epoch.

We run the training for 15 epochs on a Linux server with 2 Xeon E5-2698v2 2.3GHz processors and Nvidia Titan X GPU. The cudnn library is used to accelerate the training process on GPU. To employ mini-batch stochastic gradient descent (SGD), we set the batch size to 8 and use the RMSProp optimizer to solve it. The learning rate of the optimizer is 0.0001, where the decay, momentum, and epsilon parameters are set to 0.9, 0, and 10^{-10}, respectively. The weight of the L2-norm distance λ_{L2} is set to 100.

Accuracy of EM stress map estimation

Once the *EM-GAN* model is trained, only the generator part is preserved, which serves as the generative model. It can take any multi-segment interconnects topology with current densities as input and produce an estimated *stress map* at the specified aging year. To evaluate the estimation error of the trained model, we compare the estimated *stress maps* against the real ones, which serve as ground truth here. We employ the root-mean-square error (RMSE) and the normalized RMSE (NRMSE) given in (2.11) and (2.12) as the metrics of error.

$$RMSE = \sqrt{\frac{\sum_{(x,y)\in S}[\sigma(x,y) - \sigma'(x,y)]^2}{|S|}} \tag{2.11}$$

$$NRMSE = \frac{RMSE}{\sigma_{max} - \sigma_{min}} \tag{2.12}$$

where σ and σ' are the real and generated *stress map*, respectively. S is the set containing all pixels on the interconnects, and $|S|$ denotes the number of elements in S. σ_{max} and σ_{min} are the maximum and minimum stress values in the ground truth *stress map*, respectively.

The accuracy evaluation is conducted on the test set with 375 *designs* that were set aside during the training process. The random generation of *designs* in both the training and test set guarantees that there is no overlap of either topology or current densities between these two datasets. It means that all samples used for evaluation are unseen and completely new to the trained *EM-GAN*, which makes the testing closer to real applications. When *EM-GAN* is employed to estimate stress distribution for a real *design*, it is merely possible that the given topology or current density is identical to any *design* from the training set. The model has to extrapolate what it learned from the training set to unseen cases, which is exactly what we are testing in this evaluation experiment.

For each of the 375 *designs* used for testing, it is fed into the generator of *EM-GAN* together with 10 scalars representing 1st to 10th aging years, and the results of which are 10 *stress maps* showing the evolution of EM-induced stress distribution. Comparing all 3750 generated *stress maps* against their corresponding ground truth (real *stress maps* derived from COMSOL), *EM-GAN* achieves an average RMSE of 0.13 GPa and NRMSE of 6.6%. Considering the large numerical range (usually several GPa and 4 GPa in this work) that typical EM stresses vary in, such accuracy is more than enough for EM failure assessment, such as critical wire identification. Some testing results are visualized in Fig. 2.10. Two *designs* are randomly picked from the test set, and the estimated *stress maps* at 1st, 4th, 7th and 10th aging years are shown together with the ground truth for comparison.

Speed of inference

In what follows, we present a comparison of the speed of our *EM-GAN* with the state-of-the-art work [16] for EM stress analysis. We formulated the problem as a large multi-segment interconnects design that can be partitioned into 528 smaller *designs* of dimensions 256×256 μm^2. We generated the designs randomly using the same method as the one used to generate the training dataset. The number of interconnect branches in

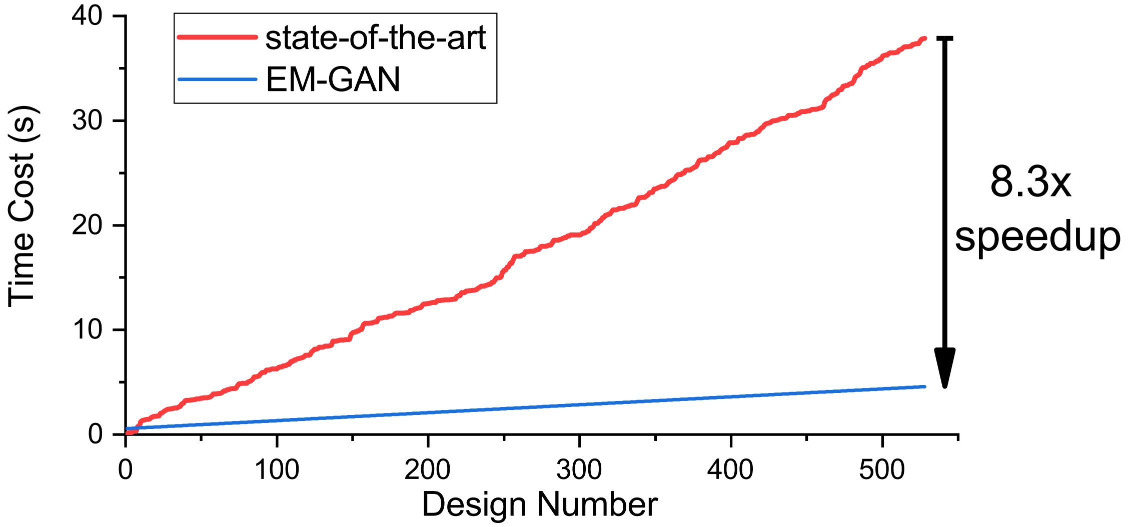


Figure 2.8: Comparison of EM stress estimation speed between state-of-the-art and EM-GAN.

each design varied from 5 to 79. Both *EM-GAN* and the [16] method were run to estimate the EM stress distribution for all 528 *designs* at the 10th aging year. The tests were run on the same server and the accumulated time cost for all 528 *designs* are plotted in Fig. 2.8.

Although [16] produced more accurate results that were consistent with the solution obtained from COMSOL, our *EM-GAN* showed an 8.3× speedup over [16]. The total time cost of *EM-GAN* and [16] was 37.86s and 4.58s, respectively. For [16], the time cost of predicting each *stress map* varied between 0.49s and 0.003s depending on the number of branches involved in the input *design*. However, for our *EM-GAN*, any given *design* is taken as a whole image with same dimensions. The inference process is essentially an image transformation process that deals with a fixed number of pixels, regardless of the number of wire segments involved in the input *design*. Therefore, the inference time of our *EM-GAN* is invariant to the varying number of interconnect branches, making it much more efficient in estimating the EM stress for large-scale designs and exhibiting better scalability.

Analysis of loss and skip connections

As described in Section 2.2.3, *EM-GAN* employs skip connections in the generator to bypass the bottleneck layer and convey geometric information directly from the encoder to the decoder. Another technique we used to improve estimation accuracy is adding L2-norm error in the loss function of generator. To analyze whether and how these modifications help to improve results, we trained two modified *EM-GAN* models. We kept most of the architecture the same as *EM-GAN* in both modified models. The only exception is that one model removed the L2-norm from the objective function, and the other model discarded all skip connections.

Both modified models were trained for 15 epochs on the same server and tested using the same test set as above. The results showed that both modified models suffered from degradation in output accuracy. Specifically, the model without L2-norm loss reached an average NRMSE of 8.4%, and the error was even worse at 15.2% for the other model with no skip connections. Additionally, compared to the modified models, *EM-GAN* performed better in terms of standard deviation, maximum, and minimum errors, as shown in Table 2.2. In Fig. 2.9, we randomly pick one *design* and show the inference results generated by all three models along with their corresponding ground truth for comparison.

We first analyze the influence of skip connections. As shown in the results above, models with skip connections outperformed the one without it by a significant margin. Employing a conventional encoder-decoder architecture means that the network has to

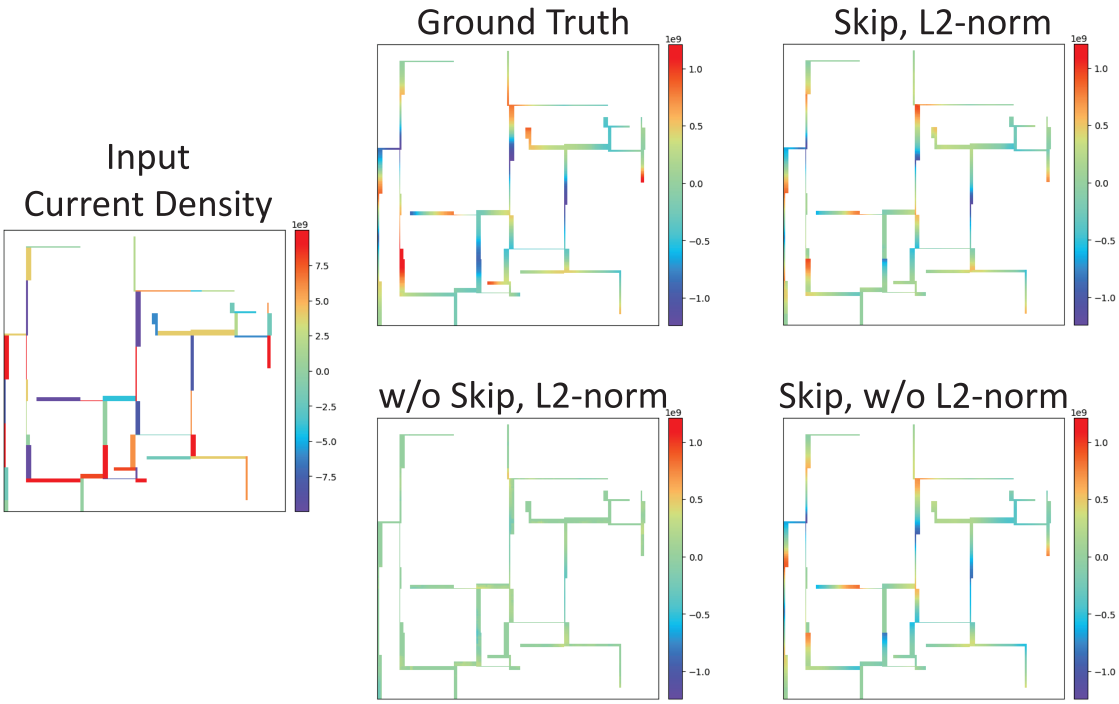


Figure 2.9: Comparison of inference results between different models and the ground truth.

process both geometric and physics information from the input. This is unnecessary in this work since the input *design* and the output *stress map* share exactly the same geometric information, i.e. the interconnects topology. The extra work added to the network occupies both computational and spatial resources that could have been used for extracting more meaningful latent features. The skip connections mitigate this problem by introducing shortcuts for the topology information to be directly passed from the input side to the output side. It alleviates the workload of the main network and spares more bandwidth for latent information flow, which then helps increase the output accuracy.

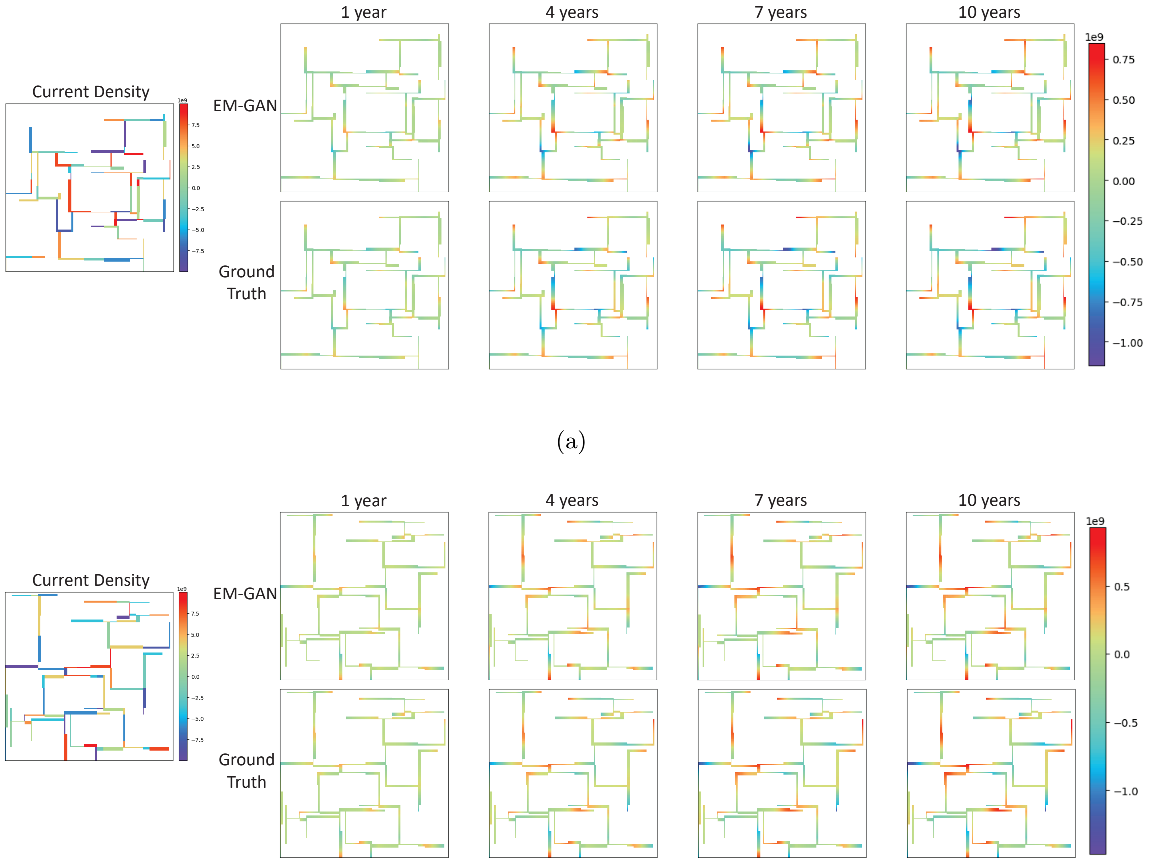
The influence of L2-norm loss on the result accuracy is not as significant as that of skip connection, but its removal still leads to a degradation of the NRMSE from 6.6%

Table 2.2: Statistics of NRMSE for EM-GAN and modified models on testing set.

Metrics	EM-GAN (Skip, L2-norm)	w/o Skip, L2-norm	Skip, w/o L2-norm
Mean	6.6%	15.2%	8.4%
Standard Deviation	1.2%	2.1%	2.1%
Max	12.9%	24.6%	18.4%
Min	3.1%	9.8%	3.8%

to 8.4%. As shown in Fig. 2.9, the *stress map* generated by *EM-GAN* is slightly closer to the ground truth than the model without L2-norm. Besides improving result accuracy, a more significant impact the L2-norm brings to *EM-GAN* is actually the speedup of training process. The modified model without L2-norm converges much slower than *EM-GAN*. This is reasonable since L2-norm is manually added to the objective function as prior knowledge from humans. It helps to guide the training process towards a partially defined target, especially at the early stages of the training process.

The loss function is two folds, one is dynamically determined by the other part of the model itself, i.e., the discriminator, and the other is a predefined goal, i.e., the L2-norm distance. At the very early stages of the training process, when both discriminator and generator are not well-trained yet, using the loss defined by the discriminator to guide the training is more like a random walk. The model with L2-norm converges much faster in the training process and is always closer to the ground truth than the one without L2-norm.



(a)

(b)

Figure 2.10: Comparing the ground truth EM stress distribution with EM-GAN generated ones using two different designs.

It is reasonable to say that the L2-norm helps the model as prior knowledge. At the very beginning of training process, both discriminator and generator are not well-trained and the discriminator cannot provide useful guidance to the generator. This is where L2-norm can complement the discriminator and provide the generator with a meaningful learning direction. In our experiment, adding the L2-norm accelerates the convergence speed by 2× and also leads to better inference accuracy.

2.4.2 EMGraph accuracy and performance

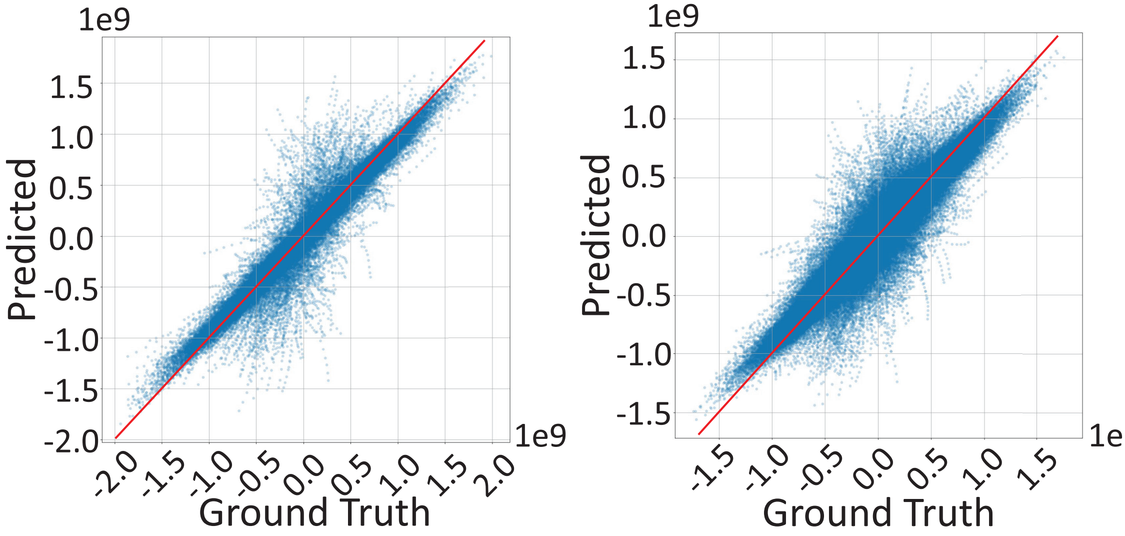
In this section, we present the accuracy and speed of *EMGraph* models on our randomly generated dataset, which comprises 2500 circuit designs. The dataset is divided into a training set of 2125 samples and a test set of 375 samples selected randomly.

The *EMGraph* model is implemented in Deep Graph Library (DGL) [91], which is developed for deep learning on graph and built on PyTorch. For the GCN part, we utilized 5 layers with number of hidden features set to 8, 16, 32, 64 and 128, respectively. For the node and edge decoders, the fully connected layers were set to [128, 256, 1024, 256, 64, 1] and [128, 256, 1024, 256, 64, 5] separately. The model was trained and tested on a Linux server with 2 Xeon E5-2699v4 2.2 GHz processors and an Nvidia Titan RTX GPU. The training batch size was set to 32, and the learning rate of the Adam optimizer was set to 10^{-4}. The cross validation technique was employed, and the model was trained for 80 epochs in 2 hours.

Accuracy of EM stress prediction

Fig. 2.11(a) and Fig. 2.11(b) show the predicted stresses versus the ground truth of all 223,380 nodes and 1,114,350 edges in the test set. The results were obtained using the trained *EMGraph* to predict EM-induced stresses on all 375 test cases, which were never seen by the model during the training process. For each case, 10 predictions for the 1st to 10th aging year were conducted.

As shown in Fig. 2.11, the stress values vary in a large numerical range from -2×10^9 to 2×10^9 Pa. For both nodes and edges, there are more outliers in the range



(a)

(b)

Figure 2.11: EMGraph prediction vs ground truth on all testing cases for (a) nodes and (b) edges.

around zero, while the results tend to be more accurate at both ends of the full range. This is acceptable and indeed what we desired, since the large stresses are the ones that may lead to reliability issues and require higher accuracy.

For better illustration and comparison, we convert all prediction results into stress maps, in which stress values are filled into the interconnects topology and shown in colors. The RMSE between the predicted stress map and its ground truth is employed to evaluate the result accuracy.

EMGraph yields RMSE values ranging from 1.6×10^7 to 1.8×10^8 Pa on the test set and achieves an averaged RMSE of 6×10^7 Pa, which translates to a 1.5% error rate considering the full stress value range of 4×10^9 Pa.

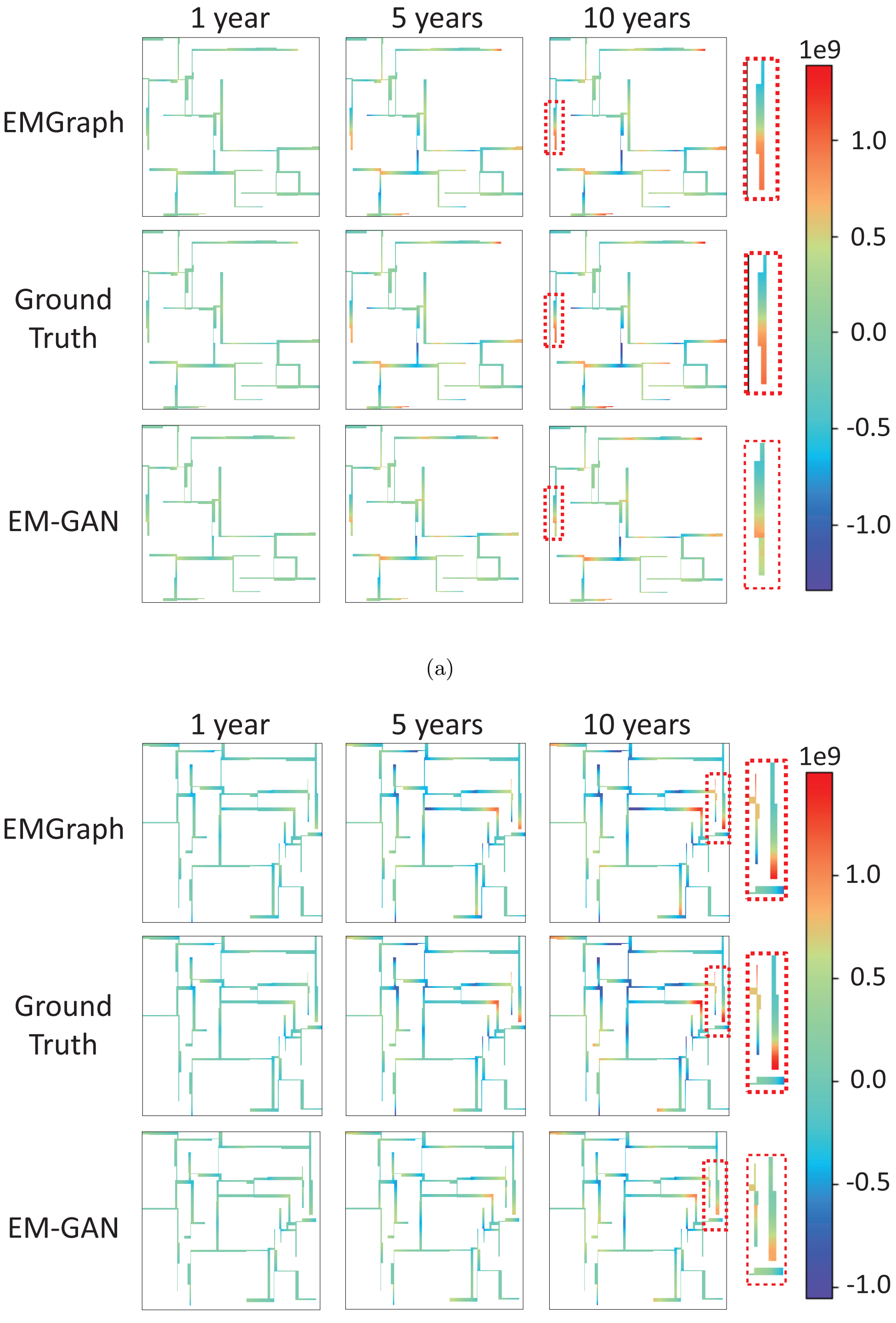
Fig. 2.12 shows the stress maps of both the best and worst cases (in terms of averaged RMSE) predicted by *EMGraph*. The results of *EM-GAN* [42] and the ground truth obtained from COMSOL are also shown in parallel for comparison. As the EM-induced stress is a time-varying process, for each case, we show the results at the 1st, 5th, and 10th aging year for a better illustration of the stress evolution.

As shown in Fig. 2.12, *EMGraph* yields much better accuracy in both cases, with 7× better in the best case and 2.5× better in the worst case. Although the RMSE of the worst case is 11× larger than that of the best case, the predicted stress map for the worst case is still quite accurate and closer to the ground truth than the result of *EM-GAN*. In comparison with *EM-GAN*, *EMGraph* demonstrates much better accuracy in both cases, with 7× better in the best case and 2.5× better in the worst case. More statistics on the comparison between *EM-GAN* and *EMGraph* are listed in Table 2.3.

Speed of inference

The training process of the *EMGraph* takes 2 hours, and the trained model consists of a 441KB GNN, a 2252KB edge decoder, and a 2251KB node decoder. The lightweight model, combined with the highly parallelizable nature of the graph input, makes *EMGraph* have the potential to yield fast inference speed. In what follows, we compare the speed performance of *EMGraph*, *EM-GAN*, and the state-of-the-art work [16], which is a separation-of-variables-based analytical method.

As shown in Table 2.3, the average inference speed of *EMGraph* for each case is only 0.27ms, which is 14× and 265× faster than the 3.8ms and 71.7ms inference speeds yielded by *EM-GAN* and work [16], respectively. These statistics were obtained by running three methods on all test cases and taking the average of the time cost for each case.



(a)

(b)

Figure 2.12: Comparison of the predicted stress maps obtained by EMGraph and EM-GAN on (a) best case and (b) worst case.

Table 2.3: Accuracy and speed comparison

Metrics	EMGraph	EM-GAN	State-of-the-art
Max RMSE	1.8×10^8 Pa	5.2×10^8 Pa	Close to ground truth
Min RMSE	1.6×10^7 Pa	1.2×10^8 Pa	
Mean RMSE	6×10^7 Pa	2.6×10^8 Pa	
Mean Error Rate	1.5%	6.6%	
Inference Speed	0.27ms	3.8ms	71.7ms

Although work [16] yields higher accuracy, *EMGraph* achieves a two orders of magnitude speedup while still rendering comparable results in accuracy. Moreover, as *EMGraph* treats each graph input as individual nodes and edges that can be processed in parallel, it has the potential to achieve even more significant speedups on large designs.

Scalability on large unseen designs

In this section, we further test the scalability of the trained *EMGraph* model on 13 large designs that are randomly generated without any limitations on their dimensions. Although we trained *EMGraph* on the dataset with a fixed size of 256×256 μm^2, we note that *EMGraph* is not limited to a certain size, in contrast to *EM-GAN*, which is only applicable to the size it was trained on. We fix the size of the dataset in this work just to make a fair apple-to-apple comparison between two models. The scalability of *EM-GAN* is limited due to its image-processing-based nature, and the cost of its forward propagation

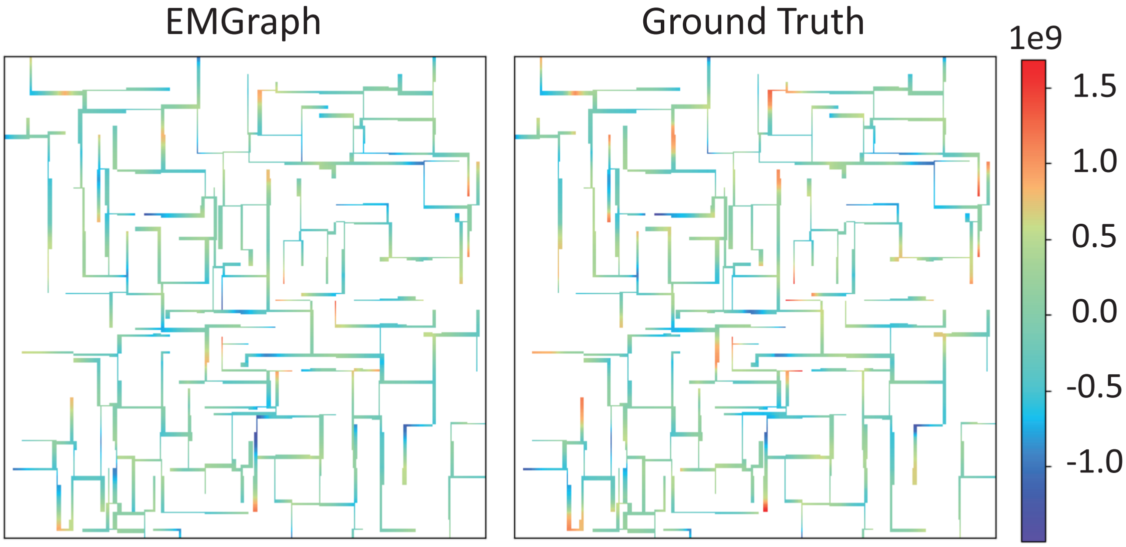


Figure 2.13: EMGraph prediction vs ground truth on large design with 401 branches.

becomes exponentially large as the input size increases, which is not the case for *EMGraph*. The inference cost of *EMGraph* is linearly dependent on the number of branches in its input graph, and such calculations are highly suitable for parallelization, which further boosts its scalability to large designs.

Fig. 2.13 shows the stress map predicted by *EMGraph* for the largest design with 401 branches at the 10th aging year. *EMGraph* maintains its high accuracy on the large designs and achieves an average RMSE of 1.1×10^8 Pa across all 13 large designs, with a minimum of 8×10^7 Pa and a maximum of 1.6×10^8 Pa. The number of branches in these large designs ranges from 113 to 401, which is much larger than the cases in the previous test set. We only compare the results with ground truth here, as *EM-GAN* is not applicable to such large designs.

Although the accuracy on large cases slightly degenerates compared to the previous test set, considering that these are much more complicated designs and were never seen by the model before, such accuracy is still acceptable. We also recorded the time cost of *EMGraph*, and the average inference speed for each case is only 0.32ms, which is still at the same level as it yields on the smaller cases, thanks to the parallel nature of the nodes and edges in the input graph.

2.5 Summary

The work presented in this chapter introduces two data-driven, learning-based EM analysis methods for multi-segment interconnects. First, we present a GAN-based transient hydrostatic stress analysis model, which is called *EM-GAN*, for EM failure assessment. In this approach, we treat the traditional numerical PDE solving problem as a time-varying 2D-image-to-image problem, where the input is the multi-segment interconnects topology with current densities, and the output is the EM stress distribution in those wire segments at the given aging time. We randomly generated the training set and trained the model with the COMSOL simulation results. Different hyperparameters of GAN were studied and compared. After the training process, *EM-GAN* is tested against 375 unseen multi-segment interconnects designs and achieved high accuracy with an average error of 6.6%. It also showed an 8.3× speedup over recently proposed state-of-the-art analytic-based EM analysis solver. Second, we present a graph neural network-based model, which is called *EMGraph*, for transient EM stress analysis. *EMGraph* performs the node-edge regression task to predict the stress at the wire segment (edge). Compared with the GAN-based image method, *EMGraph* can learn more transferable knowledge to predict stress distributions on new graphs without retraining via inductive learning. Experimental results show the model has a smaller size, better accuracy, and faster speed over *EM-GAN* on several interconnect trees benchmarks. Therefore, *EMGraph* is very powerful and suitable for the transient EM stress assessment.

Chapter 3

Physics-Informed Neural Network-Based Electromigration Analysis

3.1 Related Work and Motivation

3.1.1 Existing numerical approaches for solving PDEs

In order to solve the PDE (2.1), numerous conventional numerical and analytical methods have been proposed to attempt to solve the PDEs efficiently and accurately [22, 80, 106, 93, 16, 83, 77]. Although the numerical methods, such as finite difference method [22, 80, 83] and finite element method (FEM) [106], can work for the complex interconnect structures and obtain EM stress accurately, they require high computational costs due to discretization of space and time. Recently, semi-analytical method based on the separation of variables method has been proposed [93, 16], which shows promising performance in both accuracy and efficiency for general multi-segment interconnects. Furthermore, a very fast analytic approach [77] has been proposed, but it cannot be applied to interconnect line wires..

3.1.2 Learning based approaches for solving PDEs

Recently, machine learning, particularly deep learning based on deep neural networks, has achieved breakthrough success in many cognitive applications, such as image, text, speech, and graph recognition [49, 28]. Inspired by these observations, neural networks have been adapted to solve PDEs [78, 87, 41].

In previous chapter, a generative adversarial network (GAN)-based method, called *EM-GAN*, is proposed to perform fast transient hydrostatic stress analysis by solving Korhonen equations [42]. It achieved an order of magnitude speedup over the efficient analytic-based EM solver with good accuracy. However, this method only works for a fixed region because its output is an image with a fixed size, which restricts its application in real chips. Furthermore, the image is not a natural tool to represent multi-segment interconnects because the region with large areas is filled with nothing. We further proposed an improved GNN-based EM solver, called *EMGraph* [40]. Since GNN represents a more general and natural relationship among different design objectives, knowledge learned by GNN models tends to be more transferable for different designs, which is highly desirable. However, all these methods are still supervised learning approaches that require extensive training from numerical solvers or measured data.

To mitigate this drawback, a recently proposed unsupervised learning framework, called *physics-informed neural networks* [68, 67], PINN or *physics-constrained neural networks* [78, 61], has been introduced. The key concept is to frame the process of solving PDEs into a nonlinear optimization process using DNN with loss functions to enforce the physics laws represented by the PDE and boundary conditions. However, only very simple PDE problems were demonstrated in [68, 97, 78, 10, 67], although some progresses were made for more complicated aerodynamics simulation recently [19]. Recently, a PINN-based approach for EM analysis has been proposed [35]. The method attempts to improve the PINN method to better handle the temperature-dependent diffusivities for metal atom migrations. It tries to add more neurons representing some pre-determined allocation points and time instances into the neural networks. This method slightly improves the plain PINN method by achieving better training accuracy at the cost of longer training time under the same number of neurons.

3.2 HierPINN: Hierarchical Physics Informed Neural Network

In this section, we present the new hierarchical PINN solving strategy, which takes multi-segment interconnect tree as input and predicts the EM-induced stress for arbitrary locations in the interconnects at any given aging time. The proposed model solves the stress evolution equations in a hierarchical way, which consists of two levels (stages). The first stage, or the lower level, takes only a single-segment straight wire as input and predicts the stress inside and at both ends of the given wire. The first stage can be viewed as implicitly enforcing the physics laws related to the stress evolution inside a segment wire and leaving the boundaries as the input and output variables to be used in the second stage. The second stage, or the upper level, takes all internal junctions or boundaries as inputs and matches the stress predicted by the lower level at each junction to meet the boundary conditions among adjacent wires (i.e., stress continuity and atom flux conservation) in the original PDE using the PINN optimization framework. Each level employs a multilayer perceptron (MLP) network with different configurations as the backbone. The rest of this section will introduce both levels in detail together with the data preparation procedures.

3.2.1 Lower level: single-segment straight wire stress predictor

The lower level of the proposed hierarchical PINN is a stress predictor/solver which takes single-segment straight wire as input and predicts the EM-induced stress for any location on the wire at a given aging time instant. We note that multi-segment interconnects always consist of many wire segments with different widths and lengths, stress currents, and atomic fluxes at the two terminals. For one wire segment, once the geometrical parameters, current density, and boundary conditions are given, EM-induced stresses are determined at all locations, including terminals for a given time instant. For one wire segment under those parameterized conditions, one way to obtain the fast and compact model is by means of DNN networks via supervised learning. The whole process is illustrated in Fig. 3.1: The backbone of the stress predictor is a multilayer perceptron network with 7 layers. The input layer has 7 neurons corresponding to the location, aging time, stress current, and geometrical parameters of a straight wire. The input vector is called the wire feature vector

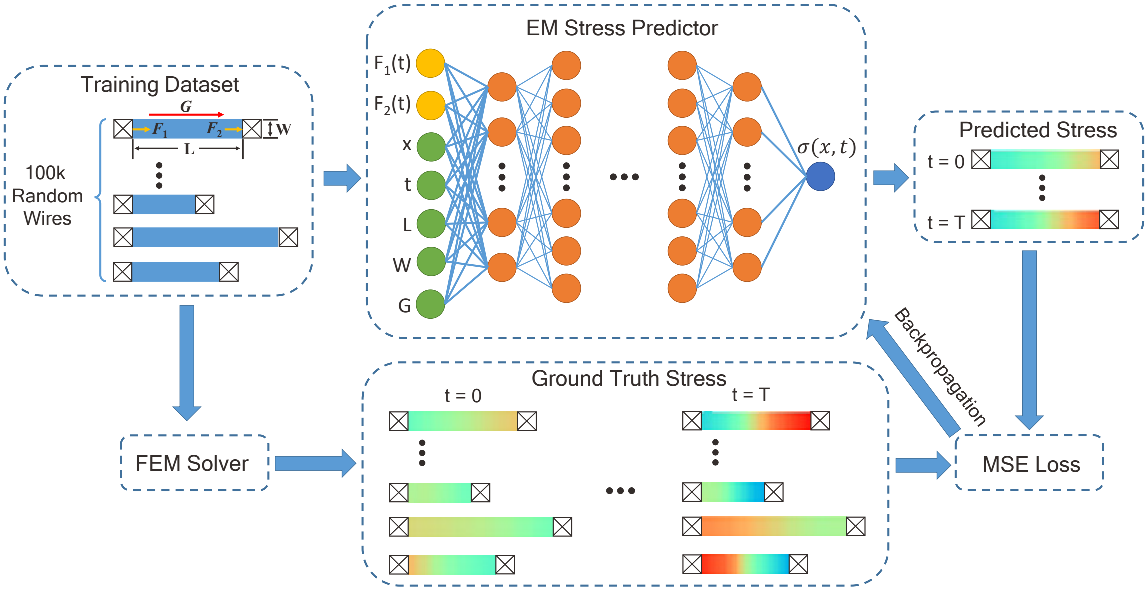


Figure 3.1: Framework of proposed stress predictor in the first stage.

denoted as u. As shown in Fig. 3.1, we use L, W, and G to denote the length, width, and driving force of the input wire. F_1 and F_2 are the atom fluxes at the left and right ends of the wire segment. x and t are the location and aging time at which the stress is to be predicted. For any input wire, the positive direction of location, driving force, and atom flux is always pointing from left to right, such that negative values are allowed to represent the opposite direction. The output layer of the stress predictor has only a single neuron with no non-linear activation function attached. The scalar value of this output neuron indicates the predicted stress at the input location x and aging time t.

For any given combination of wire geometries, driving force, and boundary conditions at both ends of a wire, the stress evolution is uniquely determined according to the EM stress PDE. With that being said, the task of the proposed stress predictor is to solve the PDE in a single-wire case. The boundary conditions at the left and right ends of the wire are defined by F_1 and F_2, respectively, while the rest of the parameters in the stress evolution equation are set by the other 5 input neurons. The input neurons are then processed through layers of non-linear forward propagation operations, and the final output neuron is a scalar value indicating the predicted stress. In this process, the stress predictor serves as an approximation to the single-wire EM stress solver. The hidden layers inside the MLP learn to capture the governing physics laws and convert the input features into stress results.

We want to stress that for practical use of this method, the first stage needs to be trained *ONLY ONCE*. Then it can be used for different multi-segment wires with different numbers of wires and topologies and stressing current density conditions. This is a real benefit of our approach as the training cost of this stage can be ignored for sufficient applications of this method. Second, our method is open to other solving solutions at the first stage (such as analytic solutions, fast numerical solutions). Third, we can pursue more accurate DNN modeling even at high computing costs.

Regarding the accuracy, the accuracy of the stress predictor as an approximated PDE solver is guaranteed in two senses. First, we limit the stress predictor to work only on simple single-wire cases instead of overcomplicated interconnect trees. Although a multilayer perceptron, as a universal approximator, can theoretically approximate any complicated non-linear solver, it is not practical to implement it in real cases with limited resources and strict speed requirements. Thus, such limitation substantially reduces the complexity of the problem given to the model, which makes it possible to achieve both high accuracy and performance.

Secondly, the stress predictor is trained on a large dataset consisting of 80k random wires, as shown on the left-hand side in Fig. 3.1. To obtain abundant high-quality training data, we generated 80k single straight wires and randomly assigned length, width, and current density to each wire. These wires were then randomly connected together to create interconnect trees. Due to the randomness in both the generation and connection procedures, the resulting interconnect trees are also completely random with various topologies. These interconnects were then passed into COMSOL, which is a commercial FEM solver, to do the EM stress simulation. The COMSOL-simulated stress evolutions in every single wire, as illustrated at the bottom of Fig. 3.1, were saved as ground truth results. During the FEM simulation process, the time-variant atom fluxes at both ends of each wire ($F_1(t)$ and $F_2(t)$) were recorded and then concatenated with wire geometries to serve as input features in the training dataset. Again, such a one-time training cost does not add significant overall computing cost to the final *HierPINN-EM* solution, as mentioned earlier.

To verify the accuracy of the proposed stress predictor, we generated another 20k single wires to serve as the test set. These wires were generated using the same methodology as we used in generating the training dataset. These test data were never seen by the model during the training process, and the trained stress predictor showed impressive accuracy on this test set, with details demonstrated in Section 3.3.1.

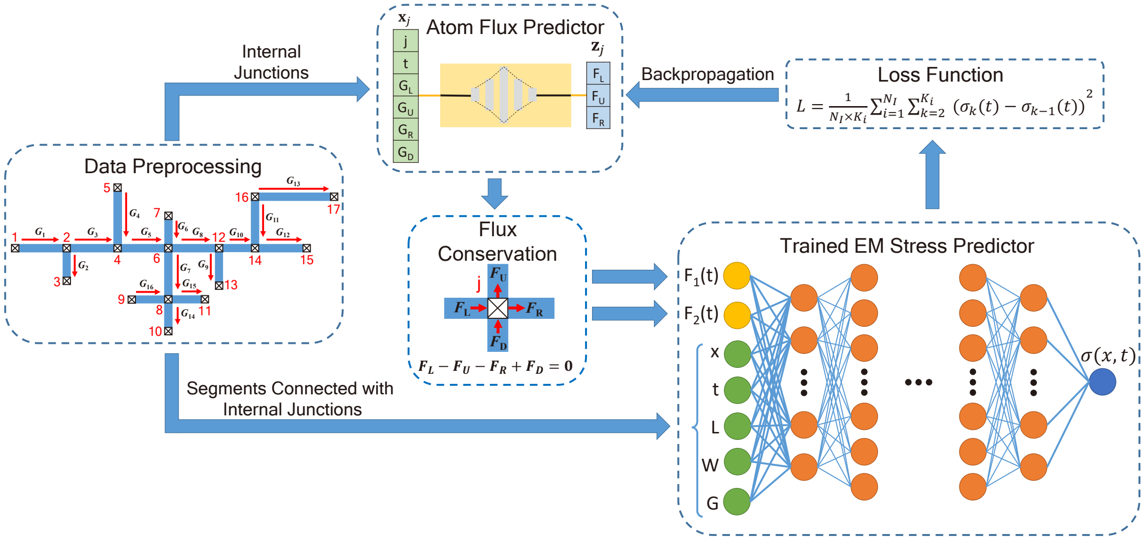


Figure 3.2: Framework of proposed hierarchical PINN based EM Predictor.

3.2.2 Upper level: atom flux predictor for all the wire segments

After the lower-level stress predictor is trained, we can then build the upper level on top of it, as illustrated in Fig. 3.2. Similar to the stress predictor, the backbone of the upper level is still an MLP but with completely different objectives and configurations.

The goal of the upper level is to predict the correct boundary conditions, i.e. atom fluxes, for every single wire in the given interconnect tree so that the EM stress in internal junctions or boundaries are continuous, and EM stress in each wire can be independently derived using the stress predictor already trained in the lower level. To achieve this, we implemented the atom flux predictor using an MLP model with seven layers.

This model takes internal junctions instead of wire segments as input. Therefore, the first stage in the upper level, as illustrated on the left-hand side of Fig. 3.2, is to label all internal junctions in the input interconnect tree starting from 1. Each internal junction is then assigned a feature vector, and the feature vector of the j-th junction is denoted as x_j. The first two entries in the feature vector are the label number j and aging time t. Since each internal junction may have up to four wires connected to it from four directions, i.e. left side, upside, right side, and downside, we appended the driving forces in these four directions (G_L, G_U, G_R, and G_D) to the feature vector, and the driving force would be set to zero if there was no connected wire in the corresponding direction. The resulting feature vectors are then passed into the atom flux predictor.

The output of the atom flux predictor for the j-th internal junction is a vector denoted as z_j. There are three entries in z_j, which correspond to the predicted atom fluxes in the left, up and right directions, respectively. As suggested by the second and third boundary conditions in EM PDE (2.1), the atom fluxes at each internal junction must satisfy the flux conservation law. As a result, we apply the flux conservation law to z_j so that we can calculate the fourth atom flux in the downward direction using the other three predicted results. In this way, the flux conservation law is strictly enforced at every internal junction in the interconnects. The predicted atom fluxes are then filled into the corresponding F_1 and F_2 entries of wire feature vectors u, in which the other entries are directly obtained from the original input interconnects.

One huge advantage of our proposed hierarchical method is that the upper level only needs to be trained at internal junctions instead of the whole interconnects. As a result, the x entries in the wire feature vectors are set to 0 or 1, corresponding to the left/down or right/up end of each wire, respectively. This avoids random sampling inside each wire ($0 < x < 1$), which would otherwise result in a large number of training points and significantly increase the training cost.

Next, the wire feature vectors are passed into the trained stress predictor to obtain the stress results at all internal junctions. For each internal junction at any aging time, there will be two to four predicted stress results depending on the number of wires connected at that junction. According to the first boundary condition in EM PDE (2.1), stress values should be continuous at boundaries, meaning that the predicted stress results should be equal to each other at any internal junction. This leads to the following physics-informed loss function, which we propose to train the atom flux predictor.

$$L = \frac{1}{N_I \times K_i} \sum_{i=1}^{N_I} \sum_{k=2}^{K_i} (\sigma_k(t) - \sigma_{k-1}(t))^2 \tag{3.1}$$

where N_I denotes the number of internal junctions in the interconnects, K_i represents the number of connected wires at the i-th junction, which ranges between 2 and 4. $\sigma_k(t)$ is the k-th predicted stress result for the current junction at aging time t. The loss function is the mean squared error (MSE) of all predicted internal junction stress, which serves as a measurement of stress discontinuity at boundaries. When training the upper level, the lower level stress predictor is fixed, and the loss is only back-propagated back to the atom flux predictor to update the weights and biases in the model.

We note that the proposed hierarchical PINN approach bears some similarity to the domain decomposition method [107] in which hierarchical solving strategies are employed. However, there are several distinguished differences between the two approaches. First, in domain decomposition, the subcircuits are typically obtained by partitioning and have to be solved for each subcircuit every time when the whole circuit is to be solved. While for *HierPINN-EM*, solutions of the single wire, whose boundary or junctions are naturally defined, can be obtained much more efficiently via inference on the DNN network, which only needs to be trained *ONCE*. At the top level, the domain decomposition method tries to solve more dense matrices due to the subcircuit reduction via matrix-solving processes like LU decomposition, while *HierPINN-EM* uses the unsupervised PINN framework to find the solution, which is meshless and easy for design space parameterizations. It uses the differential nature of the DNN model trained in the first stage to guide the backpropagation process in the second stage.

3.3 Experimental Results

In this section, we demonstrate the prediction accuracy, speed, and scalability of the proposed *HierPINN-EM* by testing it on both straight wires and interconnect trees that were randomly generated. Both Atom Flux Predictor and Stress Predictor in *HierPINN-EM* are implemented in Python 3.8.12 with PyTorch 1.7.1. The training and testing of both models were run on a Linux server with 2 Xeon E5-2699v4 2.2 GHz processors and Nvidia TITAN RTX GPU. In the training phase, the Adam optimizer was used to update the model, and the learning rate was set to 10^{-4}. The cross-validation technique was employed in the training process.

3.3.1 Accuracy of lower level on single-segment wires

The lower-level stress predictor serves as the foundation of our proposed hierarchical method. A 7-layer multilayer perceptron with configurations of [7, 256, 512, 1024, 512, 256, 1] is employed as the backbone of the stress predictor. The stress predictor takes a wire feature vector consisting of wire geometries, EM driving force, and boundary conditions as input and outputs the predicted EM-induced stress at a given position and aging time. As discussed in Section 3.2.1, we created a large dataset of 100k single-segment straight wires, using 80k of them for training and reserving the remaining 20k wires for testing. The model was trained for 20 epochs, which took approximately 23 hours.

Fig. 3.3 illustrates the predicted stress versus the ground truth at 2576392 locations for an aging time of 10^6 seconds. These locations were randomly sampled from all 20k wires in the test set, which were never seen by the stress predictor during the training process.

As shown in Fig. 3.3, the predicted stress values all closely align with the red line (slope=1, intercept=0). For all 20k wires in the test set, the trained stress predictor yields a root-mean-square error (RMSE) ranging from 7.5×10^3 to 5.7×10^4 Pa and achieves an average RMSE of 2.7×10^4 Pa. Both the worst and best predicted wire segments are shown in Fig. 3.4.

The predicted results agree very well with the ground truth, even in the worst case. Such accuracy is even more impressive, considering that the ground truth stress values vary in a large range between -5×10^7 and 5×10^7 Pa. By dividing the RMSE with the full stress range (i.e., 10^8 Pa), the worst and mean error rates of the stress predictor can be

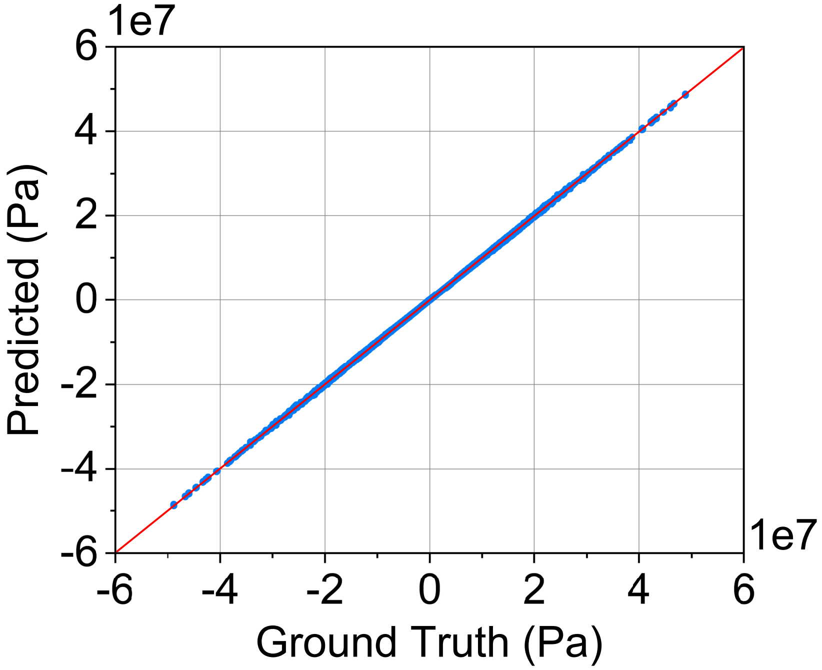
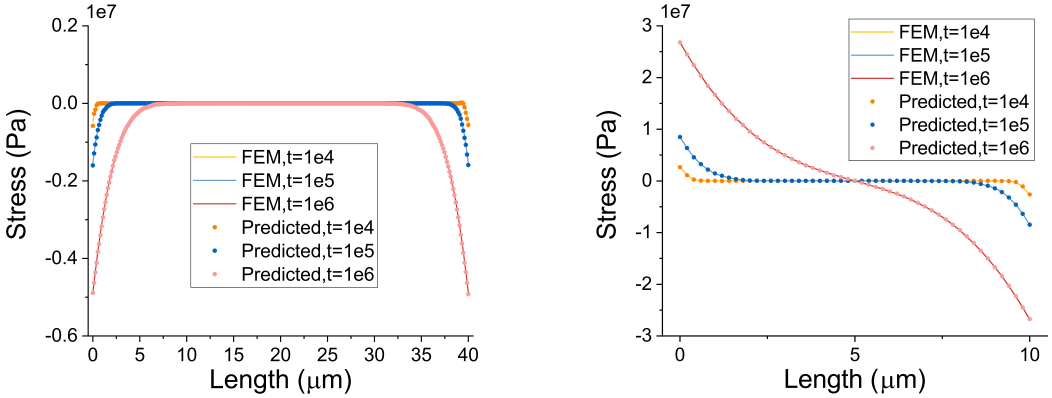


Figure 3.3: Stress predictor vs ground truth on 20k single-segment wires.



(a)

(b)

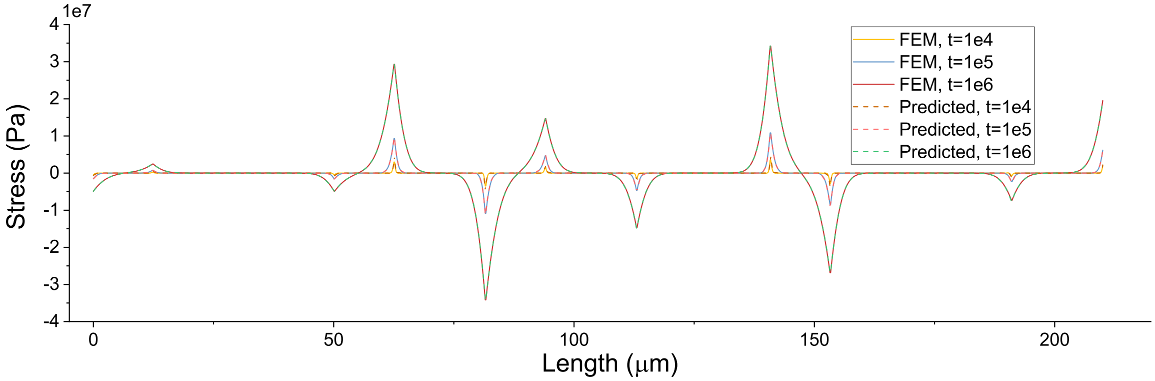
Figure 3.4: Comparison of predicted stress and ground truth in (a) best and (b) worst wire segments.

calculated as 0.008% and 0.057%, respectively. Its low error rate in predicting stress in each segment will serve as the foundation for accurate predictions in larger interconnect trees.

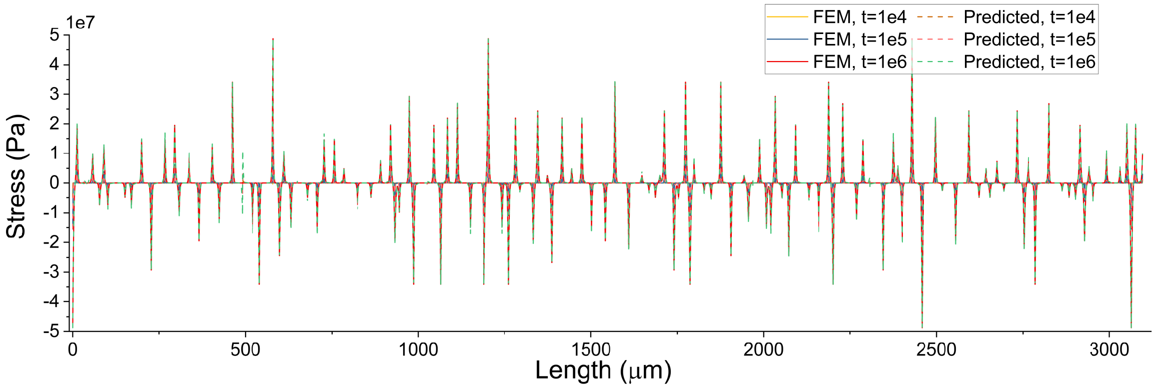
We want to note that this stress predictor only needs to be trained *ONCE* and can then be embedded into the upper level for EM analysis of different interconnect trees with different topologies. As a result, it can be viewed as a library that needs to be built once and can actually be generated using different methods as long as the stress evolution physics laws in a single wire are learned and enforced.

3.3.2 Accuracy of EM stress prediction on straight wires

To verify the accuracy of *HierPINN-EM*, we first test it on 121 randomly generated multi-segment straight wires. All test cases have random numbers of segments ranging from 10 to 130. The stressing current density and geometrical parameters of each segment are also randomly assigned. Fig. 3.5 shows the comparison of EM stress predicted by *HierPINN-EM* and ground truth FEM results simulated by COMSOL. Fig. 3.5(a) shows the results of the smallest case in the test set, which has 10 segments, and Fig. 3.5(b) shows the results of the largest case with 130 segments. To show the evolution of the EM stress as aging time increases, we plotted stress results at three aging time instants (10^4, 10^5, and 10^6 seconds) for each case. The RMSEs of the *HierPINN-EM* predicted stress results are 5.9×10^4 and 2.0×10^5 Pa, respectively, for these two cases. For all 121 test cases, *HierPINN-EM* yields RMSEs ranging from 4.7×10^4 to 4.0×10^5 Pa, and the mean error is 1.9×10^5 Pa, which can be converted to 0.19% mean error rate when taking the full 10^8 Pa stress range into consideration.



(a)



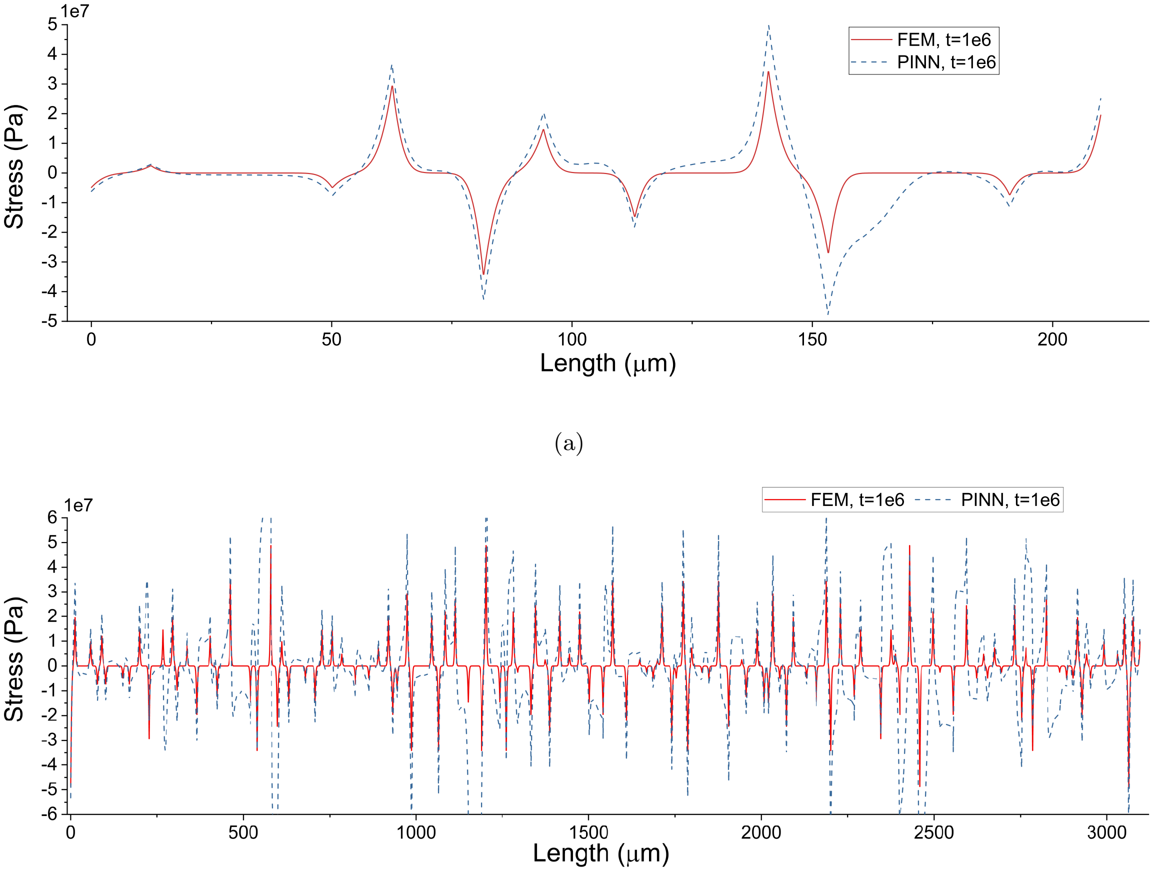
(b)

Figure 3.5: Stress comparisons of (a) a 10-segment and (b) a 130-segment straight wire between *HierPINN-EM* and COMSOL at 3 aging time instants: 1e4, 1e5 and 1e6 seconds.

To compare *HierPINN-EM* with the existing PINN method, we also implemented a plain PINN model using a 7-layer MLP with exactly the same structure as *HierPINN-EM*. Note that we did not compare our method with [35], as this method can be essentially viewed as a plain PINN method with some trade-offs between training accuracy and training time due to more complicated neuron representations.

All equations from (2.1) are formulated into a single physics-informed loss function to train the plain PINN. We tested the plain PINN on the same 10-segment and 130-segment test cases, and the comparison of plain PINN and FEM stress results at aging time of 10^6s is shown in Fig. 3.6. The RMSEs of plain PINN results in these two cases are 6.4×10^6 and 1.6×10^7 Pa separately, which are 107× and 79× worse than that of *HierPINN-EM*. Due to the large number of collocation points required in each segment in the training process of the plain PINN model, the training cost is around 30 minutes on the 10-segment case, and it leaps significantly to around 10 hours on the 130-segment case. As a comparison, the training costs of *HierPINN-EM* are only 5.5s and 39.1s separately in these two cases. The result verifies the great advantage in scalability of our proposed hierarchical method over the plain PINN model.

The physics-informed loss function used to train the plain PINN model contains all PDE equations for all domains, which leads to a complex training process. This means that the PINN model has to be trained simultaneously in all segments and boundaries to minimize every single error in the loss function to satisfy different physics laws. This makes the training process of PINN highly unstable, and successful convergence is not always guaranteed. In contrast, our proposed *HierPINN-EM* overcomes this issue by splitting the



(a)

(b)

Figure 3.6: Stress comparisons of (a) a 10-segment and (b) a 130-segment straight wire between plain PINN and COMSOL at aging time instant 1e6 seconds.

equations into two levels so that the model at each level is separately trained to satisfy a simpler physics law. The lower level is focused on a single segment while the upper level is based on a trained lower level model so that the upper level only has to be trained on a few internal junction points, which significantly alleviates the training load. Once the upper level is trained, the physics laws inside each segment are automatically satisfied thanks to the highly accurate lower level model, which is already shown in Section 3.3.2.

Moreover, different physics laws in a single loss function also require careful consideration in weight assignment to balance the influence of each equation [41]. Such a weight balancing process adds extra overhead to the training process of the plain PINN model and further limits its scalability in large interconnects.

3.3.3 Accuracy of EM stress prediction on interconnect trees

To demonstrate the generalizability of the proposed model, we further test *HierPINN-EM* on 2-D interconnect trees with more complicated topologies. The test set consists of 165 interconnect trees with random numbers of segments ranging from 10 to 105. Similar to the straight wire test set, the stressing current density and geometrical parameters of each segment are also randomly assigned.

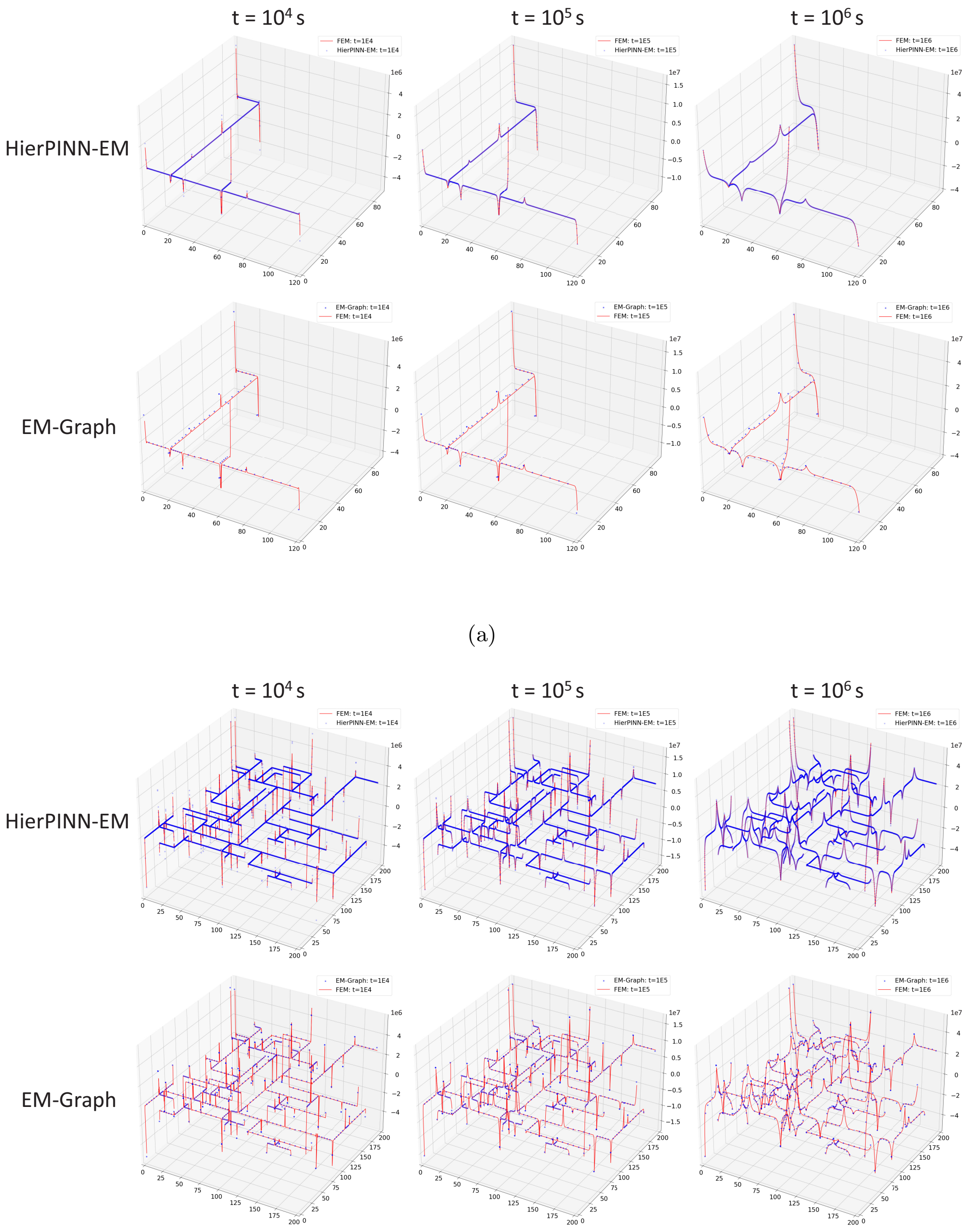
To make an apple-to-apple comparison with GNN-based method *EMGraph*, we convert all prediction results into stress maps, which are generated by projecting the predicted stress results for every single segment onto the interconnects topology and shown in 3-D formats. We show the comparison between *HierPINN-EM*, *EMGraph*, and COMSOL results of the smallest (10 segments) and largest (105 segments) designs from the test set in Fig. 3.7. The evolution of the stress at 3 time instants, i.e., 10^4, 10^5, and 10^6 seconds, is illustrated from left to right in each row.

Comparing the predicted stress maps in Fig. 3.7 with the COMSOL ground truth results, *HierPINN-EM* yields 1.2×10^5 and 3.4×10^5 Pa RMSE for these two cases, while *EMGraph* yields 3.1×10^5 and 3.9×10^5 Pa, which are slightly worse than *HierPINN-EM*. For all 165 interconnect trees in the test set, *HierPINN-EM* achieves better accuracy with a mean RMSE of 2.8×10^5 Pa, while *EMGraph* achieves 3.6×10^5 Pa. Thus, *HierPINN-EM* outperforms *EMGraph* in accuracy with 19% better RMSE when predicting EM-induced stress for 2-D interconnect trees.

Moreover, *EMGraph* sets the number of sampling points in each segment to 5, which leads to a coarse granularity in predicted stress results. This is not a big concern when the lengths of segments are relatively small. However, when the segments get much longer, it may introduce huge errors into the prediction since there are large spaces between 5 sampling points and the interpolations between them become much less reliable. This problem is solved in *HierPINN-EM* as the location input x is a scalar value in float format, which can represent any point in the segment. The granularity of the results can be easily controlled by altering the sampling density of input x. The better inference flexibility gives *HierPINN-EM* more potential in generalizability to larger interconnect trees.

3.3.4 Speed of inference

The training process of *HierPINN-EM* is conducted in stages; the lower level was trained for 23 hours, while the training cost of the upper level varies case by case between



(a)

(b)

Figure 3.7: Stress comparisons of (a) a 10-segment and (b) a 105-segment interconnect tree between *HierPINN-EM* and COMSOL at 3 aging time instants: 1e4, 1e5 and 1e6 seconds.

Table 3.1: Accuracy and speed comparison

Metrics	HierPINN-EM	EMGraph	COMSOL
Max RMSE	8.9×10^5 Pa	5.3×10^5 Pa	Ground Truth
Min RMSE	8.4×10^4 Pa	1.9×10^5 Pa	
Mean RMSE	2.8×10^5 Pa	3.6×10^5 Pa	
Mean Error Rate	0.28%	0.36%	
Training Speed	<1min	2hr	-
Inference Speed	0.8ms	0.27ms	30min

4 to 57 seconds, mainly determined by the number of internal junctions in the interconnect tree. Although the training of the lower level seems quite time-consuming, it only has to be trained *ONCE* and provides a universal predictor that can be repeatedly used in the upper level with no further tuning effort required.

Once the *HierPINN-EM* is trained, both levels will be set to inference mode. All sampling points in the interconnects will be passed simultaneously into the model to leverage the parallel computation advantage of the GPU. We tested the training and inference speeds of both *HierPINN-EM* and *EMGraph* on the interconnect tree test set, and the results are summarized in Table 3.1.

Both learning-based methods can achieve significant speedups compared to COMSOL. Specifically, the mean inference speed of *HierPINN-EM* for each interconnect tree is 0.8ms, which is 3× slower than the 0.27ms inference speed of *EMGraph*. However, the difference in inference speed is mainly caused by the difference in sampling density, as shown in Section 3.3.3. During the inference test, the number of sampling points in each segment varies according to the wire length in *HierPINN-EM*, but is fixed to only 5 in *EMGraph* due to its fixed input layer structure. This results in approximately 30× more sampling points in *HierPINN-EM*, leading to higher computational cost. However, with such adaptive sampling ability, *HierPINN-EM* can predict more accurate stress map with much better granularity. Therefore, the loss in inference speed is actually an acceptable tradeoff.

Another major advantage of *HierPINN-EM* over *EMGraph* lies in its better flexibility in the inference phase. Limited by the message-passing structure of GNN, to predict EM stress in any single segment, the whole interconnect graph requires to be fed into *EMGraph* so that the target segment can receive useful information from its multi-hop neighbors which can be leveraged to predict stress. This means that *EMGraph* can only predict stress for interconnects as a whole, but is not able to make predictions for small local regions. In contrast, *HierPINN-EM* takes position x and time t as input parameters, which enables it to make stress predictions with much better flexibility. It can predict stress for any segment or even a single point in the interconnects at any aging time by simply passing the interested location and time into the model. This enables *HierPINN-EM* to achieve more significant speedups in local stress analysis. The better accuracy, inference flexibility, and granularity of results make *HierPINN-EM* a better learning-based approach for transient EM stress analysis.

3.4 Summary

The work presented in this chapter introduces a hierarchical PINN-based method, called *HierPINN-EM*, to solve the Korhonen equations for multi-segment interconnects, which enables fast EM failure analysis. *HierPINN-EM* divides the physics laws into two levels and solves the PDE equations step by step. The lower level uses supervised learning to train a DNN model that takes parameterized neurons as inputs, serving as a universal parameterized EM stress solver for single-segment wires. The upper level employs physics-informed loss function to train a separate DNN model at the boundaries of all wire segments, enforcing stress and atom flux continuities at internal junctions in interconnects. Numerical results on a number of synthetic interconnect trees show that *HierPINN-EM* can lead to orders of magnitude speedup in training and over 79× better accuracy compared to the plain PINN method. Furthermore, *HierPINN-EM* yields 19% better accuracy with a 99% reduction in training cost over the previous graph neural network-based EM solver, *EM-Graph*. Overall, the proposed HierPINN-EM method offers a powerful and efficient solution for fast EM failure analysis in multi-segment interconnects.

Chapter 4

Full-Chip Thermal Map Estimation With Generative Adversarial Learning

4.1 Related Work and Motivation

To estimate on-chip temperature maps, there are two general strategies. The first strategy involves estimating full-chip heat maps from physics-based thermal models and power-related information [96, 88]. Such *bottom-up* numerical methods include HotSpot [36], which is based on simplified finite difference methods, finite element methods [31], equivalent thermal RC networks [27], and the recently proposed top-down behavioral thermal models based on the matrix pencil method [51] and subspace identification method [26, 56]. In general, full-chip thermal analysis from given power information requires expensive numerical analysis such as finite difference or finite element-based approaches, which are very expensive for online applications [17]. The second strategy involves using an interpolation-based approach to estimate full-chip heat maps from embedded sensor readings [20, 9]. Since the number of sensors and their placement significantly impact the accuracy of the aforementioned interpolation, smart sensor placement algorithms have been proposed that can be used during design time to find the optimal placement for the given budget of embedded temperature sensors [20, 63, 71, 70, 105, 52]. Work in [20] exploits Fourier analysis techniques to fully recover the thermal map. However, the accuracy is limited by the non-band-limited nature of the temperature signals and approximations required for non-uniform placement of the thermal sensors, which is common in heterogeneous multi-core processors. Nowroz *et al.* [63, 71] tried to minimize the number of thermal sensors in the sensor placement to recover thermal maps (or some key locations) based on interpolation of hard sensor information in frequency domain and DC domain, respectively. Such a strategy was further improved by using Eigen decomposing of the interpolation matrix, which leads to near-optimal sensor number and placement [70]. Zhang *et al.* [101, 105] proposed a statistical method for both power and thermal maps estimation, in which the correlations of power dissipation of different modules of a chip were exploited to recover the power map from sensor readings first, and temperature was estimated once the power map is obtained. However, the estimation is based on power correlation information. Recently, Ziabari *et al.* [108] introduced the power blurring method for fast 2-D temperature map computation, which is essentially the Green's function-based method in which the temperature response to unit power impulses must be computed first from FEM thermal analysis. However, this method is difficult to apply practically as accurate thermal models are not always available first.

However, the aforementioned methods either require design-time hardware changes (such as inserting or relocating sensors) or at the very least require detailed knowledge of the chip's floorplan, correlations among functional unit power sources, and constants specific to the technology-node, which are not disclosed by the original chip manufacturer. An exclusively *post-silicon* approach to real-time transient estimation of the spatial temperature distribution across the entire chip area, i.e., at time t, estimate the full-chip spatial heatmap $T(x, y)_t$, remains a challenge for existing commercial microprocessors.

On the other hand, machine learning, especially deep learning, is gaining much attention due to the breakthrough performance in various cognitive applications such as visual object recognition, object detection, speech recognition, natural language processing, etc. due to dramatic accuracy improvements in their time-series or sequential modeling capabilities [28]. Machine learning for electronic design automation (EDA) is also gaining significant traction as it provides new computing and optimization paradigms for many of the challenging design automation problems that are complex in nature. For instance, machine learning methods have been applied to power modeling [30] and design space exploration [43]. Additionally, machine-learning-based schemes have recently been explored to build a workload-dependent thermal prediction model [103], where the future steady-state temperature of the chip can be predicted by application characteristics and physical features.

Recently, long-short-term memory (LSTM) based machine learning approach based on Intel Performance Counter Monitor (PCM) metrics has been proposed for hot spot detection [73, 75] and for full-chip thermal map estimation [74] of commercial off-the-shelf multi-core processors. To improve efficiency, 2D discrete cosine transformation (DCT) is used to compress the thermal images for the learning process [74]. However, this method needs to know the historical data of both PCM and temperatures for the training, which can be expensive. Furthermore, the accuracy of this approach is still less than expected due to the data compression process.

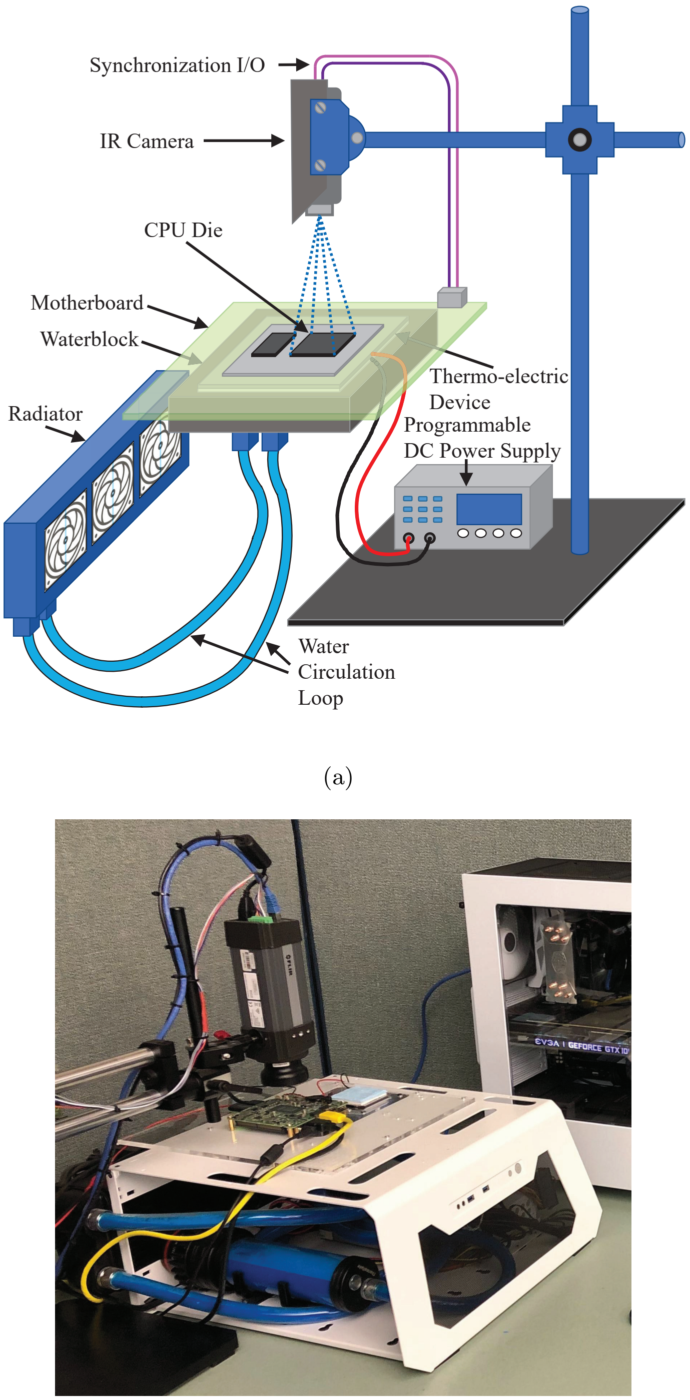
Recently, GAN-based methods have been applied for VLSI physical designs such as generation of various noise maps to facilitate the IR-drop noise sensor placement [54], for layout lithography analysis [99], and sub-resolution assist feature generation [5], for analog layout well generation [95]. However, few studies have investigated data-driven circuit-level and thermal analysis to model the dynamic systems described by the partial differential equations.

4.2 Training Data Preparation

Sufficient data is always vital for machine learning methods. To enable the proposed model to learn the distribution of PCM data and map it to the correct thermal distribution map, sufficient amount of training data is a must. In this work, a large amount of thermal distribution data of the CPU (called *thermal map* in this work) and corresponding real-time PCM data are required from which the model can learn the transformation scheme in between. In what follows, we will present the setup used to acquire the training data, as well as the necessary pre-processing methods performed on the training set prior to feeding them to the model.

To externally acquire accurate thermal maps of a working CPU, we propose to use a measurement system based on an infrared (IR) camera. Fig. 4.1 illustrates the overall setup of our thermography system. The IR camera over the chip is a FLIR A325sc (16-bit 320×240 pixels, 60Hz). The camera is rated for the temperature range of $0°C$ to $328°C$, and spectral range of $7.5\mu m$ to $13\mu m$. A microscope lens is used to provide a finer spatial resolution of $50\mu m$/px. The CPU used in our test is an Intel i7-8650U working on an Intel ® NUC7i7DNHE motherboard with the stock CPU cooler removed. The distance between the camera and the chip is approximately $70mm$. When the CPU is running, the thermo-electric device mounted at the back of the chip transfers heat from its upper side to the other. The water block and circulation loop attached below further dissipate the heat into the radiator, where the heat finally radiates to the air. With this setup, we are able to maintain the temperature of the CPU within its specified range as the stock cooler between the IR camera and the chip is removed. To synchronize the captured thermal map with its corresponding PCM data, we connect the IR camera and the CPU through a synchronization I/O. Each thermal map and PCM data that were collected at the same time instant are paired and saved together as one sample.

PCM is a tool from Intel that monitors the performance and energy metrics of all series of Intel processors. The monitored metrics range widely from basic processor monitoring utilities, such as instructions per cycle (IPC) and core frequency, to sleep and energy states of the processor, and to peripheral memory bandwidth and cache miss. A



(a)

(b)

Figure 4.1: IR thermography setup used to collect training data in this work

Table 4.1: Performance metrics (Intel PCM)

Pkg.	Socket	Socket	Core 1 to 8
INST	EXEC	C6res%	EXEC
ACYC	IPC	C7res%	IPC
TIME	FREQ	C2res%	FREQ
PhysIPC	AFREQ	C3res	AFREQ
PhysIPC%	L3MISS	C6res	L3MISS
INSTnom	L2MISS	C7res	L2MISS
INSTnom%	L3HIT	C8res%	L3HIT
C0res%	L2HIT	C9res%	L2HIT
C2res%	L3MPI	C10res%	L3MPI
C3res%	L2MPI	SKT0	L2MPI
C6res%	READ		C0res%
C7res%	WRITE		C1res%
C8res%	TEMP		C3res%
C9res%	C0res%		C6res%
C10res%	C1res%		C7res%
Energy	C3res%		TEMP

number of APIs are provided for real-time monitoring which is highly suitable for our real-time full-chip thermal modeling application. The complete list of all 170 PCM metrics that we collect and employ for the thermal modeling of Intel i7-8650U is shown in Table 4.1.

The temperatures in each thermal map vary widely from $25°C$ to $100°C$, while the values of the metrics in PCM data have all kinds of scales. Some metrics only change in a small range around zero, while others range widely with several orders of magnitude. Such inconsistencies in data scales may cause severe instability and accuracy degeneration in neural networks. Before feeding them to the machine learning model, all data must be rescaled to comparable ranges. In this work, to accommodate the *tanh* activation function employed in our model, as detailed in Section 4.3, we rescale all thermal maps to the range of [-1,1] using the min-max normalization scheme as given in (4.1). For PCM data, we rescale all metrics to a mean of 0 and a standard deviation of 1 using the data standardization method.

$$Data'_{ij} = (\frac{Data_{ij} - min(Data)}{max(Data) - min(Data)} \times 2) - 1 \tag{4.1}$$

Fig. 4.2 illustrates the flow of conventional thermal modeling for full-chip estimation and our proposed *ThermGAN* method. There are multiple stages in the conventional flow. First, only thermal-related metrics are extracted from the PCM data while the exact locations of the thermal sensors are unknown. The thermal model should predict the sensor locations prior to performing the actual thermal estimation. As the final estimation is based only on the sensor data, the accuracy of full-chip thermal modeling is inherently limited. As shown in the lower flow in Fig. 4.2, our proposed GAN-based method takes all PCM data as input and is trained on measured thermal maps. The unknown physics law governing the transmission between them is automatically learned by the model, making it possible for high-accuracy full-chip thermal modeling.

We note that the proposed thermal modeling technique is orthogonal to specific CPU being modeled and the way thermal maps are obtained. It can be applied to any real-time monitoring metrics to full-chip thermal modeling of commercial multi-processor chips. The CPU used in this work is only for illustrative purposes. Furthermore, the thermal

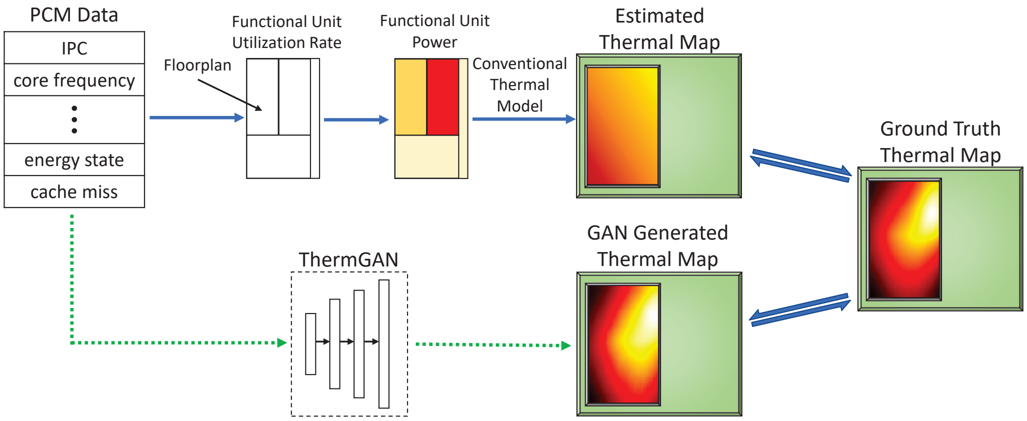


Figure 4.2: Conventional thermal modeling flow and the proposed ThermGAN flow.

maps obtained in this work were from the setup without heat sinks due to the imaging measurement requirement. However, the proposed method can be applied to any obtained or computed thermal maps. Research is underway to obtain accurate transient thermal maps from CPUs running in a practical setup with heat sinks.

4.3 CGAN-Based PCM to Temperature Transformation

4.3.1 From PCM to thermal image transformation

We first demonstrate that the process of estimating a full-chip thermal map for a multi-core processor can be viewed as an image synthesis process. In this process, the DNN can convert features (PCMs) and continuous time variables into an image.

4.3.2 Review of GANs

Generative Adversarial Nets (GANs) were first introduced by Ian Goodfellow in 2014 [29] and have drawn tremendous attention in recent years. A typical GAN consists of two networks known as the discriminator **D** and generator **G**. The generator takes a random vector **z**, usually normally distributed, as its input and maps it to an output image that is as close to those in the training dataset as possible. Images in the training set are labeled as 'real' images, and the ones produced by the generator are noted as 'fake'. The discriminator takes either a real or fake image as its input and discriminates between them. Both **D** and **G** are trained simultaneously, and this process is a contest between these two networks. The generator keeps optimizing itself to fool the discriminator with fake images while the discriminator also strives to increase its classification accuracy. Once the GAN is trained, the generator should be able to generate real-like images by mapping its random input to the learned distribution of real images. The discriminator, on the other hand, will classify all its input images as "real" or "fake" with the same possibility of 50%, indicating that fake and real images look pretty much alike and are no longer distinguishable by the discriminator.

The training of GANs is usually a tricky process and may never converge due to the gradient vanishing problem. Wasserstein GAN (WGAN) was introduced by Martin Arjovsky in [8] to mitigate this issue. Wasserstein Distance, rather than the conventional JS-Divergence, was proposed to serve as the measurement of the difference between real and fake image distributions. With such a small change in the loss function, WGAN promises a more stable training process and less likelihood of mode collapse. The results have shown significant advantages of GANs over conventional methods in terms of both performance and accuracy.

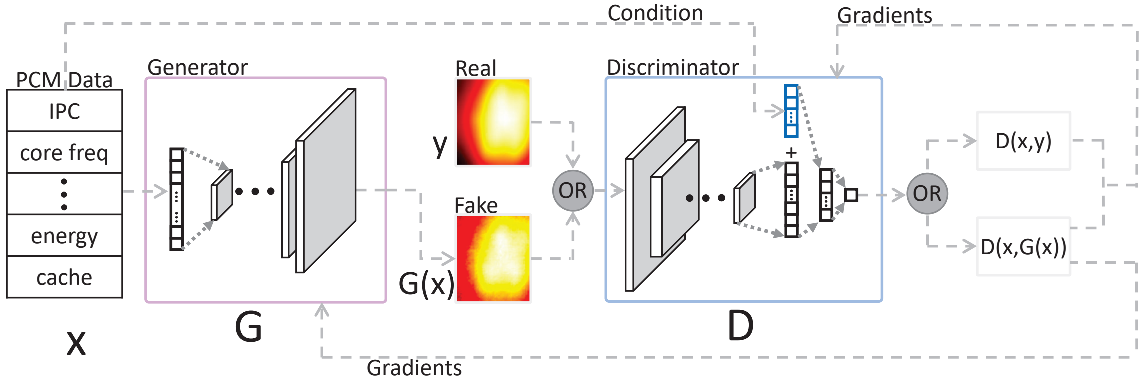


Figure 4.3: The proposed ThermGAN framework.

Fig. 4.3 illustrates our proposed structure of converting PCM data into thermal map using WGAN. The raw PCM data $\mathbf{z}$ is given to the generator $\mathbf{G}$ as a 1×170 vector with all entries standardized around zero, as described in Section 4.2. Both PCM data and thermal maps follow a unique probability distribution separately. The generator learns the mapping method between these two distributions and transforms the input PCM data $\mathbf{z}$ to its corresponding thermal map, denoted as $\mathbf{G}(\mathbf{z})$. The fake thermal map $\mathbf{G}(\mathbf{z})$ and the real ones $\mathbf{y}$ are then fed alternatively into the discriminator $\mathbf{D}$ along with their paired PCM data, which serves as the condition input. For $\mathbf{G}(\mathbf{z})$, the PCM data concatenated to it is the input of $\mathbf{G}$ used to generate $\mathbf{G}(\mathbf{z})$. For $\mathbf{y}$, the PCM data collected at the same time instant is used as the condition input. The output of the discriminator, noted as $\mathbf{D}(\mathbf{z},\mathbf{y})$ or $\mathbf{D}(\mathbf{z},\mathbf{G}(\mathbf{z}))$ depending on whether real or fake thermal map was taken as input, is a real value indicating how confident the discriminator is toward the input being a correct thermal map conditioned on the given PCM data. The objective in training the discriminator is therefore to maximize $\mathbf{D}(\mathbf{z},\mathbf{y})$ and minimize $\mathbf{D}(\mathbf{z},\mathbf{G}(\mathbf{z}))$ in terms of expectations over the distributions of $\mathbf{y}$ and

$\mathbf{z}$. Such an objective function of the discriminator can be mathematically expressed as the following equation (4.2).

$$\begin{gathered}\max_{D}\{\mathbb{E}_{\mathbf{z},\mathbf{y}}[D(\mathbf{z},\mathbf{y})]-\mathbb{E}_{\mathbf{z}}[D(\mathbf{z},G(\mathbf{z}))]- \\ \lambda_{gp}\mathbb{E}_{\hat{\mathbf{z}}}[(\|\nabla_{\hat{\mathbf{z}}}D(\hat{\mathbf{z}},\mathbf{z}))\|_2-1)^2]\}\end{gathered} \tag{4.2}$$

$\mathbb{E}_{\mathbf{z},\mathbf{y}}$ and $\mathbb{E}_{\mathbf{z}}$ represent the expectations over the distributions of $\mathbf{z}$ and $\mathbf{y}$, respectively. To maintain the 1-Lipschitz continuity of the discriminator, we adopt the gradient penalty from WGAN-GP [8]. $\hat{\mathbf{z}}$ is the interpolation between the fake and the real thermal map and λ_{gp} controls the weight of gradient penalty. The training objective of the generator is to deceive the discriminator with generated thermal maps, so its objective is to maximize the expectation of $\mathbf{D}(\mathbf{z},\mathbf{G}(\mathbf{z}))$. The objective function of the generator is defined in (4.3). Since the generator has no influence on the real thermal maps, the $\mathbf{D}(\mathbf{z},\mathbf{y})$ term is omitted in the function.

$$\min_{G}\{\mathbb{E}_{\mathbf{z}}[-D(\mathbf{z},G(\mathbf{z}))]+\lambda_{L2}\cdot\mathbb{E}_{\mathbf{z},\mathbf{y}}[\|\mathbf{y}-G(\mathbf{z})\|_2]\} \tag{4.3}$$

In both (4.2) and (4.3), we use the Wasserstein Distance as the loss function, which has the advantage of higher training stability and convergence possibility. The detailed architecture and parameters of the *ThermGAN* are shown in Table 4.2. We discard the random noise from the original GAN, as in our work, there are abundant PCM data in the training set that follow a certain distribution. This makes the PCM data itself can be seen as random noise, just as the original $\mathbf{z}$ vector does. The PCM data given to the generator is first passed through a fully connected layer and reshaped to a square array. Then it is upsampled through six transposed convolutional layers and outputted as a 256×256 fake thermal map.

Table 4.2: ThermGAN parameters used in this work

Generator				Discriminator			
Layer	Kernel	#Output	Activation	Layer	Kernel	#Output	Activation
FC	-	8192	LReLU	Conv	5x5	128×128×64	ReLU
Reshape	-	4×4×512	-	Conv	5x5	64×64×128	ReLU
Conv_trans	5x5	8×8×512	LReLU	Conv	5x5	32×32×256	ReLU
Conv_trans	5x5	16×16×512	LReLU	Conv	5x5	16×16×512	ReLU
Conv_trans	5x5	32×32×256	LReLU	Conv	5x5	8×8×512	ReLU
Conv_trans	5x5	64×64×128	LReLU	Conv	5x5	4×4×512	ReLU
Conv_trans	5x5	128×128×64	LReLU	Conv	5x5	2×2×512	ReLU
Conv_trans	5x5	256×256×1	tanh	FC	-	512	ReLU
-	-	-	-	FC	-	1	None

All thermal maps are originally 185×154 in dimensions; however, for the convenience of being handled by the discriminator, they are expanded to 256×256 by equally padding zero values in every dimension. The discriminator is a conventional convolutional classifier with only one neuron as an output and, to utilize the Wasserstein distance, no activation function is applied to it.

4.3.3 Transient thermal map estimation

Traditionally, computing thermal information from power is a time-convolutional operation, which requires historical power data. However, our thermal image generation problem from the utilization and on-chip sensor readings can be viewed as a real-time inverse or fitting problem using on-chip real-time information. Similar problems based on limited on-chip sensor readings have been explored by many pre-silicon temperature estimation methods [63, 71, 70].

For our problem, the PCM metrics indeed consist of real-time temperature sensor information for each core and the emtire chip. Although the temperature at any time instant is determined by history thermo-information, such a dependency is already decoupled by the temperature sensors, allowing for the estimation of the thermal map. As shown in the experimental section, *ThermGAN* can produce highly accurate transient thermal map estimation and outperforms the time-dependent LSTM model from [74] in terms of both accuracy and speed.

4.4 Experimental Results and Discussion

In this section, we present the experimental results demonstrating the speed and accuracy of our proposed *ThermGAN* model for PCM data to thermal map estimation.

We implemented the entire network in Python 3.7 based on TensorFlow(1.14.0) [3], a widely used open-source machine learning library. The model was trained for 10 epochs on a Linux server with 2 Xeon E5-2698v2 2.3GHz processors and Nvidia Titan X GTX GPU. The batch size was set to 8, and each data sample consisted of a pair of synchronized

Table 4.3: Benchmarks

Processor	Memory	System
AObench	PHPbench	T-test
Compress-7zip	Cyclictest	Cachebench
Encode-flac	Git	RAMspeed
Build-gcc	Mbw	Stream
Idle	Dbench	Aio-stress
-	Tinymem	Fio
-	-	Tiobench

PCM data and thermal map. We used 18 computationally intensive benchmarks from the Phoronix benchmark suite [65] to collect the training data. As listed in Table 4.3, the benchmarks were split into three categories: processor, memory, and system. The variety of the benchmarks ensured that the CPU was subjected to different kinds of workloads, leading to the diversity of the training samples. For each workload, we kept the CPU running for 4 minutes and sampled the data at a frequency of 60Hz. At each time instant, both PCM data and the thermal map were captured simultaneously and saved in pair as one sample. Finally, we obtained 14,300 samples for each benchmark, and a total of 257,400 samples were collected in the training set.

The collected raw samples are preprocessed as described in Section 4.2. To better validate the performance of our trained model, we randomly pick 25% of the samples as the test set and only use the remaining 75% for training. The learning rate and the decay parameters in the RMSProp optimizer are set to 0.0001 and 0.9. The weight of L2-norm

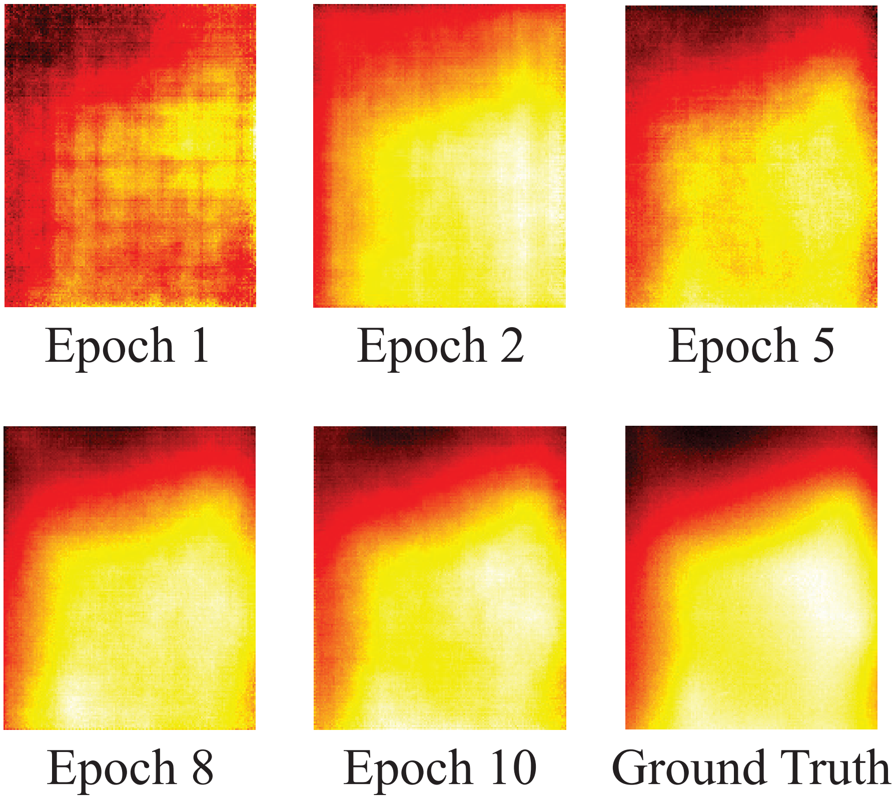


Figure 4.4: Evolution of one random sample as the training progresses.

λ_{L2} is set to 100, and λ_{gp} is set to 10. We ran the training for 10 epochs, and the results reported in this section are based on the test set, which was completely unseen by the model in the training process.

Fig. 4.4 visualizes the training process by showing the evolution of the output of the generator. We randomly picked one sample from the training set and show results in 5 epochs together with the ground truth. It can be clearly seen that the generated thermal map becomes closer to the ground truth as the training progresses.

4.4.1 Accuracy of thermal map estimation

Once the *ThermGAN* is trained, the discriminator will be discarded, and only the generator is preserved. This model can take PCM data from any time instant as input and generate a real-like thermal map indicating the full-chip thermal distribution. To verify the performance of the model, we use the root-mean-square error (RMSE) given in (4.4) as the metric to indicate the difference between the generated and real thermal map (ground truth).

$$RMSE = \sqrt{\frac{\sum_{x=1}^{W} \sum_{y=1}^{H} (T(x,y) - T'(x,y))^2}{W \times H}} \tag{4.4}$$

where T and T' are the real and generated thermal maps, respectively. Both of them are images with only one channel, which can easily suit the equation as matrices. The vertical and horizontal dimensions of the thermal maps are $H = 185$ pixels and $W = 154$ pixels, respectively. We evaluated our trained *ThermGAN* model on the test set, and the average RMSE across all 64,350 samples in the test set is 0.47°C with a standard deviation of 0.56°C. In this work, the temperature in thermal maps of our test set ranges from 25 to 100°C. Comparing the absolute values of the error with this 75°C scale, the *ThermGAN* achieves an averaged full-scale estimation error of 0.63% and a standard deviation of 0.75%. This is a quite promising result since such resolution is accurate enough for thermal estimation applications. Fig. 4.8 illustrates the comparison between generated and ground truth thermal maps, which are randomly picked from the test set. The title of each thermal map indicates the benchmark it is from and the time instant in which it was collected. We show every thermal map in both 2D-image and 3D-plot with contour lines. As shown in the figure, there are more spikes in the contour lines of the generated thermal map, indicating more noise, but the overall thermal distribution pattern is indistinguishable. The bottom row of Fig. 4.8 illustrates the error maps, which are defined as the pixel-to-pixel difference between the real and fake thermal maps. Most of the errors are within 0.5°C, except for only a few points, but still in an acceptable range, which is less than 1.5°C.

4.4.2 Real case study

The proposed *ThermGAN* is aimed at online estimation of full-chip transient thermal distribution. To evaluate the model in a real application, we run the test on another benchmark named "Gimp". It is also from the Phoronix benchmark suite and is an open-source image manipulation program that keeps the chip at an intensive workload. This benchmark was kept unseen throughout the training process and has completely no overlap with the benchmarks in the training set. We run the "Gimp" workload on the i7-8650U processor for 2 minutes while the PCM data are collected at the frequency of 60Hz and fed into the *ThermGAN* for inference. The IR camera simultaneously captures real thermal maps of the chip, which are used as ground truth to verify the *ThermGAN* inference results. A total of 7200 samples are collected, and *ThermGAN* achieves an average RMSE of 0.83°C with a standard deviation of 0.52°C. The error increases by 0.39°C compared to the result we get on the test set, which is actually a reasonable result as the distribution of data points in real cases may vary a lot from that of the training set. Despite the degradation of accuracy, the RMSE is still within 1°C, and the averaged full-scale error is only 1.1%, which is more than enough for full-chip thermal estimation in real applications. Some of the results are detailed in Fig. 4.5. We pick three time instants (883, 4260, and 6903) and

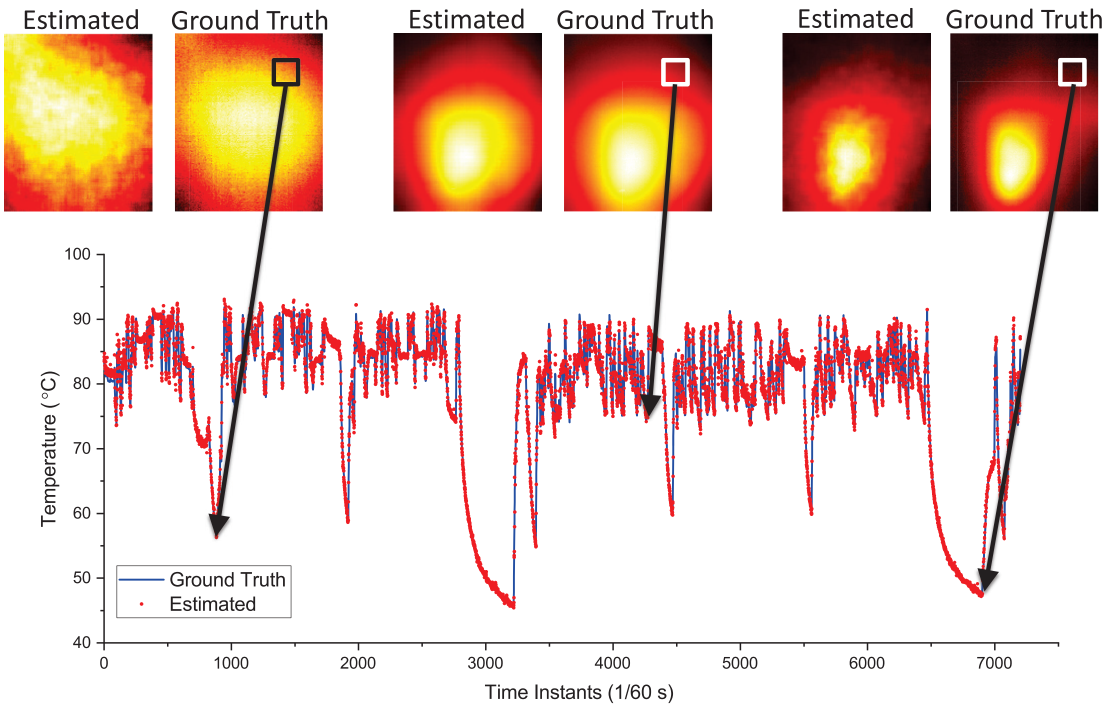
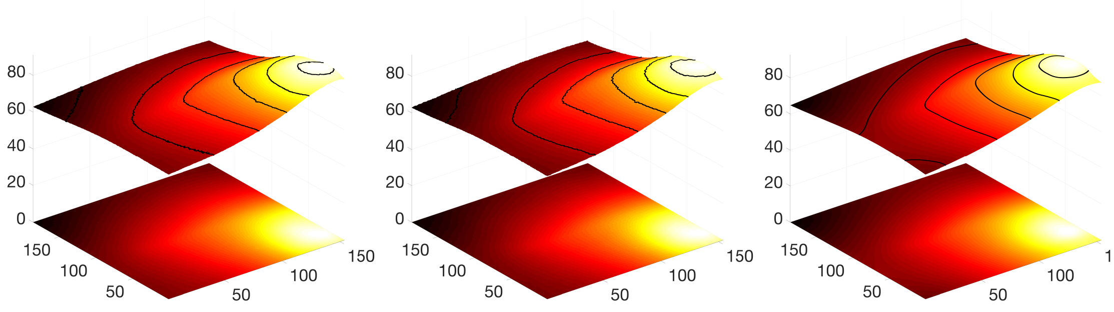


Figure 4.5: Comparison between estimated thermal distribution and ground truth on "Gimp" benchmark.

compare the estimated thermal map with its ground truth. We also fix a point on the upper right section of the chip and plot the time-series temperature prediction for this position.

4.4.3 Speed of inference

The training process of *ThermGAN* was time-consuming and cost more than 12 hours to converge. However, once the model is trained, it only reserves the generator part, which is much lighter and can be embedded into the CPU to perform real-time thermal map estimation. In our test, the time cost for each inference (one estimation of the whole chip thermal distribution based on the PCM data acquired in real-time) has a mean of *7ms* and a maximum of *7.5ms*, which translates to an inference frequency faster than 140



(a) (b) (c)

Figure 4.6: (a) Ground truth and estimated thermal map using (b) ThermGAN and (c) COMSOL FEM simulation.

thermal maps per second. Such performance further verifies that our *ThermGAN* model is capable of real-time thermal estimation. The inference may introduce some overhead into the CPU computation, but doing temperature estimation at intervals of several seconds is fast enough to meet the need for real applications such as CPU task arrangement control.

We further study the modeling efficiency by comparing it with the off-the-shelf FEM tool. Since we start with PCM parameters as inputs, we can't use traditional thermal simulator like HotSpot [36]. Instead, we first obtain the power map from the measured thermal map via 2D spatial Laplace operation and subsequent scaling based on the total power [73]. We then use COMSOL to model the setup for multi-core processors and use the power map obtained as the inputs. The FEM simulation was conducted on the same server as the GAN model, and it takes 3 seconds on average for each thermal map generation. As a result, our study shows that the proposed *ThermGAN* model can achieve a ~240X speedup over the FEM method with similar accuracy, as shown in Fig. 4.6. We remark that much faster numerical thermal analysis methods (than FEM) also exist. But the absolute speedup is less important than the millisecond performance we achieved in this work.

4.4.4 Metrics in PCM that really matters

As detailed in Table 4.1, we utilized all 170 PCM metrics as inputs for *ThermGAN*, which is actually an overkill, since not all metrics are necessarily relevant to thermal estimation. Out of the 170 metrics, only 9 of them, i.e., temperature sensor readings of 8 cores and 1 socket, are directly related to the thermal information. For the remaining 161 metrics, it is difficult to determine which ones are more correlated with CPU thermal performance and which are of lesser importance. Therefore, in this work, we leave it to the model to determine the importance of each metric as the training process will assign heavier weights to the thermo-relevant metrics automatically. During inference, the irrelevant metrics will have less influence on the accuracy of the estimated thermal map.

To verify this and identify the thermo-relevant metrics, the following PCM masking test was conducted using the trained *ThermGAN* model. For each PCM vector, we masked only one entry at a time corresponding to the metric of interest. Thus, the input dimension remained unchanged, and the trained *ThermGAN* model could still be applied to it. However, the masked metric did not participate in the feed-forward calculation. By doing so, we observed how much the output accuracy was influenced by the masked metric, while all the remaining 169 metrics remained unchanged. The RMSE of the generated thermal map against the ground truth was calculated in the same way introduced in Section 4.4.1. For each input, the mask slid through all 170 entries, resulting in 170 thermal

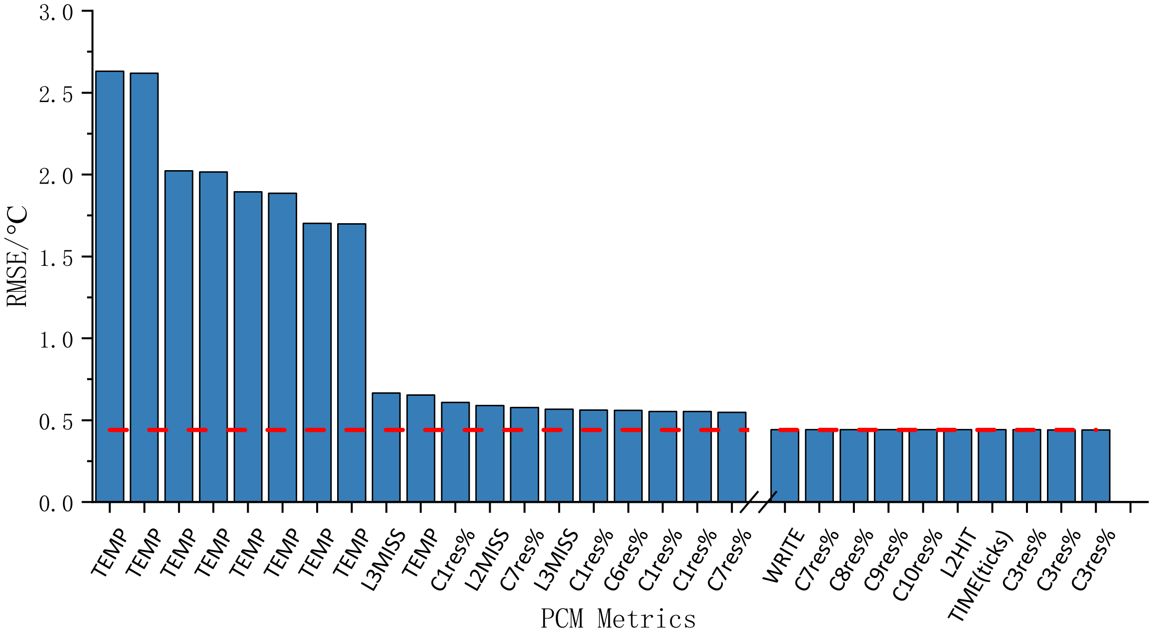


Figure 4.7: RMSE distribution across 170 masked PCM metrics.

maps, each corresponding to a masked PCM metric. We ran the masking test on the test set and plotted the average RMSEs for all masked metrics in Fig. 4.7.

The red line represents the average RMSE of thermal maps generated using unmasked PCM data. Masking different metrics results in vastly various accuracy degradation. The importance of each metric is proportional to the increment it introduces to the output RMSE. The top 8 errors are all caused by masking core temperature sensor readings. Each of them leads to an accuracy loss of more than 1.3°C. Masking the socket temperature metric caused an error of 0.65°C, which is not as much as the core temperatures but still among the top 10 metrics. Such an observation is within our expectation, but they are obviously not the only factors causing the accuracy degradation. The L3MISS is influencing the ac-

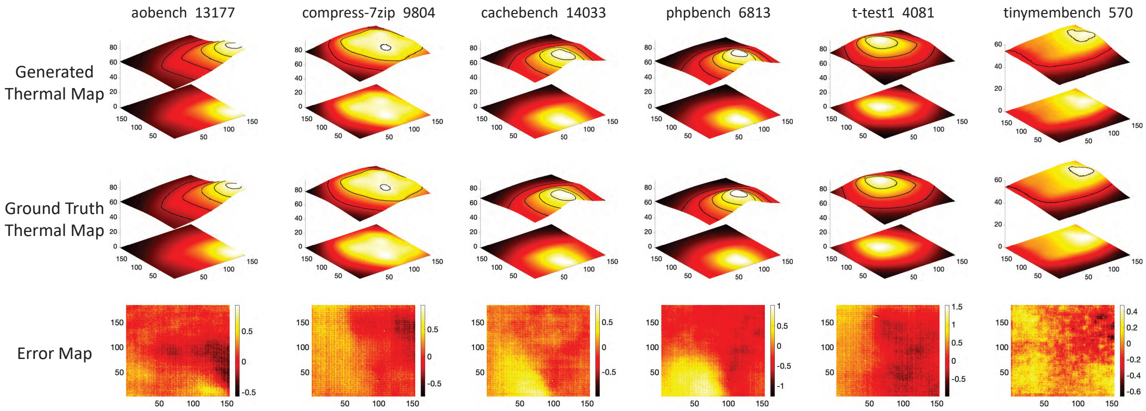


Figure 4.8: Comparison between generated and ground truth thermal maps.

curacy even more than the socket temperature. For the rest 160 metrics, 70 of them caused more than a 5% degradation in accuracy compared to the baseline 0.44°C, and among which 33 metrics led to >10% accuracy loss. We refer to these top 80 metrics as thermo-relevant metrics, and the rest 90 metrics are playing a relatively small (<5%) or even negligible role in the estimation, which implies that they are not thermo-relevant metrics. Apart from the temperatures, most of the thermo-relevant metrics are related to C-state, which reflects the idle power-saving information per core. The other thermo-relevant metrics consist of frequencies, L3 caches, instructions per cycle, and so on.

4.4.5 Comparisons with state of the arts

In this subsection, we compare *ThermGAN* with a recently proposed post-silicon full-chip thermal estimation method [74] and the pre-silicon estimation method [70].

Work in [74] is a machine-learning-based model aimed at full-chip thermal estimation using PCM data. It employed LSTM as its backbone and is implemented on the dual-core i5-3337U, which has only 80 PCM metrics as input. To conduct a fair comparison, we increased the number of units in both its input and first layers to 170 to accommodate the 170 PCM metrics of i7-8650U. The same dataset introduced in Sec 4.2 was used for both training and testing.

The average RMSE across all testing workloads is 1.84°C, and the standard deviation is 1.11°C. In contrast, the proposed *ThermGAN* yields an average RMSE of 0.47°C and a standard deviation of 0.56°C, respectively, as previously mentioned in Sec 4.4.1. Furthermore, the computational cost for each inference is ∼17ms, which is also slower than the ∼7ms inference time yields by *ThermGAN*.

Since there is no other research on post-silicon thermal estimation other than [74], we further compare our method with the state-of-the-art pre-silicon method known as "Eigenmaps" proposed in [70]. We note that this is not an apples-to-apples comparison as the "Eigenmaps" method requires optimized sensor locations in the chip design process. For commercial off-the-shelf microprocessors, both the number and locations of the temperature sensors are fixed and may not meet the requirements of the "Eigenmaps" method. However, in this comparative research, we assume such optimizations are done and allow the "Eigenmaps" method to get the temperatures from the measured thermal maps instead of the physical sensors. The locations where the temperatures are sampled can be seen as virtual sensors which are optimized according to the algorithms in [70]. To make a fair comparison, the number of virtual sensors is set to one for each of the 4 physical cores and one for the socket. We ran the "Eigenmaps" method on the test set, and the average RMSE of estimated thermal maps is 0.94°C with a standard deviation of 0.45°C. It is slightly better than [74], but still worse than the proposed *ThermGAN* method. In terms of the overhead in real-time thermal estimation, the "Eigenmaps" method requires pre-calculating and saving a dense matrix with 811680100 single-precision floating-point entries, which translates to 3.25 GB in memory. This is quite expensive and therefore not suitable for real-time applications.

4.5 Summary

The work presented in this chapter introduces a data-driven full-chip transient thermal map estimation method for commercial multi-core microprocessors based on the generative adversarial learning method. The proposed method, named *ThermGAN*, only uses the existing embedded temperature sensors and system-level utilization information, which are available in real-time. Consequently, the methods presented in this work can be implemented by either the original chip manufacturer or a third party alike. In our approach, we treat the traditional thermal modeling problem as the image generation task based on customized conditional generative adversarial networks. The resulting *ThermGAN* can provide tool-accurate full-chip transient thermal maps from the given performance monitor traces of commercial off-the-shelf multi-core processors. Experimental results show that the trained model is very accurate in thermal estimation with an average RMSE of 0.47°C, namely, 0.63% of the full-scale error. Our data further show that the speed of the model is faster than *7.5ms* per inference, which is two orders of magnitude faster than the traditional finite element-based thermal analysis. Furthermore, the new method is ∼4x more accurate than the recently proposed LSTM-based thermal map estimation method and has faster inference speed. It also achieves $\sim$2x the accuracy with much less computational cost than a state-of-the-art pre-silicon-based estimation method.

Chapter 5

Physics-Constrained Deep Learning-Based Electrostatics Analysis

5.1 Related Work and Motivation

In this section, we review some related work. The use of neural networks with physics laws to solve classic differential equations as constraints was originally proposed in the late 1990s [50, 48]. However, those works were limited by computational power at the time. Recently, this idea has received much attention due to the recent advances in deep learning for many cognitive tasks, along with ever-increasing computational resources [49]. Recently, Raissi *et al* proposed to solve 1-D PDEs [68, 67] using the so-called PINN, as shown in Fig. 5.1. In this neural network, the physics law in terms of partial differential

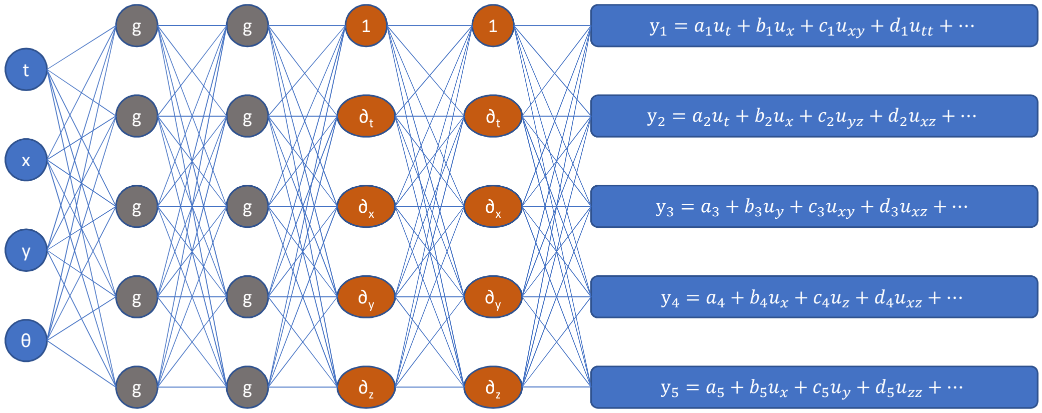

t
x
y
θ
g
1
∂_t
∂_x
∂_y
∂_z
$y_1 = a_1u_t + b_1u_x + c_1u_{xy} + d_1u_{tt} + \cdots$
$y_2 = a_2u_t + b_2u_x + c_2u_{yz} + d_2u_{xz} + \cdots$
$y_3 = a_3 + b_3u_y + c_3u_{xy} + d_3u_{xz} + \cdots$
$y_4 = a_4 + b_4u_x + c_4u_z + d_4u_{xz} + \cdots$
$y_5 = a_5 + b_5u_x + c_5u_y + d_5u_{zz} + \cdots$

Figure 5.1: Concept of physics-informed neural networks

equations, boundary conditions, and initial conditions are explicitly checked for each input (coordinates). This allows the resulting loss function and its gradient to be computed for back-propagation based training.

The PINN or PCNN approaches have recently been extended to solve high-dimensional PDEs by approximating the Galerkin method using neural networks [78], assimilate multi-fidelity training data [61], and its variational analyses have been explored based on arbitrary polynomial chaos [102] and adversarial inference [98]. However, these models can only solve small-sized PDEs with simple boundary conditions. PDE problems with complex boundaries and geometries actually still remain a challenging problem, as demonstrated by this work. Berg *et al.* proposed a new loss function so that the boundary conditions can be hard-coded to be automatically fulfilled by using two well-defined neural network-enabled auxiliary functions [10]. Recently, Sun *et al.* applied the PCNN concept to solve uncertainty quantification problems of fluid flows described by the Navier-Stokes PDEs [82]. This method introduced more parameters into the neural networks to model the parameter variations and showed that PCNN-based models can yield significant speedup over the first-principle-based numerical approaches. However, all these published works were demonstrated on many small problems with simple boundary conditions, and the claimed accuracies were not verified on large engineering problems. As we show in this paper, the PINN/PCNN-based analysis framework still remains challenging for practical large analysis problems in terms of both speedup and accuracy.

5.2 Preliminaries

5.2.1 Electrostatics problem

As mentioned above, many VLSI-related problems can be concluded to the electrostatics problem, where electric current does not exist, and there are only static electric fields due to the voltages applied or charges. They are governed by the first equation of Maxwell's equations, also known as Gauss's law:

$$\nabla^2 u(x) = \frac{-\rho}{\epsilon}, x \in \Phi \tag{5.1}$$

with following Dirichlet and Neumann boundary conditions:

$$\begin{aligned} u &= f(x),\ x \in \Gamma_D, \\ \nabla u \cdot \vec{n} &= g(x),\ x \in \Gamma_N, \end{aligned} \tag{5.2}$$

where Φ is the solution domain, Γ_D is the part of the boundary where Dirichlet (voltage) boundary conditions are given, Γ_N is the part of the boundary where Neumann boundary conditions are given, $u(x)$ is the unknown potential to be found, ρ is the charge density, ϵ is the permittivity, and $f(x)$ and $g(x)$ are the given voltage sources and current sources at the boundaries.

In cases where static charges are absent, which is the focus of this work, Equation (5.1) becomes the Laplace equation:

$$\nabla^2 u = 0 \tag{5.3}$$

After solving Equation (5.3), the distribution of the electric field is usually obtained by calculating the gradient as per its definition:

$$\vec{E} = -\nabla u \tag{5.4}$$

Solving for $\vec{E}$ under the given voltage boundary conditions $f(u)$ is often of more interest for many practical problems. For example, in capacitance extraction, the voltage is first set to $1V$ for one interconnect wire (indexed i) and $0V$ for other wires. Then, the induced static charge in any other wire j can be computed using Gauss's flux theorem.

5.2.2 Finite element method

In conventional methods, electrostatics problems are solved using discretization techniques such as FEM or FDM. Results obtained from commercial tools based on FEM, such as COMSOL, are usually considered reliable.

In conclusion, FEM first discretizes the domain to be solved by a mesh. The chosen mesh and the shape function define a function space. Elements of the function space are defined by expansion coefficients of the shape functions. A numerical solution to the original PDEs can then be found by searching in this function space for the one that best fits the original equations. This is done by setting up and solving a linear system from the original PDEs, with the expansion coefficients to be solved. Typically, the final linear system is composed of tens of thousands of unknowns, known as degrees of freedom (DOF).

Solving a problem of this scale is not very expensive, yet it is still noticeable in a longer routine.

5.3 Physics-Constrained Neural Network Solver for Electrostatics

In this section, we present the proposed physics-constrained neural network solver for the electrostatics problem. We first describe how the loss functions are built and then show how to extend the PCNN concept for a parameterized PCNN network so that the trained models can be applied to simulation works for different parameters and conditions.

5.3.1 PCNN models for electrostatics analysis

As we can see, PCNN essentially leverages the well-known capability of DNN as a universal function approximator [66, 67]. PCNN learns to model the behaviors of any dynamic time-dependent, nonlinear system, expressed by the given PDE with boundary and initial conditions. For the electrostatic problem, as we see from (5.1), we are computing the steady-state solution. As a result, we only use a multilayer perceptron (MLP) neural network architecture as we do not need to learn temporal information. Fig. 5.2 shows the proposed MLP-based PCNN electrostatic solver that takes the inputs (location x, y) and outputs an approximate solution. The specific hyperparameters, such as the number of hidden layers and the number of nodes in each layer, are determined experimentally. For each hidden layer, ReLU is used as the activation function. For the output layer, sigmoid is used as the activation function as we scale the output voltage range to $[0, 1]$.

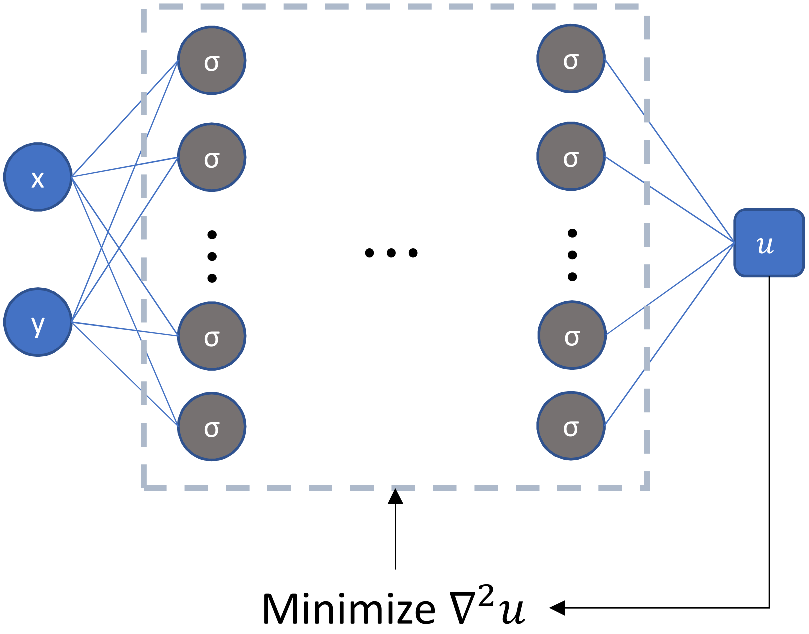


Figure 5.2: The proposed PCNN based electrostatic model. σ stands for nonlinear activations. The training of PCNN is done by minimizing the physics-based loss.

The training process of PCNN is essentially an optimization process that seeks a set of parameters $(\mathbf{W}, \mathbf{b})$ that minimizes the physics-based loss defined by the original differential equation.

Specifically, for our electrostatics problems, the physics-based loss function can be defined by the original equations (5.3) and (5.2):

$$L_{phy}(\mathbf{W}, \mathbf{b}) = \underbrace{||\nabla^2 u||_{\Phi}}_{\text{Gauss's law}} + \underbrace{||u - f(x)||_{\Gamma_D}}_{\text{Boundary condition}} \tag{5.5}$$

where $||\cdot||$ is the L2 norm over a specific domain. Then, the training of the network is defined as an optimization problem that seeks the optimal weights and biases $(\mathbf{W}^*, \mathbf{b}^*)$ that minimize the loss. The process is also shown in Fig. 5.2:

$$\mathbf{W}^*, \mathbf{b}^* = \underset{\mathbf{W},\mathbf{b}}{\operatorname{argmin}}\, L_{phy}(\mathbf{W}, \mathbf{b}) \tag{5.6}$$

In practice, the L2 norm is computed using the collocation method [48]. The domain Φ and the boundary Γ are discretized into sets of collocation points Φ_d and Γ_d, with the number of points $|\Phi_d| = N_f$ and $|\Gamma_d| = N_b$, respectively. Then, the loss is computed through the root-mean-square error on these points. For the part that corresponds to Gauss's law in the PDE form (equation residual),

$$L_{pde}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_f} \sum_{i=0}^{N_f} |\nabla^2 U(x^i, y^i)|^2, (x^i, y^i) \in \Phi_d \tag{5.7}$$

where $U(x, y)$ stands for the PCNN solution.

For the part that covers the boundary condition,

$$L_{bou}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_b} \sum_{i=0}^{N_b} |U(x^i, y^i) - u_b^i|^2, (x^i, y^i) \in \Gamma_d \tag{5.8}$$

where u_b^i is the value of the voltage boundary condition at collocation point (x^i, y^i).

Combining the two parts of the loss function, the complete loss function used in practice is defined as

$$L_{pcnn}(\mathbf{W}, \mathbf{b}) = L_{pde}(\mathbf{W}, \mathbf{b}) + L_{bou}(\mathbf{W}, \mathbf{b}) \tag{5.9}$$

By minimizing the loss function, the network $U(x, y)$ will converge to an accurate solution to the original problem.

5.3.2 Improved loss function with labels

Our study shows that for many practical problems with complicated boundary conditions, the PCNN loss function defined in (5.9) may still lead to large errors, especially for the region far away from the boundary. In this case, introducing some data from numerical solutions or measurements as labels can be instrumental for the training of PCNN networks. For example, one can obtain the solution with FEM on a coarse mesh and use it to aid in training the PCNN. Compared with the much longer runtime required to solve the PDE on a finer mesh or train the label-free PCNN, the cost of getting such assistant data is negligible.

Denote this set of data $((x, y), u)$ as Ψ_{label}, a new part of data-defined loss is then

$$L_{label}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_l} \sum_{\Psi_{label}} |U(x, y) - u|^2 \tag{5.10}$$

where Ψ_{label} is the data points and $|\Psi_{label}| = N_l$.

Furthermore, we can add first derivatives into the label data. In this case the data set is in the form of $((x, y), u, \nabla u)$. The loss function becomes

$$L_{label}(\mathbf{W}, \mathbf{b}) = \frac{1}{N_l} \sum_{\Psi_{label}} |U(x, y) - u|^2 + \lambda |\nabla U(x, y) - \nabla u|^2 \tag{5.11}$$

where λ is a coefficient, which can be determined experimentally. The final loss function with label data $L_{pcnn,L}(\mathbf{W}, \mathbf{b})$ becomes

$$\begin{aligned} L_{pcnn,L}(\mathbf{W}, \mathbf{b}) = & L_{pde}(\mathbf{W}, \mathbf{b}) + L_{bou}(\mathbf{W}, \mathbf{b}) + \\ & L_{label}(\mathbf{W}, \mathbf{b}) \end{aligned} \tag{5.12}$$

5.3.3 Parameterized PCNN surrogate models

The trained PCNN surrogate model discussed in the previous section can only solve a particular problem described by given PDEs and boundary and initial conditions. To make the model adaptive to various scenarios, PCNN models need to be parameterized [82]. For our problems, the supply voltages are parameterized as a demonstration of this idea and denoted as θ. Then the final loss function with labels and parameters becomes:

$$\begin{aligned} L_{pcnn,L,P}(\mathbf{W}, \mathbf{b}, \theta) =& L_{pde}(\mathbf{W}, \mathbf{b}, \theta) + L_{bou}(\mathbf{W}, \mathbf{b}, \theta) + \\ & L_{label}(\mathbf{W}, \mathbf{b}, \theta) \end{aligned} \tag{5.13}$$

We note that once the parameterized PCNN model is trained, the inference time of the network is much faster than running a conventional FEM-based solver for the differential equations for different parameters, which can be very useful for process variation and uncertainty quantification analysis.

5.4 Numerical Results and Discussion

In this section, we present the experimental results of our proposed PINN solver for the electric potential and electric fields of VLSI interconnects. All the models are implemented in Python based on the TensorFlow (1.14.0) library [3], which is an open-source machine learning platform.

5.4.1 Label-free PCNN network

We first implement the original PCNN, which is based on [67]. It is called the label-free PCNN solver as the model is solely trained by loss functions defined by the physics constraint of PDE in (5.6) with no simulation data (labels).

We test this solver on a VLSI design with a complicated contour of interconnects. The resulting electric field is shown in Fig. 5.3(a), and the result derived from COMSOL is shown as the ground truth [41] in Fig. 5.3(b).

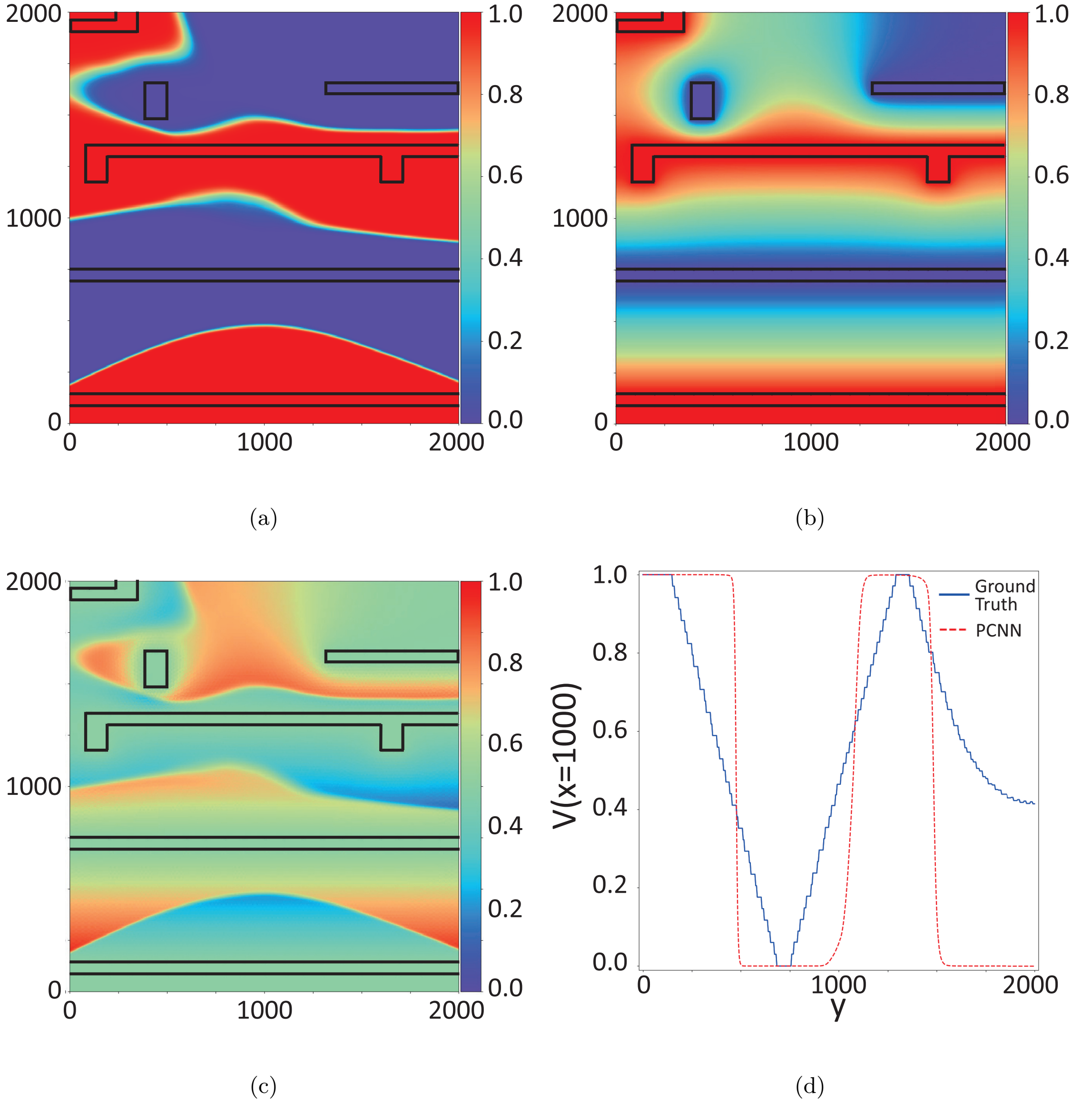


Figure 5.3: (a) Label-free PCNN solver result (b) Ground truth (c) Error map (d) Centerline voltage profile

As shown in Fig. 5.3(c), the label-free PCNN solver yields accurate results near the boundaries while errors increase in other parts, especially at points that are far away from any adjacent boundary. The overall root-mean-square-error (RMSE) of the estimated electric potential result is 0.29V, which can also be translated to 29% normalized-RMSE (NRMSE) considering the full 1V voltage range. The reason for this low accuracy is that the solver got stuck at a local minimum where it fails to generate smooth transitions from high to low electric potentials. As mentioned at the end of Section 5.1, our result here reveals the low capability of existing PCNNs in modeling practical engineering problems with complicated boundaries.

The loss function (5.9) of the label-free PCNN solver consists of two parts: boundary loss ($\boldsymbol{L_{bou}}$), which penalizes errors on the boundaries, and equation residuals ($\boldsymbol{L_{pde}}$), which enforce the Laplace equation. Both parts contribute equally to the loss function. However, the loss of equation residuals is derived by gradient-based calculation, which inherently has a gradient vanishing problem. The solver has 8 fully-connected layers, and the equation residuals loss was observed to be 5 to 7 orders of magnitude smaller than the boundary loss. This biased loss leads to a bad learning scheme in which the model is more focused on minimizing the boundary errors and ignores the equation residuals part. To make both parts more balanced, we introduce weight parameters ($\boldsymbol{W_{bou}}$ and $\boldsymbol{W_{pde}}$) to the loss function.

$$L_{pcnn}(\mathbf{W}, \mathbf{b}) = W_{pde} L_{pde}(\mathbf{W}, \mathbf{b}) + W_{bou} L_{bou}(\mathbf{W}, \mathbf{b}) \tag{5.14}$$

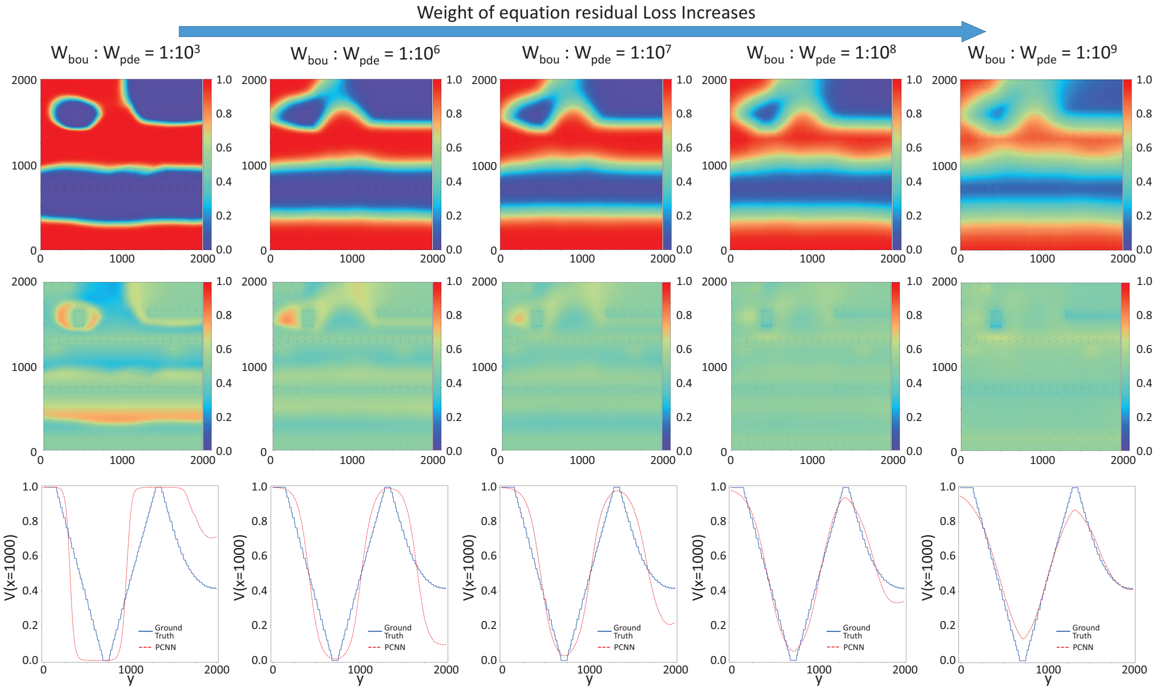


Figure 5.4: Influence of weights on PCNN results

To determine the optimal configuration of the weights, we set $\boldsymbol{W_{bou}}$ to 1 and gradually increase $\boldsymbol{W_{pde}}$ from 10^3 to 10^9. Some of the results are presented in Fig. 5.4. It is clear from the figure that increasing $\boldsymbol{W_{pde}}$ leads to improved accuracy in the transition areas. However, as $\boldsymbol{W_{pde}}$ continues to increase, the accuracy near the boundaries deteriorates. Ultimately, setting $\boldsymbol{W_{bou}}$ and $\boldsymbol{W_{pde}}$ to 1 and 10^8, respectively, yields an optimized solver with a minimized RMSE of 0.059V or NRMSE of 5.9%. The accuracy is significantly better than the original implementation, which had an RMSE of 0.29V.

The improvement in the loss function has significantly enhanced the accuracy of the existing PCNN model, but the results are still unsatisfactory due to the poor accuracy in the transition areas, which are not well-handled by PCNN. Actually, the idea of entirely excluding any simulation data from the training process in the existing PCNN is debatable. Some prior data, especially those that can be obtained at a low cost, should be incorporated into the model training. Therefore, we explore the potential of our solver further by proposing the label-assisted PCNN model.

5.4.2 Simulation-label assisted PCNN

As discussed in Section 5.3.2, we introduce some simulation data as labels to assist the training, under the assumption that these simulation data can be obtained inexpensively via existing numerical approaches, as only small or coarse meshes are needed. As shown in Fig. 5.5(a), only a limited number of simulation results are sampled (black dots) from the coarse result derived by COMSOL, and most sample points are located in the transition area. The label-assisted solver is trained in the same way as the label-free solver, except that coarse data are added to facilitate the training process, and the results are presented in Fig. 5.5. The overall accuracy is enhanced, and the RMSE is reduced to 0.049V, which is equivalent to 4.9% in terms of NRMSE. The smoothness of the transition area is also improved as the label penalizes the model for abrupt changes and forces the solver to generate a continuously fading transition from high to low electric potentials.

5.4.3 Parameterized PCNN surrogate models

The PCNN models mentioned earlier are designed for a specific case with fixed boundary conditions, meaning that the voltages on the interconnects are predetermined. Whenever a new boundary condition is given, even if the voltages only slightly differ, the

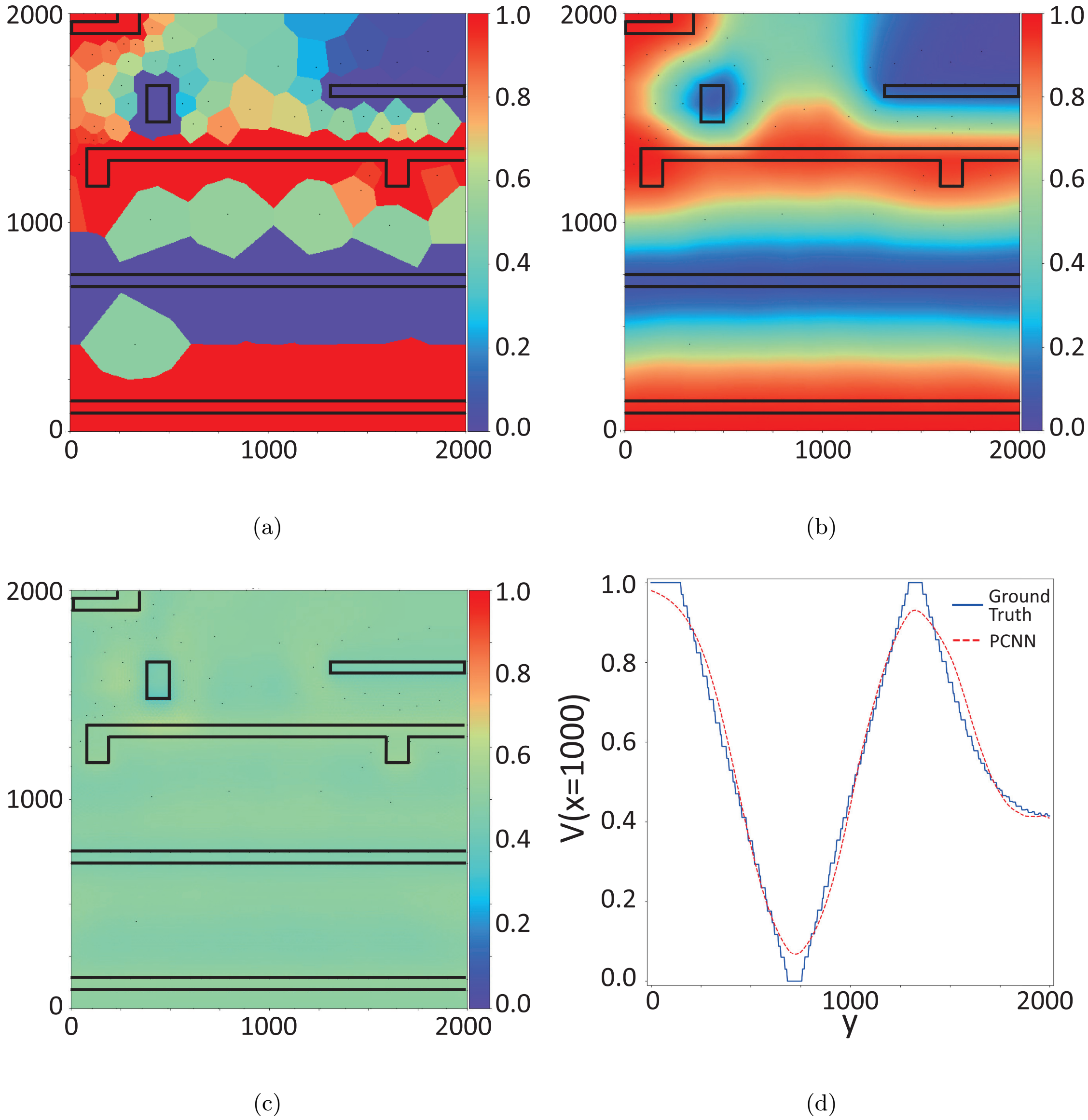
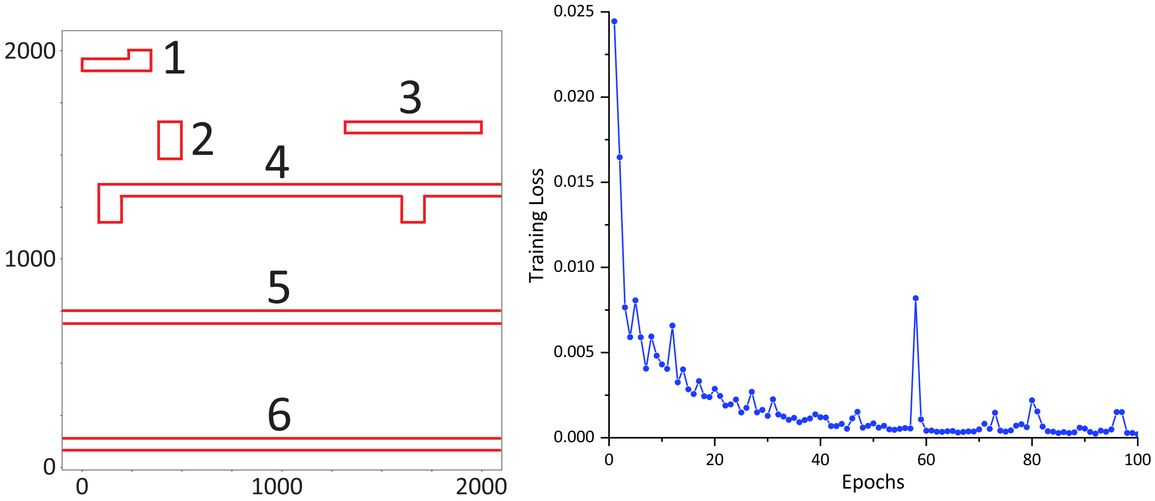


Figure 5.5: (a) Coarse label data (b) Label-assisted PCNN solver result (c) Error map (d) Centerline voltage profile



(a) (b)

Figure 5.6: (a) Boundary voltage inputs of parameterized PCNN (b) Learning curve of parameterized PCNN

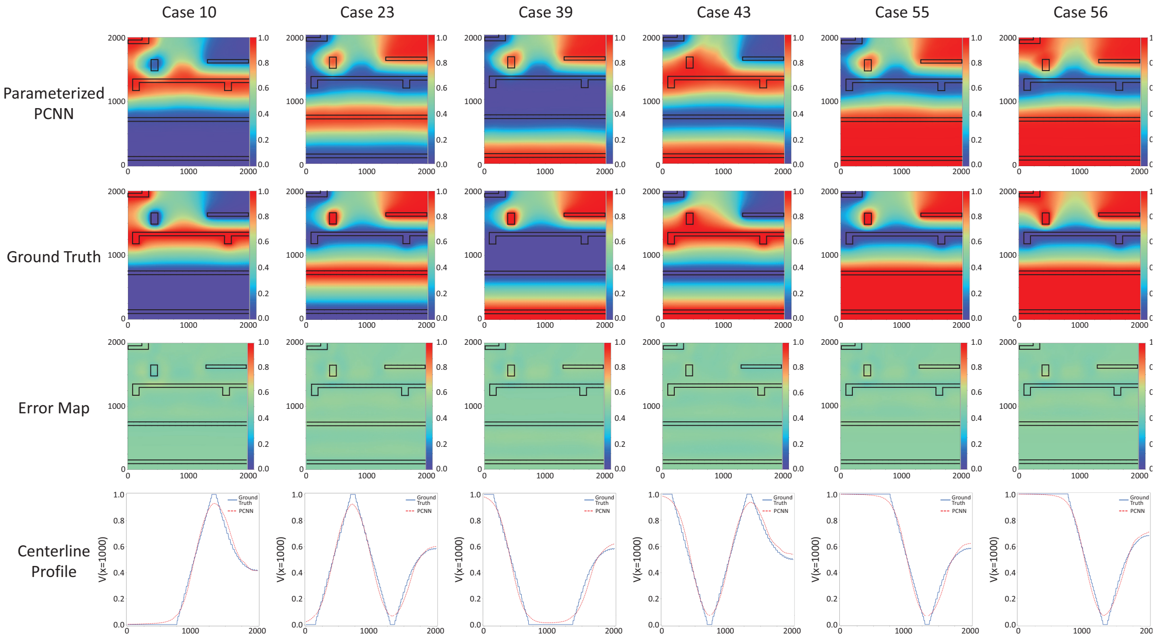


Figure 5.7: Selected training results of Parameterized PCNN

training process must start anew. This is also true for COMSOL, which requires rerunning for every new boundary condition. To address this issue, we add the voltages on boundaries as extra parameter inputs to PCNN, in addition to the original x, y coordinate inputs, and we refer to this modified model as parameterized PCNN.

The interconnects topology used to train parameterized PCNN remains the same as that in the aforementioned experiments. However, the boundary voltages can now be set to random values. We generated 64 different boundary conditions by randomly assigning either 0V or 1V voltage to each of the six wires, as shown in Fig. 5.6(a). In this work, the parameterized PCNN has eight inputs, with two for location and six for voltages. It is trained using all 64 cases simultaneously, and the learning curve is shown in Fig. 5.6(b). The electric field estimation results for some of the training cases are shown in Fig. 5.7. The average RMSE across all training cases is 0.036V, and the maximum and minimum RMSEs are 0.052V and 0.007V, respectively. While the accuracy remains at the same level as the previous solver, the model's capacity is significantly enhanced. The parameterized PCNN is capable of modeling much more cases and achieves good accuracy for each of them.

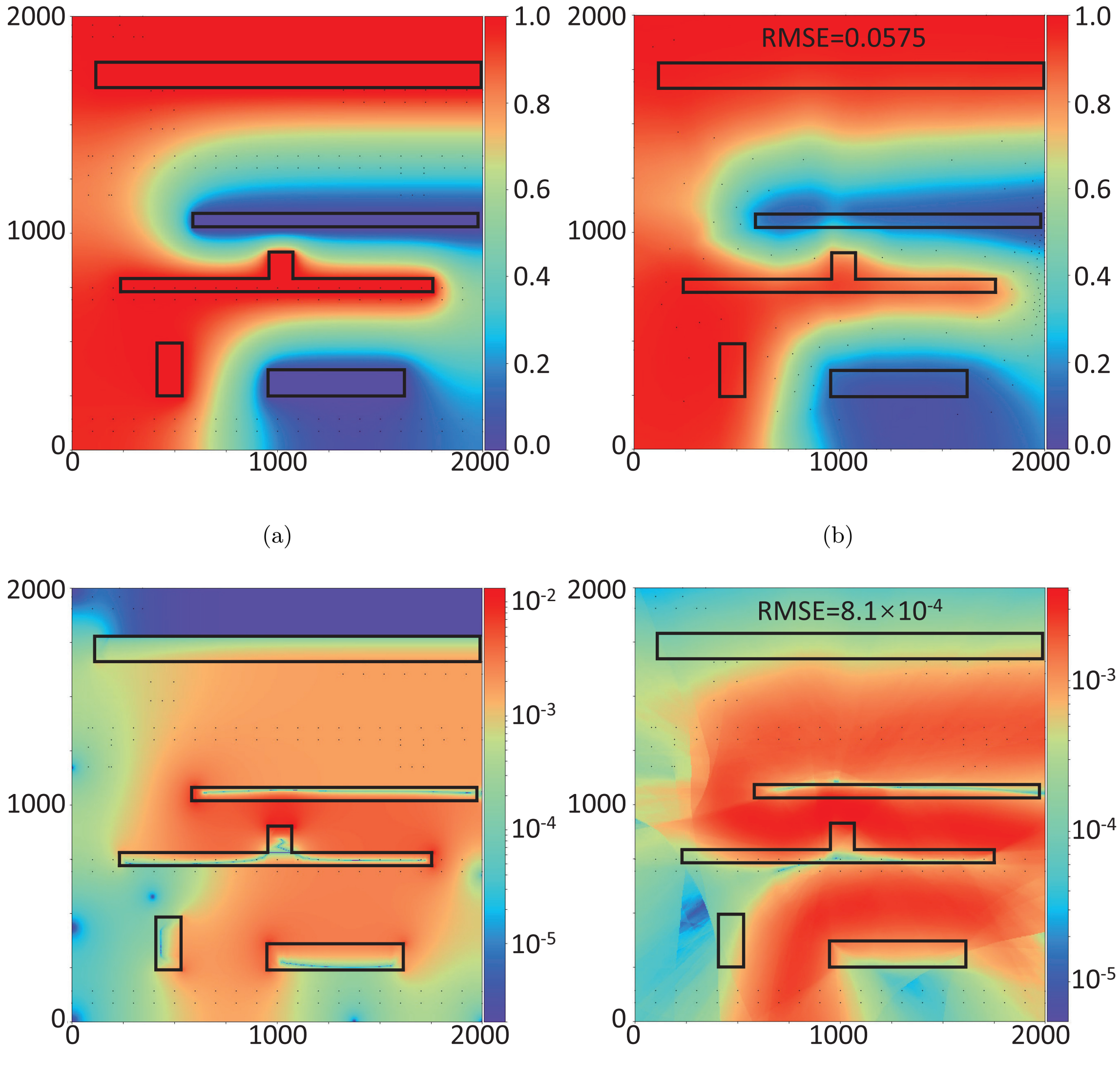
Another significant advantage of parameterized PCNN is its ability to extrapolate to brand-new cases. By altering the six voltage inputs, the trained model can perform inference for unseen test cases without the need for retraining. We tested the model on two new cases with random boundary voltages, and the RMSEs were 0.21V and 0.23V, respectively. Moreover, as no training is required for inference, the time costs for both cases were 0.84s and 0.42s, respectively, which is considerably faster than the 23.14s COMSOL required to solve each case.

The speedup can be more significant when doing inferences for a large number of unseen new cases. Additionally, when the interconnects boundary becomes more complex, it takes COMSOL more time to generate the mesh and solve for the result. However, the time cost of PCNN remains the same as the forward propagation structure is unchanged. Moreover, parameterized PCNN can solve for a small sub-area or even a single point. If the voltage potential at only one point is required, the time cost of parameterized PCNN is drastically reduced to 6×10^{-4}, which makes the speedup against COMSOL boost to $\sim 38000\times$ since COMSOL still has to solve the complete layout and then pick the required point out.

5.4.4 Electric field estimation

Once the PCNN model is trained, both the x- and y-axis components of the electric field can be derived by computing the partial derivatives of the voltage potential with respect to the x and y inputs. In this work, we leverage existing automatic differentiation, i.e., back-propagation, to compute the derivatives. We apply the label-assisted solver to a different layout with new boundary conditions. The estimated voltage potential and its corresponding electric field are shown in Fig. 5.8. The RMSE is 0.058V for voltage potential and 8.1×10^{-4} V/nm for electric field. We use this result as our baseline error in this subsection.

As the accuracy of the electric field is a new consideration in this study, the model and hyperparameters optimized in the previous section may not apply to this new case. Therefore, we fine-tuned the solver further to obtain the best accuracy in electric field



(a)

(b)

(c)

(d)

Figure 5.8: (a) Ground truth of voltage potential (b) PCNN voltage potential estimation (c) Ground truth of electric field (d) PCNN electric field estimation

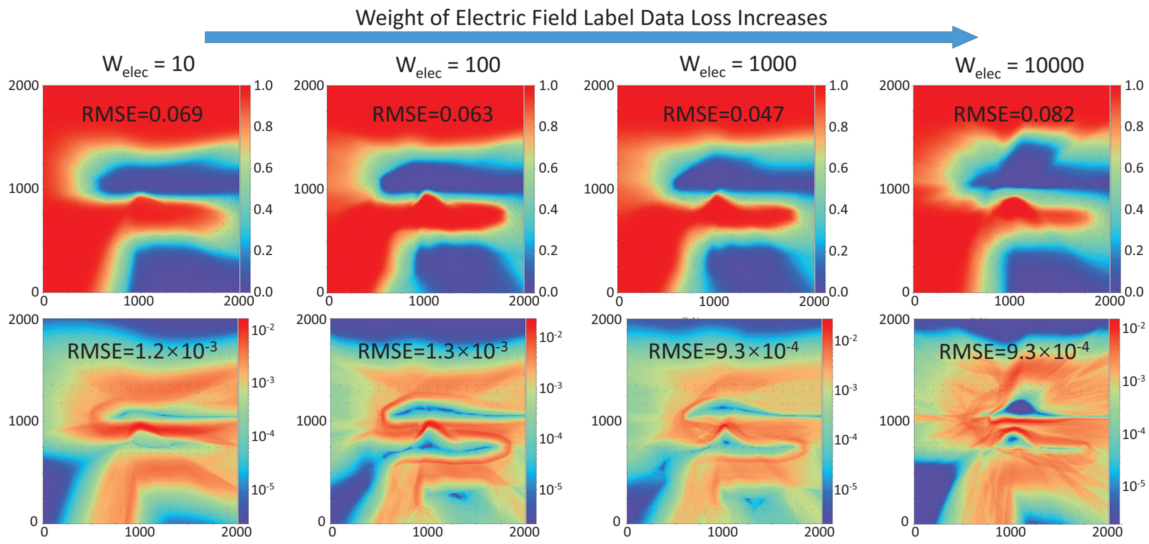


Figure 5.9: Influence of $\boldsymbol{W_{elec}}$ on voltage potential and electric field results

estimation. The coarse simulation results of electric field are used as label data in addition to existing electric potential labels. The weight for these new labels are also introduced into the loss as $\boldsymbol{W_{elec}}$, which converts the loss function into (5.15)

$$L_{pcnn,L}(\mathbf{W},\mathbf{b}) = W_{pde}L_{pde}(\mathbf{W},\mathbf{b}) + W_{bou}L_{bou}(\mathbf{W},\mathbf{b}) + \frac{1}{N_l}\sum_{\Psi_{label}} |U(x,y)-u|^2 + W_{elec}|\nabla U(x,y) - \nabla u|^2 \tag{5.15}$$

To determine the optimized weight configuration, we set $\boldsymbol{W_{pde}}$ to zero to eliminate the influence of second-order derivatives. Then, we fixed $\boldsymbol{W_{bou}}$ to 1 and gradually increased $\boldsymbol{W_{elec}}$. The evolution of the results is shown in Fig. 5.9. The lowest error in both voltage potential (0.047V) and electric field (9.3×10^{-4} V/nm) are achieved when the weights ($\boldsymbol{W_{bou}}$: $\boldsymbol{W_{elec}}$: $\boldsymbol{W_{pde}}$) are set to $1 : 10^3 : 0$. The result accuracy is worse than the baseline, which is within our expectation since the influence of $\boldsymbol{W_{pde}}$ is temporarily isolated.

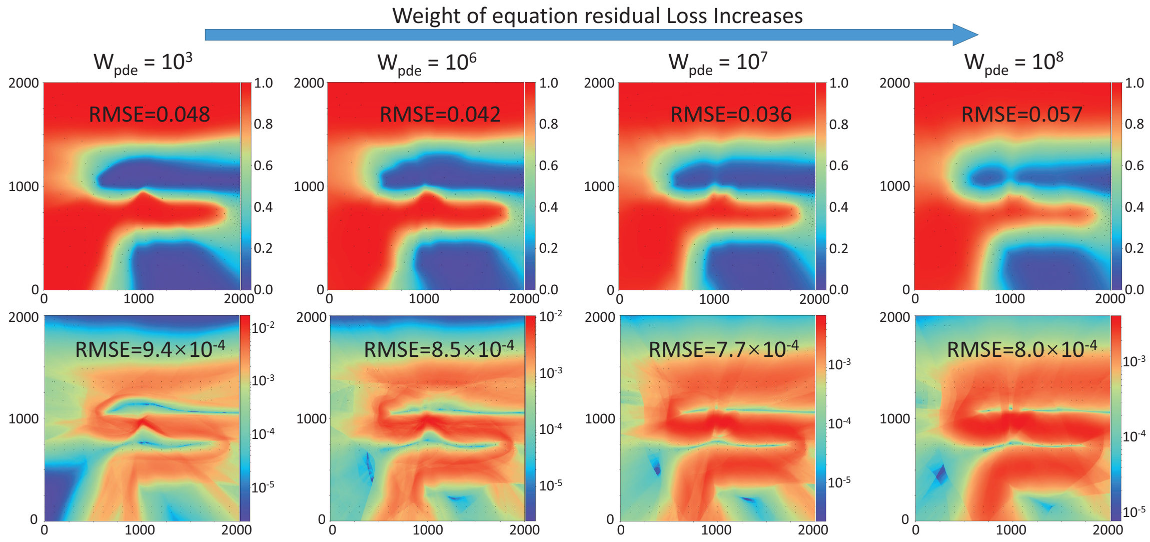


Figure 5.10: Influence of $\boldsymbol{W_{pde}}$ on voltage potential and electric field results

We then added $\boldsymbol{W_{pde}}$ back to the loss function and gradually increased it to find the final optimized combination of three weights. The evolution of the results is shown in Fig. 5.10. The best accuracy is achieved when the weights are set to $1 : 10^3 : 10^7$. Compared to the baseline, the RMSE of the voltage potential is reduced from 0.058V to 0.036V and from 8.1×10^{-4} V/nm to 7.7×10^{-4} V/nm for the electric field.

5.5 Summary

This chapter presents a 2D electric field analysis method based on the physics-constrained deep learning concept. We show how to formulate the loss functions to consider the Laplace differential equations with voltage boundary conditions for typical electrostatic analysis problems so that the supervised learning process can be carried out. We apply the resulting *PCEsolve* solver to calculate electric potential and electric field for VLSI interconnects with complicated boundaries. Our study for purely label-free training (in which no information from FEM solver is provided) shows that *PCEsolve* can give accurate results around the boundaries, but the accuracy degenerates in regions far away from the boundaries. However, with the assistance of coarse simulation data at collocation points derived from FEM analysis, *PCEsolve* can be much more accurate across all the solution domain. Numerical results demonstrate that *PCEsolve* achieves an average error rate of 3.6% on 64 cases with random boundary conditions, and it is 27.5× faster than COMSOL on test cases. The speedup can be further boosted to $\sim 38000\times$ in single-point estimations. We also studied the impacts of weights on different components of loss functions to improve the model accuracy for both voltage and electric field.

Chapter 6

Conclusions

The reliability of Very Large Scale Integration (VLSI) circuits is a crucial concern for the design and manufacture of modern electronic devices. In this article, we reviewed the culmination of our work and shared our contributions to major VLSI reliability issues, including electromigration (EM), time-dependent dielectric breakdown (TDDB), and temperature variation. Specifically, our contributions and results are summarized below.

6.1 Data-Driven Learning-Based Electromigration Analysis

Chapter 2 introduces two data-driven learning-based EM analysis methods for multi-segment interconnects. Firstly, we present an innovative model called *EM-GAN*, which is a generative adversarial network (GAN)-based transient hydrostatic stress analysis model for EM failure assessment. In this approach, the numerical partial differential equations (PDE) solving problem is considered a time-varying 2D image-to-image problem, where the input is the multi-segment interconnects topology with current densities, and the output is the EM stress distribution in those wire segments at the given aging time. The model was trained using randomly generated training sets and COMSOL simulation results. Various hyperparameters of GAN were studied and compared, and after the training process, *EM-GAN* was tested against 375 unseen multi-segment interconnect designs. The model achieved high accuracy with an average error of 6.6% and showed a significant improvement in speed with an 8.3× speedup over the recently proposed state-of-the-art analytic-based EM analysis solver.

Secondly, we introduce *EMGraph*, a graph neural network-based model for transient EM stress analysis. This model performs the node-edge regression task to predict stress at the wire segment (edge). Compared with the GAN-based image method, *EMGraph* can learn more transferable knowledge to predict stress distributions on new graphs without retraining via inductive learning. The experimental results showed that *EMGraph* has a smaller size, better accuracy, and faster speed than *EM-GAN* on several interconnect tree benchmarks. Therefore, *EMGraph* is a powerful and suitable method for transient EM stress assessment.

6.2 Physics-Informed Neural Network-Based Electromigration Analysis

Chapter 3 introduces a hierarchical Physics-informed neural network (PINN)-based method, named *HierPINN-EM*, for solving the Korhonen equations for multi-segment interconnects, aimed at fast EM failure analysis. *HierPINN-EM* splits the physics laws into two levels and solves the PDE equations step by step. The lower level uses supervised learning to train a DNN model that takes parameterized neurons as inputs and serves as a universal parameterized EM stress solver for single-segment wires. The upper level employs a physics-informed loss function to train a separate DNN model at the boundaries of all wire segments to enforce stress and atom flux continuities at internal junctions in interconnects. Numerical results on several synthetic interconnect trees demonstrate that *HierPINN-EM* can achieve orders of magnitude speedup in training and more than 79× better accuracy than the plain PINN method. Furthermore, *HierPINN-EM* yields 19% better accuracy with 99% reduction in training cost over the previous graph neural network-based EM solver, *EMGraph*.

6.3 Full-Chip Thermal Map Estimation With Generative Adversarial Learning

Chapter 4 introduces a novel data-driven approach, named *ThermGAN*, for estimating full-chip transient thermal maps of commercial multi-core microprocessors. The proposed method leverages the existing embedded temperature sensors and system level utilization information available in real-time, making it suitable for implementation by either the original chip manufacturer or a third party. In our approach, we treat the traditional thermal modeling problem as an image generation task based on the customized conditional generative adversarial networks. The resulting *ThermGAN* accurately estimates the thermal maps with an average root-mean-square-error (RMSE) of 0.47°C, which is only 0.63% of the full-scale error, and a speedy inference time of less than 7.5ms per inference. *ThermGAN* outperforms traditional finite element based thermal analysis by two orders of magnitude in inference speed and is ∼4x more accurate than the recently proposed long-short-term memory (LSTM)-based thermal map estimation method, while achieving ∼2x accuracy with much less computational cost than a state-of-the-art pre-silicon based estimation method.

6.4 Physics-Constrained Deep Learning-Based Electrostatics Analysis

Chapter 5 introduces a novel method for 2D electric field analysis using physics-constrained deep learning. This method, called *PCEsolve*, formulates loss functions to consider Laplace differential equations with voltage boundary conditions for typical electrostatic analysis problem so that the supervised learning process can be carried out. We apply *PCEsolve* to calculate the electric potential and field for VLSI interconnects with complex boundaries. Our study demonstrates that purely label-free training (in which no information from the finite element method (FEM) solver is provided) results in accurate boundary solutions, but accuracy degrades further from the boundaries. However, the incorporation of coarse simulation data at collocation points from FEM analysis greatly improves accuracy across the entire solution domain. Numerical results show that *PCEsolve* achieves an average error rate of 3.6% on 64 cases with random boundary conditions and is 27.5× faster than COMSOL on test cases. In single-point estimations, speedup can be further boosted to $\sim 38,000\times$. We also explore the impacts of weight variations on different components of loss functions to improve the model accuracy for both voltage and electric field.

Bibliography

[1] Comsol multiphysics. https://www.comsol.com/.

[2] International technology roadmap for semiconductors (ITRS), 2015. http://www.itrs2.net/itrs-reports.html.

[3] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association.

[4] Ali Abbasinasab and Malgorzata Marek-Sadowska. RAIN: A tool for reliability assessment of interconnect networks—physics to software. In *Proc. Design Automation Conf. (DAC)*, pages 133:1–133:6, New York, NY, USA, 2018. ACM.

[5] Mohamed Baker Alawieh, Yibo Lin, Zaiwei Zhang, Meng Li, Qixing Huang, and David Z. Pan. GAN-SRAF: Sub-Resolution Assist Feature Generation Using Conditional Generative Adversarial Networks. In *Proceedings of the 56th Design Automation Conference*, DAC '19, pages 1–6, New York, NY, Jun. 2019. ACM Press.

[6] AMD. AMD uProf. `https://developer.amd.com/amd-uprof/`.

[7] H. Amrouch and J. Henkel. Lucid infrared thermography of thermally-constrained processors. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 347–352, July 2015.

[8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv e-prints*, page arXiv:1701.07875, Dec. 2017.

[9] F. Beneventi, A. Bartolini, P. Vivet, and L. Benini. Thermal analysis and interpolation techniques for a logic + wideio stacked dram test chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(4):623–636, April 2016.

[10] Jens Berg and Kaj Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28 – 41, 2018.

[11] J. R. Black. Electromigration-A Brief Survey and Some Recent Results. *IEEE Trans. on Electron Devices*, 16(4):338–347, Apr. 1969.

[12] I. A. Blech. Electromigration in thin aluminum films on titanium nitride. *Journal of Applied Physics*, 47(4):1203–1208, 1976.

[13] S. Chatterjee, V. Sukharev, and F. N. Najm. Power Grid Electromigration Checking Using Physics-Based Models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(7):1317–1330, July 2018.

[14] H. Chen, S. X.-D. Tan, X. Huang, T. Kim, and V. Sukharev. Analytical modeling and characterization of electromigration effects for multibranch interconnect trees. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 35(11):1811–1824, 2016.

[15] H.-B. Chen, S. X.-D. Tan, J. Peng, T. Kim, and J. Chen. Analytical modeling of electromigration failure for vlsi interconnect tree considering temperature and segment length effects. *IEEE Transaction on Device and Materials Reliability (T-DMR)*, 17(4):653–666, 2017.

[16] L. Chen, S. X.-D. Tan, Z. Sun, S. Peng, M. Tang, and J. Mao. Fast analytic electromigration analysis for general multisegment interconnect wires. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pages 1–12, 2019.

[17] Y.-K. Cheng, C.-H. Tsai, C.-C. Teng, and S.-M. Kang. *Electrothermal Analysis of VLSI Systems.* Kluwer Academic Publishers, 2000.

[18] Tai-Yu Chou and Zoltan J Cendes. Capacitance calculation of ic packages using the finite element method and planes of symmetry. *IEEE transactions on computer-aided design of integrated circuits and systems*, 13(9):1159–1166, 1994.

[19] Sanjay Choudhary. Physics informed neural networks. Electronic Design Process Symposium Lecture, 10 2019.

[20] Ryan Cochran and Sherief Reda. Spectral techniques for high-resolution thermal characterization with limited sensor data. In *Proc. Design Automation Conf. (DAC)*, pages 478–483, 2009.

[21] C. Cook, Z. Sun, E. Demircan, M. D. Shroff, and S. X.-D. Tan. Fast electromigration stress evolution analysis for interconnect trees using krylov subspace method. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 26(5):969–980, May 2018.

[22] Chase Cook, Zeyu Sun, Ertugrul Demircan, Mehul D. Shroff, and Sheldon X.-D. Tan. Fast Electromigration Stress Evolution Analysis for Interconnect Trees Using Krylov Subspace Method. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 26(5):969–980, May 2018.

[23] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

[24] RL De Orio, Hajdin Ceric, and Siegfried Selberherr. Physically based models of electromigration: From black's equation to modern tcad models. *Microelectronics Reliability*, 50(6):775–789, 2010.

[25] K. Dev, A. N. Nowroz, and S. Reda. Power mapping and modeling of multi-core processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*, pages 39–44, Sept 2013.

[26] T. Eguia, S. X.-D. Tan, R. Shen, D. Li, E. H. Pacheco, M. Tirumala, and L. Wang. General parameterized thermal modeling for high-performance microprocessor design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 2011.

[27] Y. C. Gerstenmaier and G. Wachutka. Rigorous model and network for transient thermal problems. *Microelectronics Journal*, 33:719–725, September 2002.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. `http://www.deeplearningbook.org`.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[30] Joseph L Greathouse and Gabriel H Loh. Machine learning for performance and power modeling of heterogeneous systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2018.

[31] Siva P. Gurrum, Yogendra K. Joshi, William P. King, Koneru Ramakrishna, and Martin Gall. A compact approach to on-chip interconnect heat conduction modeling using the finite element method. *Journal of Electronic Packaging*, 130:031001.1–031001.8, September 2008.

[32] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1024–1034. Curran Associates, Inc., 2017.

[33] Vinay Hanumaiah and Sarma Vrudhula. Energy-efficient operation of multicore processors by DVFS, task migration, and active cooling. *IEEE Trans. on Computers*, 63(2):349–360, February 2014.

[34] M Hauschildt, C Hennesthal, G Talut, O Aubel, M Gall, K B Yeap, and E Zschech. Electromigration Early Failure Void Nucleation and Growth Phenomena in Cu And

Cu(Mn) Interconnects. In *IEEE Int. Reliability Physics Symposium (IRPS)*, pages 2C.1.1–2C.1.6, 2013.

[35] Tianshu Hou, Ngai Wong, Quan Chen, Zhigang Ji, and Hai-Bao Chen. A space-time neural network for analysis of stress evolution under dc current stressing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2022.

[36] Wei Huang, Shougata Ghosh, Siva Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R. Stan. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, May 2006.

[37] Xin Huang, Armen Kteyan, Sheldon X.-D. Tan, and Valeriy Sukharev. Physics-Based Electromigration Models and Full-Chip Assessment for Power Grid Networks. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 35(11):1848–1861, Nov. 2016.

[38] Intel. Intel Performance Counter Monitor (PCM). `https://software.intel.com/en-us/articles/intel-performance-counter-monitor`.

[39] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[40] Wentian Jin, Liang Chen, Sheriff Sadiqbatcha, Shaoyi Peng, and Sheldon X.-D. Tan. Emgraph: Fast learning-based electromigration analysis for multi-segment interconnect using graph convolution networks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 919–924, 2021.

[41] Wentian Jin, Shaoyi Peng, and Sheldon X.-D. Tan. Data-driven electrostatics analysis based on physics-constrained deep learning. In *Proc. Design, Automation and Test In Europe Conf. (DATE)*, pages 1–6, Feb. 2021.

[42] Wentian Jin, Sheriff Sadiqbatcha, Zeyu Sun, Han Zhou, and Sheldon X.-D. Tan. Em-gan: Data-driven fast stress analysis for multi-segment interconnects. In *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, pages 296–303, Oct. 2020.

[43] Ryan Gary Kim, Janardhan Rao Doppa, and Partha Pratim Pande. Machine learning for design space exploration and optimization of manycore systems. In *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2018.

[44] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representation*, 2017.

[45] Joonho Kong, Sung Woo Chung, and Kevin Skadron. Recent thermal management techniques for microprocessors. *ACM Comput. Surv.*, 44(3):13:1–13:42, jun 2012.

[46] M. A. Korhonen, P. Bo/rgesen, K. N. Tu, and C.-Y. Li. Stress evolution due to electromigration in confined metal lines. *Journal of Applied Physics*, 73(8):3790–3799, 1993.

[47] K.Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *International Symposium on Computer Architecture*, pages 2–13, 2003.

[48] Isaac E Lagaris, Aristidis Likas, and Dimitrios I Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[49] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, May 2015.

[50] Hyuk Lee and In Seok Kang. Neural algorithm for solving differential equations. *Journal of Computational Physics*, 91(1):110–131, 1990.

[51] Duo Li, Sheldon X.-D. Tan, Eduardo H. Pacheco, and Murli Tirumala. Parameterized architecture-level dynamic thermal models for multicore microprocessors. *ACM Trans. Des. Autom. Electron. Syst.*, 15(2):1–22, 2010.

[52] X. Li, X. Li, W. Jiang, and W. Zhou. Optimising thermal sensor placement and thermal maps reconstruction for microprocessors using simulated annealing algorithm based on pca. *IET Circuits, Devices Systems*, 10(6):463–472, 2016.

[53] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, and Wenbin Xu. A customized graph neural network model for guiding analog ic placement. In *2020 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[54] Jinglan Liu, Yukun Ding, Jianlei Yang, Ulf Schlichtmann, and Yiyu Shi. Generative Adversarial Network Based Scalable On-chip Noise Sensor Placement. In *Proceedings of the 30th International System-on-Chip Conference*, SOCC '17, pages 239–242, New York, NY, Sep. 2017. IEEE Press.

[55] Z. Liu, S. X.-D. Tan, X. Huang, and H. Wang. Task migrations for distributed thermal management considering transient effects. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 23(2):397–401, 2015.

[56] Z. Liu, S. X.-D. Tan, H. Wang, Y. Hua, and A. Gupta. Compact thermal modeling for packaged microprocessor design with practical power maps. *Integration, the VLSI Journal*, 47(1), January 2014. in press, online access: http://www.sciencedirect.com/science/article/pii/S0167926013000412.

[57] Jingwei Lu, Pengwen Chen, Chin-Chih Chang, Lu Sha, Dennis Jen-Hsin Huang, Chin-Chi Teng, and Chung-Kuan Cheng. Eplace: Electrostatics-based placement using fast fourier transform and nesterov's method. *ACM Trans. Des. Autom. Electron. Syst.*, 20(2), March 2015.

[58] Y. C. Lu, S. S. Kiran Pentapati, L. Zhu, K. Samadi, and S. K. Lim. Tp-gnn: A graph neural network framework for tier partitioning in monolithic 3d ics. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[59] Yonghong Luo, Xiangrui Cai, Ying ZHANG, Jun Xu, and Yuan xiaojie. Multivariate time series imputation with generative adversarial networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1596–1607. Curran Associates, Inc., 2018.

[60] J.W. McPherson and H.C. Mogul. Underlying Physics of the Thermochemical E Model in Describing Low-Field Time-Dependent Dielectric Breakdown in SiO_2 Thin Films. *Journal of Applied Physics*, 84(3):1513–1523, 1998.

[61] Xuhui Meng and George Em Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *Journal of Computational Physics*, 401:109020, 2020.

[62] V. Mishra and S. S. Sapatnekar. Predicting Electromigration Mortality Under Temperature and Product Lifetime Specifications. In *Proc. Design Automation Conf. (DAC)*, pages 1–6, Jun. 2016.

[63] A. Nowroz, R. Cochran, and S. Reda. Thermal monitoring of real processors: Techniques for sensor allocation and full characterization. In *Proc. Design Automation Conf. (DAC)*, 2010.

[64] M. Pedram and S. Nazarian. Thermal modeling, analysis, and management in VLSI circuits: Principles and methods. *Proc. of the IEEE*, 94(8):1487–1501, Aug. 2006.

[65] Phoronix. Open-Source, Automated Benchmarking. `https://www.phoronix-test-suite.com/`.

[66] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[67] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[68] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. *arXiv e-prints*, page arXiv:1711.10561, November 2017.

[69] Giorgia Ramponi, Pavlos Protopapas, Marco Brambilla, and Ryan Janssen. T-CGAN: Conditional Generative Adversarial Network for Data Augmentation in Noisy Time Series with Irregular Sampling. *arXiv e-prints*, page arXiv:1811.08295, November 2018.

[70] Juri Ranieri, Alessandro Vincenzi, Amina Chebira, David Atienza, and Martin Vetterli. Eigenmaps: Algorithms for optimal thermal maps extraction and sensor placement on multicore processors. In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, pages 636–641, New York, NY, USA, 2012. ACM.

[71] S. Reda, R. Cochran, and A. N. Nowroz. Improved thermal tracking for processors using hard and soft sensor allocation techniques. *IEEE Transactions on Computers*, 60(6):841–851, June 2011.

[72] H. Ren, G. F. Kokai, W. J. Turner, and T. S. Ku. Paragraph: Layout parasitics and device parameter prediction using graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[73] S. Sadiqbatcha, H. Zhao, H. Amrouch, J. Henkel, and S. X.-D. Tan. Hot spot identification and system parameterized thermal modeling for multi-core processors through infrared thermal imaging. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2019.

[74] S. Sadiqbatcha, Y. Zhao, J. Zhang, H. Amrouch, J. Henkel, and S. X. D. Tan. Machine learning based online full-chip heatmap estimation. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 229–234, 2020.

[75] Sheriff Sadiqbatcha, J. Zhang, H. Zhao, H. Amrouch, J. Hankel, and Sheldon X.-D. Tan. Post-silicon heat-source identification and machine-learning-based thermal modeling using infrared thermal imaging. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 2020.

[76] K. Settaluri and E. Fallon. Fully automated analog sub-circuit clustering with graph convolutional neural networks. In *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1714–1715, 2020.

[77] Al Shohel, Mohammad Abdullah, Vidya A. Chhabria, Nestor Evmorfopoulos, and Sachin S. Sapatnekar. Analytical modeling of transient electromigration stress based on boundary reflections. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8, 2021.

[78] Justin Sirignano and Konstantinos Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339 – 1364, 2018.

[79] V. Sukharev. Beyond Black's Equation: Full-Chip EM/SM Assessment in 3D IC Stack. *Microelectronic Engineering*, 120:99–105, 2014.

[80] V. Sukharev and F. N. Najm. Electromigration Check: Where the Design and Reliability Methodologies Meet. *IEEE Transactions on Device and Materials Reliability*, 18(4):498–507, December 2018.

[81] Valeriy Sukharev, Armen Kteyan, and Xin Huang. Postvoiding stress evolution in confined metal lines. *IEEE Transactions on Device and Materials Reliability*, 16(1):50–60, 2016.

[82] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[83] Z. Sun, S. Yu, H. Zhou, Y. Liu, and S. X.-D. Tan. EMSpice: Physics-Based Electromigration Check Using Coupled Electronic and Stress Simulation. *IEEE Transactions on Device and Materials Reliability*, 20(2):376–389, June 2020.

[84] Cher Ming Tan. *Electomigration in ULSI Interconnects*. International Series on Advances in Solid State Electronics and Technology. Word Scientific, 2010.

[85] Sheldon X.-D. Tan, Mehdi Tahoori, Taeyoung Kim, Shengcheng Wang, Zeyu Sun, and Saman Kiamehr. *VLSI Systems Long-Term Reliability – Modeling, Simulation and Optimization*. Springer Publishing, 2019.

[86] Michael Taylor. A landscape of the new dark silicon design regime. *IEEE/ACM International Symposium on Microarchitecture*, 33(5):8–19, October 2013.

[87] Jonathan Tompson, Kristofer Schlachter, Pablo Sprechmann, and Ken Perlin. Accelerating Eulerian fluid simulation with convolutional networks. volume 70 of *Proceedings of Machine Learning Research*, pages 3424–3433, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[88] H. Wang, S. X.-D. Tan, G. Liao, R. Quintanilla, and A. Gupta. Full-chip runtime error-tolerant thermal estimation and prediction for practical thermal management. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, Nov. 2011.

[89] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H. S. Lee, and S. Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[90] Hai Wang, Jian Ma, Sheldon X.-D. Tan, Chi Zhang, He Tang, Keheng Huang, and Zhenghong Zhang. Hierarchical dynamic thermal management method for high-performance many-core microprocessors. *ACM Trans. on Design Automation of Electronics Systems*, 22(1):1:1–1:21, July 2016.

[91] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[92] S. Wang, Z. Sun, Y. Cheng, S. X.-D. Tan, and M. Tahoori. Leveraging recovery effect to reduce electromigration degradation in power/ground TSV. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 811–818. IEEE, Nov. 2017.

[93] X. Wang, Y. Yan, J. He, S. X.-D. Tan, C. Cook, and S. Yang. Fast physics-based electromigration analysis for multi-branch interconnect trees. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 169–176. IEEE, Nov. 2017.

[94] Wei Wu, Lingling Jin, Jun Yang, Pu Liu, and Sheldon X.-D. Tan. Efficient power modeling and software thermal sensing for runtime temperature monitoring. *ACM Trans. on Design Automation of Electronics Systems*, 12(3):1–29, 2007.

[95] Biying Xu, Yibo Lin, Xiyuan Tang, Shaolan Li, Linxiao Shen, Nan Sun, and David Z. Pan. WellGAN: Generative-Adversarial-Network-Guided Well Generation for Analog/Mixed-Signal Circuit Layout. In *Proceedings of the 56th Design Automation Conference*, DAC '19, pages 1–6, New York, NY, Jun. 2019. ACM Press.

[96] Y. Yang, Z. P. Gu, C. Zhu, R. P. Dick, and L. Shang. ISAC: Integrated space and time adaptive chip-package thermal analysis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 16(1):86–99, 2007.

[97] Yibo Yang and Paris Perdikaris. Physics-informed deep generative models. *arXiv e-prints*, page arXiv:1812.03511, December 2018.

[98] Yibo Yang and Paris Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019.

[99] Wei Ye, Mohamed Baker Alawieh, Yibo Lin, and David Z. Pan. LithoGAN: End-to-End Lithography Modeling with Generative Adversarial Networks. In *Proceedings of the 56th Design Automation Conference*, DAC '19, pages 1–6, New York, NY, Jun. 2019. ACM Press.

[100] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series generative adversarial networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5508–5518. Curran Associates, Inc., 2019.

[101] Yufu Zhang, A. Srivastava, and M. Zahran. Chip level thermal profile estimation using on-chip temperature sensors. In *2008 IEEE International Conference on Computer Design*, pages 432–437, 2008.

[102] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[103] K. Zhang, A. Guliani, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman. Machine learning-based temperature prediction for runtime thermal management across system components. *IEEE Transactions on Parallel and Distributed Systems*, 29(2):405–419, Feb 2018.

[104] Y. Zhang, H. Ren, and B. Khailany. Grannite: Graph neural network inference for transferable power estimation. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[105] Y. Zhang, B. Shi, and A. Srivastava. Statistical framework for designing on-chip thermal sensing infrastructure in nanoscale systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(2):270–279, 2014.

[106] H. Zhao and S. X.-D. Tan. Postvoiding fem analysis for electromigration failure characterization. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 26(11):2483–2493, Nov. 2018.

[107] Quming Zhou, Kai Sun, Kartik Mohanram, and Danny C. Sorensen. Large power grid analysis using domain decomposition. In *Proc. Design, Automation and Test In Europe. (DATE)*, pages 27–32, 3001 Leuven, Belgium, Belgium, 2006. European Design and Automation Association.

[108] A. Ziabari, J. Park, E. K. Ardestani, J. Renau, S. Kang, and A. Shakouri. Power blurring: Fast static and transient thermal analysis method for packaged integrated circuits and power devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(11):2366–2379, 2014.