Methodologies for Hardware Reliability and Efficiency

CHEN, Tinghuan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
Sep. 2021

Thesis Assessment Committee

Professor LEUNG Kwong Sak (Chair)
Professor YU Bei (Thesis Supervisor)
Professor YANG Ming-Chang (Committee Member)
Professor HU Shiyan (External Examiner)

Abstract of thesis entitled:

Methodologies for Hardware Reliability and Efficiency

Submitted by CHEN, Tinghuan

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in Sep. 2021

The advanced sensor system is composed of various hardware circuits. It is principal to achieve desired hardware reliability and efficiency. With continued scaling, the sensitivity to aging-associated wear-out phenomena has significantly increased. Consequently, electrical signals (e.g., current) will shift from their nominal values, which causes a gradual circuit failure, inaccurate measurement and sensing. Besides, overheat temperatures cause the reduction of the mean time between failures for the printed circuit board (PCB) layout in the sensor system. Therefore, methodologies for hardware reliability are required. On the processing unit, with the rapid development of convolutional neural networks (CNNs), many innovative applications of artificial intelligence have been promoted. Nevertheless, the massive CNNs' parameters bring huge challenges to model storage and data transfer on the resource-limited hardware. Thus, the potence of the sensor system necessitates methodologies for hard-

ware efficiency.

This thesis endeavors to present our research about several novel methodologies for hardware reliability and efficiency. Our research includes Bayesian modeling in hardware measurement calibration optimization, graph learning in hardware aging verification optimization, deep learning in thermal-driven hardware design optimization, and Bayesian modeling in hardware-efficiency CNNs optimization.

Firstly, Bayesian modeling methodology is studied for sensor measurement calibration optimization. During the application process, due to the inevitable slow-aging effect, sensor measurement and sensing have errors. We propose a spatial correlation model to achieve a robust calibration and a better trade-off between accuracy and runtime.

Secondly, the graph learning methodology is investigated for sensor hardware aging verification in the verification stage. Due to expensive transient simulation and user-defined stress conditions, the traditional model-based methodology may bring a long design-validation cycle and inaccurate results. We present a heterogeneous graph convolutional network to achieve significant speedup while maintaining a low accuracy loss.

Thirdly, CNNs are leveraged to guide sensor hardware design for thermal optimization in the design stage. The traditional verification then fix approaches are ill-equipped when faced with the ever-growing thermal violations. We devise CNNs-based thermal-driven PCB routing methodology to achieve thermal

well PCB layout design.

For hardware-efficiency CNNs optimization, to reduce parameters in grouped convolution-based CNNs, Bayesian learning-based framework to compress grouped convolutional layers in grouped convolution-based CNNs is firstly proposed. With our proposed framework, the grouped convolutional layer is compressed by sharing parameters among different groups.

The proposed methodologies are demonstrated in extensive experiments on advanced industrial benchmarks and open source academic benchmarks. These methodologies can accelerate hardware reliability validation, achieve a better trade-off between accuracy and runtime, and enable hardware efficiency.

摘要

先進的傳感器系統由各種硬件電路組成。實現所需的硬件可靠性和效率是重要的要求。隨著特徵尺度的持續縮小，老化相關的磨損現象的敏感性顯著增加。因此電信號（例如電流）會偏離其標稱值，從而導致電路逐漸失效、測量和感知的不准確。此外，過熱溫度會導致傳感器系統中印刷電路板 (PCB) 版圖的平均故障間隔時間減少。因此在設計中需要硬件可靠性的方法。在處理單元上，卷積神經網絡（CNN）的快速發展推動了人工智能的許多創新應用。然而海量的 CNNs 參數給資源有限的硬件上的模型存儲和數據傳輸帶來了巨大的挑戰。因此，傳感器系統需要硬件有效性的方法論。

本論文力圖展示我們對硬件可靠性和有效性幾個方面的研究。我們的研究內容包括圖學習在硬件老化驗證中的優化方法、深度學習在熱驅動硬件設計中的優化策略、貝葉斯建模在硬件測量校準中的優化手段以及基於硬件有效性的 CNN 優化。

首先，我們研究了貝葉斯建模方法在傳感器測量校準中的優化。在應用過程中，由於不可避免的緩慢老化效應，傳感器測量和感知存在誤差。我們提出了一個空間相關模型以實現穩健的校準以及在準確性和運行時間之間的更好權衡。

其次，在驗證階段研究了用於傳感器硬件老化驗證的圖學習方法。由於昂貴的瞬態仿真和用戶定義的應力條件，傳統的基於模型的方法可能會帶來較長的設計驗證週期和不准確的結果。我們提出了一個異構圖卷積網絡以在保持低精度損失的同時實現顯著的加速。

第三，利用 CNN 來指導傳感器硬件設計，以便在設計階段進行熱優化。當面臨不斷增長的熱違規時，傳統的驗證然後修復方法是不合適的。我們設計了基於 CNN 的熱驅動 PCB 佈線方法來實現具有良好熱性能的 PCB 佈線設計。

對於硬件有效性的 CNN 優化，為了減少基於組卷積的 CNN 中的參數，提出了基於貝葉斯學習框架來壓縮基於組卷積的 CNN 中的組卷積層。使用我們提出的框架，組卷積層通過在不同組之間共享參數來實現參數壓縮。

所提出的方法在先進工業數據集和開源學術數據集中得到了驗證。這些方法可以在準確性和運行時間之間實現更好的權衡，加速硬件可靠性驗證，並實現硬件有效性。

Acknowledgement

I would like to express my sincere and forever appreciation to my supervisor, Professor Bei YU, for his sophisticated guidance, sustained support, sparkling enlightenment. Without his encouragement, patience and supervision, the completion of my thesis and my Ph.D. studies must be impossible.

I also would like to greatly thank my committee members, Professor Kwong Sak LEUNG, Professor Ming-Chang YANG and Professor Shiyan HU, who have offered useful feedback throughout the process.

It has been a great honor to work with many great collaborators during my Ph.D. studies. Thanks to Hao GENG, Qi SUN, Lu ZHANG, Ran CHEN, Dr. Haoyu YANG and Dr. Yuzhe MA for heaps of constructive discussions and cooperation.

I would like to extend my appreciation to the staff in the CSE department for generously helping me.

I owe the deepest gratitude to my family for their ever-encouraging and -supporting me to pursue what I am interested in.

This work is dedicated to my ever supportive and loving family. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the development of electronic and computer engineering, the advanced sensor system is widely used in many areas, such as smart building. The advanced sensor system is composed of various hardware circuits, as shown in Fig. 1.1. The typical hardware circuits contain the integrated-circuit (IC) and the printed circuit board (PCB). Large numbers of tiny MOSFETs (metal oxide semiconductor field effect transistors) integrate into a small chip, which is called IC. While PCB mechanically supports and electrically connects electronic components (e.g., ICs) using conductive tracks, pads and other features etched from one or more sheet layers of copper laminated onto and/or between sheet layers of a non-conductive substrate. In the advanced sensor system, typical ICs are designed for various sensors, radio-frequency modules and processing units. Typical sensors and radio-frequency modules are implemented by analog circuits while processing units are implemented by digital

circuits. ICs and PCBs are the hardware fundamental of the sensor system. Some advanced applications are implemented in processing units to control the whole system.

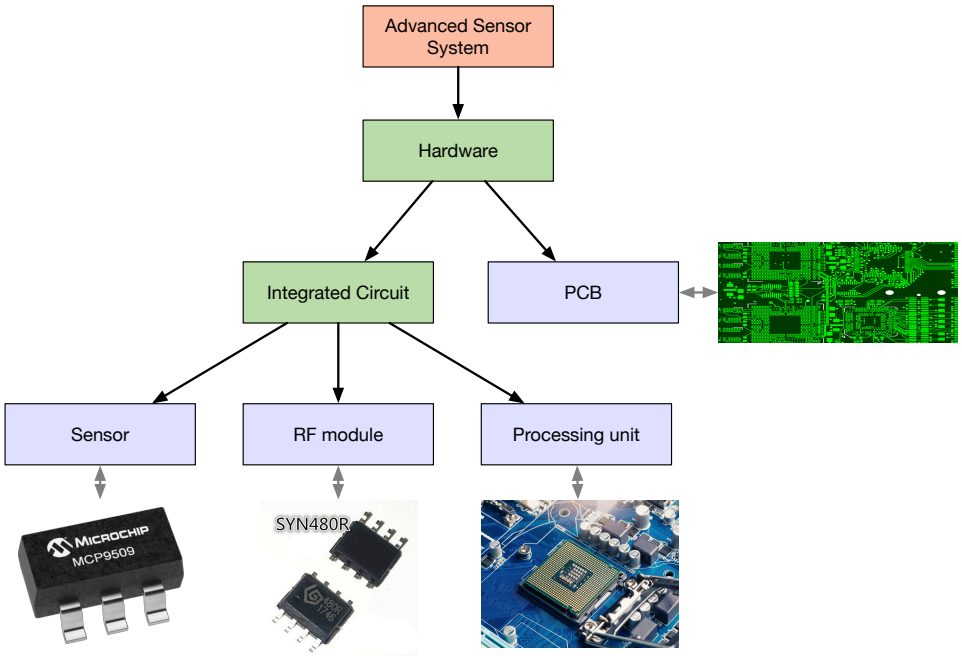


Figure 1.1 Advanced sensor system.

In order to achieve reliability and efficiency of sensor system, traditional design flow is shown in Fig. 1.2. The main steps contain the circuit topology design, placement, routing, post-layout simulation, verification, fabrication and system design. According to the system specifications, the engineers will design some circuits. The designed circuits consist of the circuit topology and design parameters. To commit the designs to manufacture, some advanced electronic design automation (EDA) tools are used to facilitate map design to the layout. Typically, this process contains placement and routing. Placement is placing

all instances or components on the layout without overlapping and satisfying some special constraints and design rules. While routing is to connect all the nets on the layout without any design rule violations. Note that this flow can be used in analog circuits, digital circuits and PCB designs. To achieve hardware reliability, reliability verification needs to be performed after the design is finished while before the design is fabricated. Besides, once the hardware is fabricated, reliability is only achieved in the system level, such as calibration. When all hardware designs are finished, some advanced applications are employed in the processing unit to control the whole system. Recently, convolutional neural networks (CNNs) have achieved impressive successes in various applications. They are also used to calibrate sensor [124], data analysis and prediction and estimation. To achieve hardware efficiency, these applications, including CNNs needs to be well customized.

1.1 Challenges

In spite of the great successes in the advanced sensor system, there are also various emerging challenges in hardware reliability and efficiency. The issues can be categorized into two parts.

Firstly, traditional design flow rarely focuses on the reliability, which causes unreliable hardware circuits. These unreliable hardware circuits have a shorter lifetime so that it ultimately brings a fateful consequence, such as measurement errors. Thus

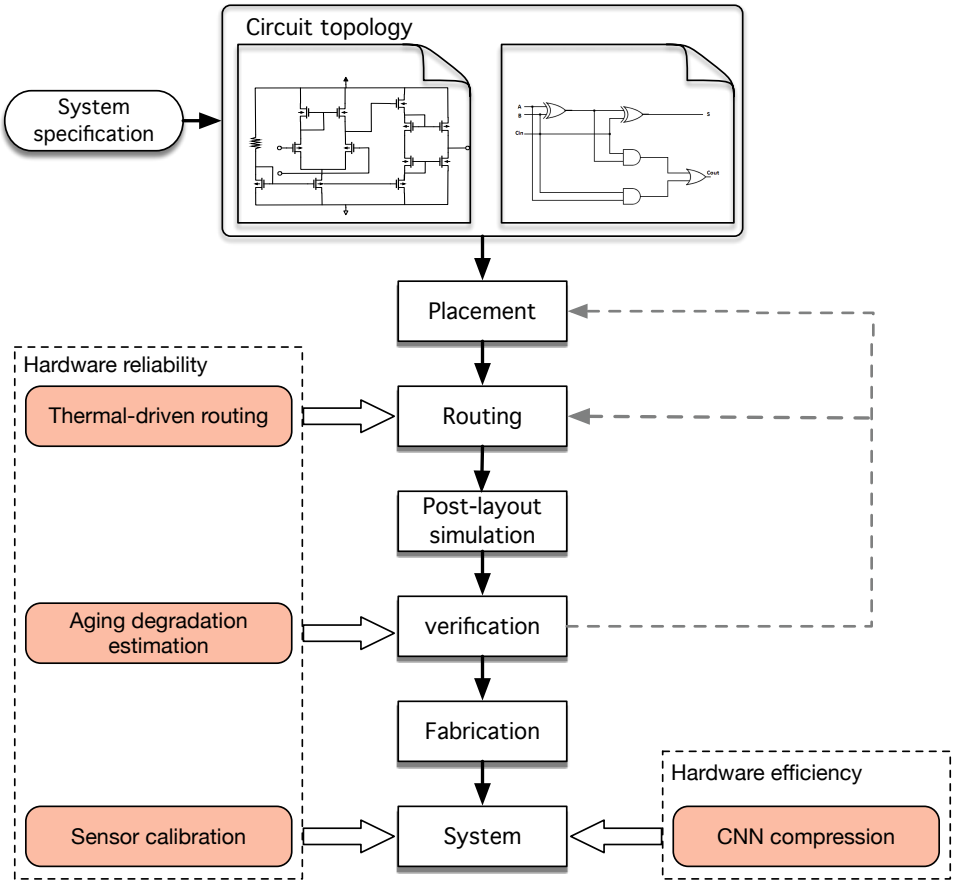


Figure 1.2 Design flow.

measurements need to be calibrated to extend the lifetime of sensors when sensors have unreliable effects. However, sensor calibration is hard to achieve perfect measurement reconstruction while replacing frequently unreliable hardware circuits bring extra cost. In order to achieve a good reliability, reliability needs be considered in design, verification and application stages. However, reliability verification is very time-consuming since some computational expensive analytic models are implemented in these reliability simulation. In addition, reliability

do not be considered in the design stage so that it is difficult to achieve design closure.

Secondly, the development of CNNs has catalyzed many innovative applications. However, large model size hinders the broad deployment of CNNs on resource-constrained hardware since data transfer will consume runtime and hardware resource. In order to fully exploit the capability of CNNs, designing hardware-efficiency learning models become essential. Particularly, since CNNs has covered a wide range of applications and scenarios, the methodologies of enabling hardware-efficiency learning should contain general techniques so that CNNs can be performed in hardware with high efficiency.

1.2 Thesis Overview

This thesis attempts to investigate several methodologies lightening the aforementioned issues mainly at hardware reliability and efficiency.

Our first contribution is calibrating measurement errors caused by sensor aging issue. Sensor drift is an intractable obstacle to practical temperature measurement in the intelligent electronic system. In this thesis, we propose a sensor spatial correlation model. Given prior knowledge, Maximum-a-posteriori (MAP) estimation is performed to calibrate drifts. MAP is formulated as a non-convex problem with three hyper-parameters. An alternating-based method is proposed to solve this non-convex

formulation. Cross-validation, Gibbs Expectation-maximum (EM) and variational Bayesian EM are further exploited to determine hyper-parameters. Experimental results on widely used benchmarks from the simulator EnergyPlus demonstrate that compared with state-of-the-art methods, the proposed framework can achieve a robust drift calibration and a better trade-off between accuracy and runtime. On average, compared with state-of-the-art, the proposed framework can achieve about 3× accuracy improvement. In order to attain the same drift calibration accuracy with variational Bayesian EM, Gibbs EM needs 10000 samples, which will incur a 30× runtime overhead.

The second contribution of the thesis is improving the efficiency of reliability verification. With continued scaling, the transistor aging induced by hot carrier injection (HCI) and bias temperature instability (BTI) causes an increasing failure of nanometer-scale ICs. Compared with digital ICs, analog ICs are more susceptible to aging effects. The industrial large-scale analog ICs bring grand challenges in the efficiency of aging verification. In this thesis, we propose a heterogeneous graph convolutional network (H-GCN) to fast estimate aging-induced transistor degradation in analog ICs. To characterize the multi-typed devices and connection pins, a heterogeneous directed multigraph is adopted to efficiently represent the topology of analog ICs. A latent space mapping method is used to transform the feature vector of all typed devices into a unified latent space. We further extend the proposed H-GCN to be a deep version via ini-

tial residual connections and identity mappings. The extended deep H-GCN can extract information from multi-hop devices without an over-smoothing issue. A probability-based neighborhood sampling method on the bipartite graph is adopted to ease the model training on large-scale graphs and achieve good scalability. Experiments on very advanced $5nm$ industrial benchmarks show that, compared with traditional graph learning methods and static aging reliability simulations by an industrial design-for-reliability (DFR) tool, the proposed deep H-GCN can achieve more accurate estimations of aging-induced transistor degradation. Compared with the dynamic and static aging reliability simulations, our extended deep H-GCN on average can achieve 241× and 39× speedup, respectively.

The third contribution is integrating thermal reliability into design stage. The thermal effect has an impact on the reliability of PCB layout. Routing is one of the key steps to reduce the PCB layout heat. The traditional verification-then-fix approaches are ill-equipped when faced with the ever-growing violations of thermal limits. In this thesis, we propose TRouter, a thermal-driven PCB routing framework via CNNs. CNNs are leveraged to predict thermal distribution by taking the routing layout as an input. A gradient in each routing grid cell obtained from the backpropagation (BP) of CNNs is integrated into a full-board routing algorithm to guide thermal-driven routing. To achieve a significant speedup, an adaptive-size bounding-box is adopted to reduce routing search space. We conduct experi-

ments on open-source benchmarks to illustrate our TRouter can achieve significant speedup and thermal well layouts, comparing with state-of-the-art PCB routing algorithm.

Our fourth contribution is designing a hardware-efficiency convolution. Compared with traditional convolutions, grouped convolutional neural networks are promising for both model performance and network parameters. However, existing models with the grouped convolution still have parameter redundancy. In this thesis, concerning the grouped convolution, we propose a sharing grouped convolution structure to reduce parameters. To efficiently eliminate parameter redundancy and improve model performance, we propose a Bayesian sharing framework to transfer the vanilla grouped convolution to be the sharing structure. Intra-group correlation and inter-group importance are introduced into the prior of the parameters. We handle the Maximum Type II likelihood estimation problem of the intra-group correlation and inter-group importance by a group LASSO type algorithm. The prior mean of the sharing kernels is iteratively updated. Extensive experiments are conducted to demonstrate that on different grouped convolutional neural networks, the proposed sharing grouped convolution structure with the Bayesian sharing framework can reduce parameters and improve prediction accuracy. The proposed sharing framework can reduce parameters up to 64.17%. For ResNeXt-50 with the sharing grouped convolution on ImageNet dataset, network parameters can be reduced by 96.875% in all grouped convolutional layers,

and accuracies are improved to 78.86% and 94.54% for top-1 and top-5.

The structure of the thesis is organized as follows. Chapter 2 provides a comprehensive review about the works in this thesis. Chapter 3 introduces the first contribution with corresponding technique details. Chapter 4 illustrates the the second contribution. Chapter 5 gives an illustration about the third contribution and the last contribution is shown in Chapter 6. Chapter 7 concludes this thesis.

□ **End of chapter.**

Chapter 2

Literature Review

In this chapter, we will review some related literatures about our works proposed in this thesis. This thesis works on hardware reliability and efficiency. It contains the sensor calibration, analog aging degradation estimation, thermal-driven PCB routing, and grouped convolution neural networks compression. Next we will give literature review from these aspects.

2.1 Sensor calibration

Although advanced technologies in the semiconductor industry and micro-electromechanical systems are developed in recent years, in practice, sensors outputs exist errors, which are one of the major barriers to the use of sensor networks. There are three main types of errors: gain, drift and noise [92]. Compared with gain and noise, the sensor drift is considered with vital importance since it has a significantly negative effect on measurement accuracy in aged sensors [2]. Although sensors with

high accuracy and reliability can be deployed, these sensors always have expensive prices.

The sensor calibration is studied in many previous research works. These works can be classified into micro- and macro-calibration or non-blind, semi-blind and blind calibration [110, 151]. The micro-calibration is a cumbersome method that each sensor is individually tuned so that the ground-truth data can be recovered from measurements [89].

In the macro-calibration schemes, data from uncalibrated sensors are collected to optimize the overall system performance. Therefore, macro-calibration schemes are widely used in practice. For macro-calibration schemes, according to information whether they need, there are three categories of sensor calibration: non-blind, semi-blind and blind. In the non-blind calibration, one or more than one prior knowledge is adopted to calibrate measurements (e.g., the ground-truth data measured by some of sensors with high accuracy [32] and the distances between sensors and the sink node [129]). However, extra costs are required to obtain this information. For example, several sensors with high accuracy have to be deployed to obtain ground truth data, and a Global Position System module is installed in sensors.

In order to reduce extra costs, some works focused on semi-blind calibration schemes. In [53], only partial position information and the ground-truth data are required to calibrate all measurements. In order to further decrease extra costs, blind

calibration schemes were proposed.

In literature about the blind calibration, there are two models to estimate measurement errors: the first-order autoregression (AR) model and the signal space projection (SSP) model. For the AR model, assuming that errors are a time series following Gaussian distribution with zero-mean and constant variance, adaptive filters such as Kalman filter [72, 126], unscented Kalman [116] filter and particle filter are adopted to track the error in each time slot so that the measurement can be calibrated [116, 126]. In the AR model, however, it is assumed that only one sensor has measurement error in each time instant. In fact, this assumption is hard to satisfy in practice.

The calibration problem is naturally studied extensively to be a sparse reconstruction problem, where a sparse set of sensors are assumed to have significant drifts. Balzano and Nowak [15] first proposed SSP, in which the measured data are over-sampled by sensors. In other words, the sensor number is more than the variance source number. Therefore, the ground-truth vector has one or more than one null-space. Assuming that sensors are free-aged and free-error in the initial time, this null space can be obtained by principal component analysis or independent component analysis [30]. In addition, assuming that measurement errors are sparse, the error can be estimated by sparse regression techniques, such as the maximum likelihood estimation [125] and deep learning [124]. In [125], Wang et al. adopt temporal sparse Bayesian learning (TSBL) [156] to calibrate time-variant and

incremental drifts for the sparse set of sensors. However, due to the sparsity assumption, not all sensors can be calibrated. In addition, since the observation matrix is directly determined by drift-free measurement, the method cannot calibrate drifts if signals lie in a time-variant subspace.

Very recently, in order to calibrate all sensors, Ling and Strohmer presented three models, which are formulated as bilinear inverse problems [75]. However, these models heavily rely on partial information about the sensing matrix. For the temperature sensor calibration in a smart building, the sensing matrix depends on the weather, the position of sensors and parameters of the building, e.g., material characteristics, geometry and equipment power per area [25, 26, 73]. In practice, it is hard to obtain this complex and tedious information. As a result, these models cannot be directly used to calibrate temperature sensors in a smart building.

According to the discussion above, previous works cannot handle the situation where dense aged sensors have measurement errors meanwhile there is no partial information, such as weather and material, provided for calibration.

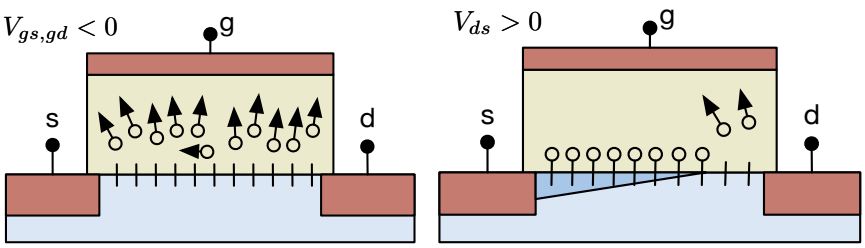
2.2 Analog aging degradation estimation

With continued scaling, the susceptibility of nanometer-scale transistors to aging-related wear-out phenomena has increased significantly in ICs [105]. As illustrated in Fig. 2.1, two primary

aging-related wear-out mechanisms of semiconductor based microelectronic devices are BTI and HCI [145]. BTI mechanism is that accumulated holes in silicon/oxide interface result in breaking of Si-H bonds as shown in Fig. 2.1(a). HCI mechanism is that due to high electric field in drain side, hot carriers cause breaking of Si-H bond and traps oxide bulk as shown in Fig. 2.1(b) [16]. These aging effects cause transistor parameters, e.g., threshold voltage, to shift from their nominal values over time, resulting in a gradual circuit failure. Compared with digital ICs, analog ICs are more susceptible to these transistor parameters.

In order to save development costs and provide the opportunity for interactive feedback during the design process, estimating aging-induced transistor degradation before committing the design to silicon is a key step. In the industry, according to the estimated aging-induced transistor degradation, the aging violations can be judged. If there are one or more aging violations in the aging verification, as a typical verification-then-fix approach, the DFR designer will go back to the design stages to fix them. Besides, compared with pre-layout netlists, post-layout netlists contain parasitic capacitances and resistances. Thus the aging reliability simulation on post-layout netlists has more accurate judgment. However, post-layout netlists have a larger scale size than pre-layout netlists so that they bring grand challenges in the efficiency of aging verification.

In the industry, as shown in Fig. 2.2, a traditional aging reliability simulator is adopted to verify the circuit reliability. The



(a) (b)

Figure 2.1 BTI and HCI mechanisms.

typical aging reliability simulation is classified into static simulation and dynamic simulation [114]. They both take a circuit netlist and its stress conditions (stimuli) as inputs. A fresh simulation is performed to obtain the fresh transistor parameters. According to the fresh transistor parameters, the stress simulation is performed to get device degradations. By using the device degradations, the aging simulation is adopted to obtain the aging transistor parameters. At last, the shift values from fresh transistor parameters to the aging transistor parameters are output to judge the circuit reliability. Compared with the static simulation, the traditional dynamic simulation considers dynamic stress conditions such as clock speeds in the fresh simulation and the aging simulation stages so that it needs a large number of accepted transient steps. Thus the dynamic aging reliability simulation can obtain a more accurate aging-induced transistor degradation while takes more time. Nevertheless, the accuracy of traditional dynamic aging reliability simulation heavily relies on the given dynamic stress conditions.

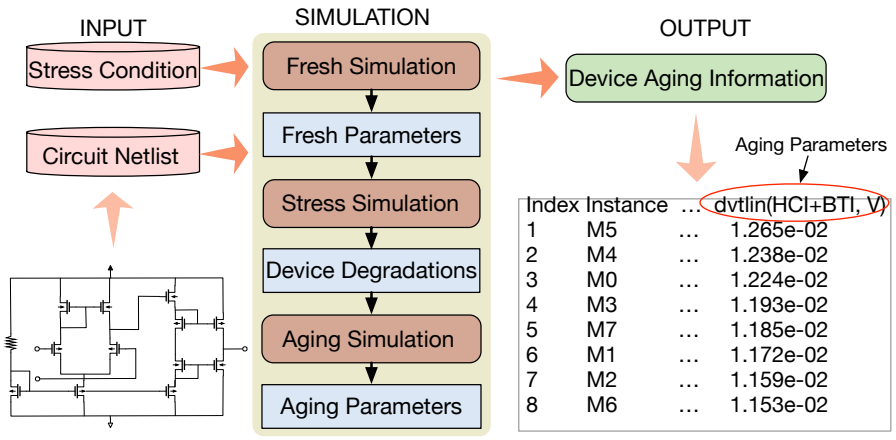


Figure 2.2 The traditional aging reliability simulation.

Modeling aging-induced transistor degradation has been widely studied in the literature. Tu et al. adopted a voltage-controlled current source model to predict HCI issues in the design stage [117]. A transistor drain-current surrogate model was developed to explore the effects of degradation on analog circuits [144]. Other analytical models were also surveyed and concluded in [16]. However, there are also several limitations and drawbacks to these analytical models. Firstly, a correct judgment on the reliability of the circuits heavily relies on the appropriate stress conditions [104]. The static aging reliability simulation causes inaccurate judgment on the aging-prone transistors since the dynamic stress conditions are completely ignored. Secondly, it is time-consuming to achieve accurate detections since the dynamic aging simulation needs a large number of accepted transient steps. It is usually difficult to find a compromise between computational complexity and model accuracy.

CNNs, as a data-driven approach, have achieved great successes in circuit design, such as the design-for-manufacturability [24, 36]. Typically, the aging reliability simulations are performed on circuit netlists. Intuitively, an analog IC netlist can be naturally represented as a graph. However, the graph is irregular grid-based data, which is not as straightforward as the convolution and pooling in traditional CNNs. GCNs were proposed to perform machine learning tasks on these irregular grid-based data [40, 60]. Recently, GCNs were adopted to predict observation point candidates on the design-for-testability [86], select timing models [85], estimate layout parasitics [99], annotate netlists [35, 64], guide placement [71] and tune transistor sizing [120]. However, the analog IC netlists have heterogeneity since typical analog ICs contain multi-typed basic devices (e.g., transistor, resistor) and multi-typed connection pins (e.g., drain, gate). Typically, exiting GCNs adopt one-hot encoding to distinguish among multi-typed basic devices and connection pins. However, it will miss some structural information among multi-typed nodes as well as unstructured content associated with each node [20].

Despite GCN can achieve enormous successes in the graph learning task, it is very shallow. Such a shallow model limits its ability to extract information from multi-hop devices. Simply stacking more layers and adding non-linearity may cause performance degradation, due to the over-smoothing phenomenon [70]. Very recently, several arts try to tackle the over-smoothing issue.

Jumping knowledge networks (JKNets) combine the output of each layer to the last layer to keep the local properties of the node representations [137]. A few edges are randomly removed from the input graph in DropEdge to alleviate the over-smoothing issue [101]. Simplifying GCNs captures higher-order features in the graph by using the k-th power of the graph convolution in each neural network layer [131]. Personalized PageRank is generalized to an arbitrary graph diffusion process in graph diffusion convolution [61,62]. GCNs with initial residual connections and identity mappings (GCNII) ensure that the final representation of each node retains at least a fraction from the input layer, even if many layers are stacked [22]. Nevertheless, these models do not consider the heterogeneity of the graph.

According to the discussion above, the typical homogeneous GCNs and their shallow structure brings performance degradation in the analog aging degradation estimation.

2.3 Thermal-driven PCB routing

The thermal effect has an impact on the reliability of the PCB layout. Specifically, overheat temperatures on the PCB layout cause the reduction of the mean time between failures [18]. Routing is one of the key steps to reduce the PCB layout temperatures since the thermal distribution is sensitive to locations of the segment and via in the post-routing layout. In order to achieve PCB reliability, the traditional verification-then-fix

approaches are used in the traditional PCB design flow.

In the traditional PCB design flow, many advanced routing algorithms are adopted to automatically route all nets in both academia and industry. The PCB routing algorithms are classified into the escape routing and the area routing [139]. They both assume that the PCB designs contain only ball grid array (BGA) packages. The escape routing is to route the I/O pads or solder bumps on a die or package to the lines that can escape to the area surrounding the die to be routed out of the package or its immediate surroundings [140]. While the area routing is to connect the previously escaped routes of I/O pads and usually subject to an upper/lower bound of routed length for each connection [94]. The escape routing can be further categorized into the ordered escape and the unordered escape. The former needs to route the connections with specific ordering on the boundary while the latter needs not. Based on the traditional escape routing and area routing, some excellent arts were proposed to route all nets under considering complicated constraints or differential pairs [31,141]. However, escape routing and area routing may be inappropriate for dealing with irregularities of non-BGA. Very recently, a unified routing framework was proposed to handle such designs with the routing of power/ground nets, differential pairs, signal nets and irregularities of non-BGA packages together [74].

After a PCB layout is well-designed, some advanced and complex analytical models in the thermal simulation are used to

verify the thermal distribution. The thermal simulation provides a chance for interactive feedback for reliability during the design process. In these analytical models, the thermal distribution can be calculated by solving the heat equations [153, 154]. This requires assuming the amount of heat generated by different components, pads, vias and segments then calculating the temperature increase in different regions of the board by solving the steady-state heat equations. However, the traditional verification-then-fix approaches are ill-equipped when faced with the ever-growing violations of thermal limits. In other words, as shown in Fig. 2.3, if there are one or more thermal violations detected by thermal simulation, the designers will go back to the previous routing step to fix them. Thus the traditional verification-then-fix approaches have a low efficiency to achieve a thermal well PCB layout and reach design closure.

In order to improve the efficiency of design and verification, several analytical performance models were integrated into a router to improve routing quality [28, 45]. However, the thermal model is very complex so that it is difficult to directly integrate it into a router.

Recently, CNNs are adopted for fast predictions and estimations after the knowledge is acquired from large amounts of historical data. In computer-aided design (CAD) field, CNNs were leveraged to detect reliability violations [136, 159] and predict routability [135]. However, these detections and predictions cannot provide any guidance for previous design stages. Very

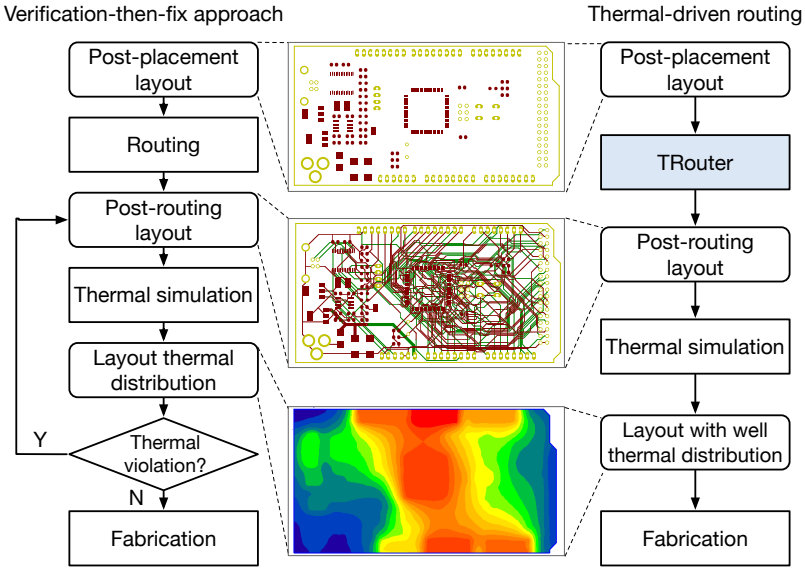


Figure 2.3 The typical verification-then-fix routing approach and our thermal-driven routing flow.

recently, CNNs-based performance models were used to guide performance-driven placement [77, 78]. However, the typically simulated annealing methodology brings expensive computation since the inference of the CNN-based performance model has to be performed in each search step [71]. Moreover, the convolutional generative adversarial networks were developed to predict the congestion heatmap, which was further used to avoid unnecessary searches and accelerating the overall routing process [160]. Besides, a generative neural network was proposed to provide routing guidance via mimicking the sophisticated manual layout approaches [161]. However, these routers do not be directly driven by specific layout performance or reliability. Since the relationship between routing and thermal distribu-

tion is extremely complex. In other words, the high temperature area is not straightforwardly set as a keepout or large cost area. While other factors such as neighborhood environments and thermal dissipation ability need to be considered. Thus the routing cannot be simply guided by predicted thermal distribution obtained from inference of CNN-based performance model, as previous works [160, 161]. Thus, these routers do not be directly driven by specific layout performance or reliability. To our best knowledge, none of the prior art handles the performance- or reliability-driven routing problem via deep learning or CNNs.

2.4 Grouped convolutional neural networks compression

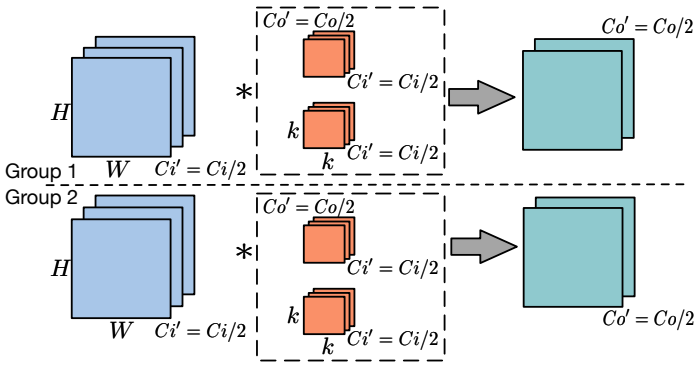
CNNs have achieved impressive successes in various applications of computer vision, such as object recognition [43, 76], object detection [36, 38, 79], and video analysis [142]. To handle complicated applications, CNN models become deeper and wider, which causes massive network parameters. The massive network parameters, however, bring huge challenges to model storage, data transfer, computation overhead, and energy consumption [113, 143]. Besides, the massive network parameters may contain redundancy, which causes overfitting and performance degradation.

The grouped convolution has been adopted to decrease parameter redundancy and improve accuracy in popular CNNs,

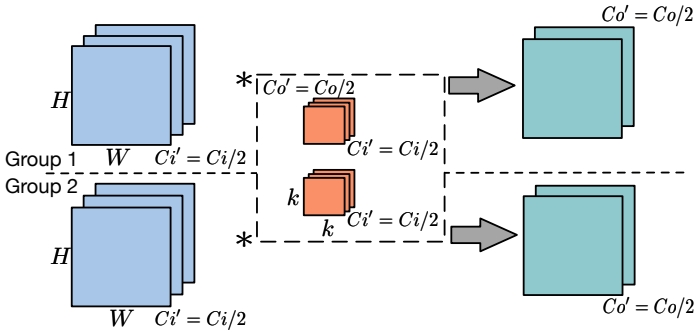
such as AlexNet [63] and ResNeXt [134]. The vanilla grouped convolution is shown in Fig. 2.4(a), where the inputs, the weights, and the outputs are divided into several groups to perform the convolution operation. In practice, the grouped convolution is proven to be able to alleviate overfitting and improve the model accuracy, outperforming its counterpart, e.g., non-grouped ResNet vs. grouped ResNeXt [134, 155]. Moreover, the grouped convolution is also proven to be more efficient and effective than wider and deeper networks [134].

Although the grouped convolution has the aforementioned advantages, the network parameters may still have redundancy. Various arts are proposed to reduce parameter redundancy. These methods can be classified into two types: model compression methods and architecture design methods [123]. Although existing compression methods have good compression performance in the traditional convolution models, they may lead to performance degradations while being applied to grouped convolutions since they ignore the diversities of importances and correlations (i.e., inter-group importance and intra-group correlation) among the different parameter groups. Without specific optimization techniques, directly training models with these group architectures may also degrade the performance.

The wider and deeper CNN models bring great challenges to model storage, computation, data communication, and system power consumption [66]. Therefore, many model compression methods were proposed to address these challenges. Con-



(a)



(b)

Figure 2.4 The vanilla grouped convolution and our proposed sharing grouped convolution (2 groups, the blue boxes are the input features, the orange boxes are the kernels, the green boxes are the output features. H and W are height and weight of the input features. Ci and Co are the numbers of the input and output channels. Ci' and Co' are the numbers of the input and output channels in each group. k is kernel size): (a) The vanilla grouped convolution. Each group has its own weights. (b) Our proposed sharing grouped convolution. All of these groups share the same weights.

sequently, the model compression can reduce the inference runtime. The inference runtime consists of memory access and computation [113, 143], where memory access is usually the runtime bottleneck [127].

Typical model compression methods can be categorized into model pruning, bit quantization, low-rank approximation, and knowledge distillation [147]. Model pruning methods can be used to prune parameters in different manners, e.g., channel-wise and depth-wise prunings [42,44,47,69,80,128,150], or structural and non-structural prunnings [119, 121]. In particular, attention statistics were adopted to evaluate the importance of channels so that channels can be pruned by the evaluation [138]. The low-rank approximation and the model parameters sparsification can accelerate inference and reduce model storage [21, 23, 57, 67, 84, 90, 93]. Due to the redundancy of the data precision, bit quantization approaches learn low-bit representations of features and parameters [41, 130]. Therefore, bit quantization approaches are very useful for model deployment tasks on domain-specific hardware [143]. Different from other categories, knowledge distillation methods facilitate the training of lightweight models by using knowledge learned from large networks [21, 57].

Although these compression methods have good compression performance in the traditional convolution, they may lead to performance degradation for the grouped convolution since they ignore the diversities of inter-group importance and intra-group correlation among the parameter groups.

Considering the aforementioned drawbacks in model compression methods, some works adopt other ways to design efficient architectures to improve network performance. Various

parameter normalization layers were proposed to avoid performance degradation, such as batch normalization, switchable normalization, exemplar normalization [54, 83, 106, 107, 148]. However, these normalization schemes cannot remove parameter redundancy even make the networks cumbersome. Besides, some tricky neural architecture search methods are proposed to determine network configurations [56]. Some works replace large filters with smaller ones [43, 48]. In order to further eliminate redundancies, some arts adopt separable convolution [68], that is replacing a 3D convolution with multiple 2D convolutions. For example, a 3D convolution is factorized to be two 2D ones in Inception V3 [115]. The pre-defined sparse 2D kernels are used to make a trade-off between accuracy and energy consumption [66]. The depthwise separable convolution is adopted in MobileNets [46] and Xception [27].

In particular, the grouped convolution is an efficient architecture outperforming its counterpart, e.g., non-grouped ResNet vs. grouped ResNeXt [134, 155]. The grouped convolution was firstly proposed in AlexNet [63], which allocates models on two GPUs to facilitate parallelism. The grouped convolution adopts the sparse convolution connections between input and output channels, by dividing the input channels, output channels, and their connections into several groups as shown in Fig. 2.4(a). Compared with the traditional convolutions with "fully connected" features and weights, the parameters and computation costs are reduced.

The successes of grouped convolution have inspired its wide applications, e.g., ShuffleNet [152] and CondenseNet [48]. Recently, to improve the model accuracy, an interleaved grouped convolutions were designed to further improves parameter efficiency and classification accuracy [149]. Wu et al. proposed a group normalization layer, where the mean and variance are computed within each group [133]. A dynamic grouped convolution was designed with different numbers of channels in the same layer [112,155]. A fully learnable grouped convolution was proposed to freely choose the grouping strategy in the training stage [123]. The optimal channel permutation was developed to explore group configuration [158]. However, the correlation among parameters and groups does not be considered so that these methods will cause performance degradation.

According to the discussion, these is no art about grouped convolution compression.

2.5 Summary

According to our literature review, there are several challenges in hardware reliability and efficiency:

- Previous sensor calibration works cannot handle the situation where dense aged sensors have measurement errors meanwhile there is no partial information, such as weather and material, provided for calibration;

- The typical homogeneous GCNs and their shallow structure brings performance degradation in the analog aging degradation estimation;
- None of the prior art handles the performance- or reliability-driven routing problem via deep learning or CNNs to achieve design closure;
- These is no art about grouped convolution compression by considering the correlation among parameters and groups.

In this thesis, we will propose methodologies to handle these challenges to achieve hardware reliability and efficiency.

□ **End of chapter.**

Chapter 3

Sensor Calibration

3.1 Preliminary

In smart building, several low-cost sensors are deployed to sense in-building temperatures. Furthermore, all sensors are non-removable once they are deployed in smart building. Besides, due to a slow-aging effect, all sensors have unknown time-invariant drifts. As shown in Fig. 3.1, unlike communication channels [156], for a sensor signal to be output, i.e., current (I_{out}), it is contaminated by a time-invariant drift. Sensor 1 has a drift ϵ_1 so that its transfer function shifts downward denoted by the blue line in Fig. 3.1. While Sensor 2 has a drift ϵ_2 so that its transfer function shifts upward denoted by the red line in Fig. 3.1. In order to achieve high-accurate measurements, drifts need to be estimated and calibrated. Based on the above description, we define the sensor drift calibration problem as follows.

Problem 1 (Sensor Drift Calibration)**.** *Given the measurement*

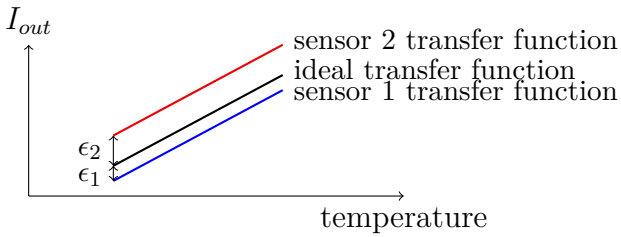


Figure 3.1 Drift vs. temperature [10].

values sensed by all sensors during several time-instants, drifts will be accurately estimated and calibrated.

3.2 Mathematical Formulation

We assume that n sensors are deployed to sense in-building temperatures in a smart building. During a short time after new sensors are deployed, drifts are assumed to be insignificant in all sensors. Furthermore, as in [125], we assume that all sensors are drift-free during m_0 initial time-instants. Due to over-sampling, as illustrated in [15,125], signals measured by sensors lie in a low dimensional subspace. Furthermore, in a smart building, all actual temperatures measured by sensors have a high correlation, for example, the dense deployment of sensors. Therefore, we build a linear model among all actual temperatures as follows:

$$x_i^{(k)} \approx \sum_{j=1, j\neq i}^{n} a_{i,j} x_j^{(k)} + a_{i,0}, \qquad k = 1, 2, \cdots, m_0, \tag{3.1}$$

where $x_i^{(k)}$ is the ground-truth temperature sensed by ith sensor at kth time-instant. $a_{i,j}$ is the drift-free model coefficient. We

define $\boldsymbol{x} \triangleq [x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}, \cdots, x_n^{(m_0)}]^\top$, $\boldsymbol{a}_i \triangleq [a_{i,0}, \cdots, a_{i,i-1}, a_{i,i+1}, \cdots, a_{i,n}]^\top \in \mathbb{R}^n$, $\boldsymbol{a} \triangleq [\boldsymbol{a}_1^\top, \boldsymbol{a}_2^\top, \cdots, \boldsymbol{a}_n^\top]^\top \in \mathbb{R}^{n^2}$.

Due to a slow-aging effect, all sensors have unknown time-invariant drifts. For multiple measurements during a short period, all sensors are assumed to suffer a time-invariant drift. As shown in Fig. 3.1, unlike communication channels [156], electric signal output, e.g., current (I_{out}), by electronic devices causes a time-invariant drift. During m time-instants, Equation (3.1) is naturally extended as

$$\hat{x}_i^{(k)} + \epsilon_i \approx \sum_{j=1, j\neq i}^{n} \hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) + \hat{a}_{i,0}, k = 1, 2, \cdots, m, \quad (3.2)$$

where $\hat{x}_i^{(k)}$ is the measurement value sensed by ith sensor at kth time-instant. In particular, in order to obtain enough information, we assume $m_0, m > n$. For ith sensor, ϵ_i is a time-invariant drift calibration, which is independent of time-instant k. $\hat{a}_{i,j}$ is the model coefficient when all sensors have unknown time-invariant drifts. We vectorize these variables as $\hat{\boldsymbol{x}} \triangleq [\hat{x}_1^{(1)}, \hat{x}_2^{(1)}, \cdots, \cdots, \hat{x}_n^{(m)}]^\top$, $\hat{\boldsymbol{a}}_i \triangleq [\hat{a}_{i,0}, \cdots, \hat{a}_{i,i-1}, \hat{a}_{i,i+1}, \cdots, \hat{a}_{i,n}]^\top \in \mathbb{R}^n$, $\hat{\boldsymbol{a}} \triangleq [\hat{\boldsymbol{a}}_1^\top, \hat{\boldsymbol{a}}_2^\top, \cdots, \hat{\boldsymbol{a}}_n^\top]^\top \in \mathbb{R}^{n^2}$ and $\boldsymbol{\epsilon} \triangleq [\epsilon_1, \epsilon_2, \cdots, \epsilon_n]^\top \in \mathbb{R}^n$.

Note that Equation (3.2) is the essential in our proposed sensor spatial correlation model. Furthermore, the model error in Equation (3.2) is assumed to follow identical independent Gaussian distribution with zero-mean and unknown precision (inverse variance) δ_0. Therefore, the likelihood function $\mathcal{P}(\hat{\boldsymbol{x}}|\hat{\boldsymbol{a}}, \boldsymbol{\epsilon})$ is de-

fined as follows:

$$\mathcal{P}(\hat{\boldsymbol{x}}|\hat{\boldsymbol{a}},\boldsymbol{\epsilon}) \propto \exp(-\frac{\delta_0}{2}\sum_{i=1}^{n}\sum_{k=1}^{m}[\hat{x}_i^{(k)}+\epsilon_i - \sum_{j=1,j\neq i}^{n}\hat{a}_{i,j}(\hat{x}_j^{(k)}+\epsilon_j)-\hat{a}_{i,0}]^2). \tag{3.3}$$

However, the likelihood function $\mathcal{P}(\hat{\boldsymbol{x}}|\hat{\boldsymbol{a}},\boldsymbol{\epsilon})$ cannot be directly used to calibrate drifts using maximum likelihood estimation (MLE) since it has not enough information. Therefore, we need give two priors in development.

For all sensors, drifts are assumed to follow identical independent Gaussian distribution with zero-mean and unknown precision $\delta_{\boldsymbol{\epsilon}}$ as follows:

$$\mathcal{P}(\boldsymbol{\epsilon}) \propto \exp(-\frac{\delta_{\boldsymbol{\epsilon}}}{2}\sum_{i=1}^{n}\epsilon_i^2). \tag{3.4}$$

In addition, we assume that the model coefficient $\hat{a}_{i,j}$ follows identical independent Gaussian distribution. Intuitively, $\hat{a}_{i,j}$ has high dependency on $a_{i,j}$ in statistics. Furthermore, the probability density function of $\hat{a}_{i,j}$ is assumed to take a maximum value at $a_{i,j}$. Therefore, the prior mean of $\hat{a}_{i,j}$ is $a_{i,j}$. In addition, in order that each model coefficient $\hat{a}_{i,j}$ is provided with a relatively equal probability to deviate from the corresponding drift-free model coefficient $a_{i,j}$, the precision of model coefficient $\hat{a}_{i,j}$ is defined to be $\lambda a_{i,j}^{-2}$, where λ is a nonnegative hyper-parameter to control the precision of $\hat{a}_{i,j}$. Therefore, each model coefficient $\hat{a}_{i,j}$ follows identical independent Gaussian distribution with $a_{i,j}$

mean and $\lambda a_{i,j}^{-2}$ precision [51, 52, 118]. For all model coefficients, we have

$$\mathcal{P}(\hat{\boldsymbol{a}}) \propto \exp\left(-\sum_{i=1}^{n}\sum_{j=0,j\neq i}^{n}\frac{\lambda}{2a_{i,j}^2}(\hat{a}_{i,j}-a_{i,j})^2\right). \tag{3.5}$$

This prior manner is named Bayesian Model Fusion, which was developed to combine the early-stage information and the late-stage information using Bayesian inference in Computer-Aided Design applications [51, 52, 118].

In order to calibrate drifts for all sensors, the posterior $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}})$ needs to be maximized in MAP estimation manner. According to Bayes' rule, the posterior $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}})$ can be expressed by two priors and the likelihood function as follows

$$\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}) \propto \mathcal{P}(\hat{\boldsymbol{x}}|\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}) \cdot \mathcal{P}(\hat{\boldsymbol{a}}) \cdot \mathcal{P}(\boldsymbol{\epsilon}). \tag{3.6}$$

Taking the logarithm, MAP can be transferred to be the equivalent formulation as follows:

$$\begin{aligned}\min_{\hat{\boldsymbol{a}},\boldsymbol{\epsilon}} \quad & \delta_0 \sum_{i=1}^{n}\sum_{k=1}^{m}[\hat{x}_i^{(k)} + \epsilon_i - \sum_{j=1,j\neq i}^{n}\hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) - \hat{a}_{i,0}]^2 \\ & + \lambda\sum_{i=1}^{n}\sum_{j=0,j\neq i}^{n}\frac{1}{a_{i,j}^2}(\hat{a}_{i,j}-a_{i,j})^2 + \delta_{\boldsymbol{\epsilon}}\sum_{i=1}^{n}\epsilon_i^2.\end{aligned} \tag{3.7}$$

To calibrate these unknown drifts in all sensors, Formulation (3.7) will be optimized efficiently. Next, we will propose an efficient alternating-based method to optimize Formulation (3.7).

3.3 Alternating-based Optimization

There are two challenges for Formulation (3.7): how to handle Formulation (3.7) and how to induce hyper-parameters λ, δ_0 and $\delta_{\boldsymbol{\epsilon}}$. Formulation (3.7) is a non-convex problem, thus it is difficult to obtain an optimal solution. In this section, we propose a fast and efficient alternating-based optimization methodology to optimize Formulation (3.7) by alternatively updating in each iteration.

According to the alternating-based methodology, at each iteration, the values of $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$ are updated by optimizing Formulation (3.7) w.r.t. $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$. Furthermore, note that with fixed drift calibration variable $\boldsymbol{\epsilon}$, Formulation (3.7) w.r.t. $\hat{\boldsymbol{a}}$ is regarded as a convex unconstrained Quadratic Programming (QP) problem, which can be solved by Gaussian elimination [39]. However, the computational complexity of Gaussian elimination is $\mathcal{O}(n^6)$ (n is the sensor number) if all model coefficients $\hat{\boldsymbol{a}}$ are calculated in one sub-formulation [39]. Consider that Formulation (3.7) w.r.t. $\hat{\boldsymbol{a}}$ can be decomposed into n independent sub-Formulations w.r.t. $\hat{\boldsymbol{a}}_i$. In order to reduce computational complexity, instead of calculating all model coefficients $\hat{\boldsymbol{a}}$ in one sub-formulation, $\hat{\boldsymbol{a}}_i$ will be calculated in ith sub-formulation ($i = 1, 2, \cdots, n$). As a result, the computational complexity of Gaussian elimination is $\mathcal{O}(n^4)$ in total for n sub-formulations. Formulation (3.7) w.r.t. $\hat{\boldsymbol{a}}$ is decomposed into n independent

sub-formulations w.r.t. $\hat{\boldsymbol{a}}_i$ as follows:

$$\begin{aligned}\min_{\hat{\boldsymbol{a}}_i} \quad & \delta_0 \sum_{k=1}^{m} [\hat{x}_i^{(k)} + \epsilon_i - \sum_{j=1, j\neq i}^{n} \hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) - \hat{a}_{i,0}]^2 \\ & + \lambda \sum_{j=0, j\neq i}^{n} \frac{1}{a_{i,j}^2}(\hat{a}_{i,j} - a_{i,j})^2,\end{aligned} \tag{3.8}$$

with the first-order optimality condition:

$$\delta_0 \sum_{k=1}^{m} (\hat{x}_t^{(k)} + \epsilon_t) \left[\sum_{j=1}^{n} \hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) + \hat{a}_{i,0} \right] + \lambda \frac{(\hat{a}_{i,t} - a_{i,t})}{a_{i,t}^2} = 0, \tag{3.9}$$

where $t = 0, 1, \cdots, i-1, i+1, \cdots, n$. In particular, we define $\hat{a}_{i,i} \triangleq -1$ and $\hat{x}_0^{(k)} + \epsilon_0 \triangleq 1$. The system of linear equations (3.9) can be addressed by Gaussian elimination [39].

In the same manner, with fixed model coefficients $\hat{\boldsymbol{a}}$, Formulation (3.7) w.r.t. the drift calibration $\boldsymbol{\epsilon}$ can also be regarded to be a convex unconstrained QP problem as follows:

$$\begin{aligned}\min_{\boldsymbol{\epsilon}} \quad & \delta_0 \sum_{i=1}^{n} \sum_{k=1}^{m} [\hat{x}_i^{(k)} + \epsilon_i - \sum_{j=1, j\neq i}^{n} \hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) - \hat{a}_{i,0}]^2 \\ & + \delta_{\boldsymbol{\epsilon}} \sum_{i=1}^{n} \epsilon_i^2,\end{aligned} \tag{3.10}$$

with the corresponding first-order optimality condition:

$$\delta_0 \sum_{i=1}^{n} \sum_{k=1}^{m} \left[\hat{a}_{i,t} (\sum_{j=1}^{n} \hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) + \hat{a}_{i,0}) \right] + \delta_{\boldsymbol{\epsilon}} \epsilon_t = 0, \tag{3.11}$$

where $t = 1, 2, \cdots, n$.

A local optimum can be obtained by the proposed alternating-based method while the convergence speed and solution quality

Algorithm 1 Alternating-based Method

Require: Sensor measurements $\hat{\boldsymbol{x}}$, prior $\boldsymbol{a}$ and hyper-parameters λ, δ_0, $\delta_{\boldsymbol{\epsilon}}$.

1: Initialize $\hat{\boldsymbol{a}} \leftarrow \boldsymbol{a}$ and $\boldsymbol{\epsilon} \leftarrow \mathbf{0}$;
2: **repeat**
3: **for** $i \leftarrow 1$ to n **do**
4: Fix $\boldsymbol{\epsilon}$, solve the system of linear equations (3.9) using Gaussian elimination to update $\hat{\boldsymbol{a}}_i$;
5: **end for**
6: Fix $\hat{\boldsymbol{a}}$, solve the system of linear equations (3.11) using Gaussian elimination to update $\boldsymbol{\epsilon}$;
7: **until** Convergence
8: **return** $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$.

depend on the initialization of variables. In our proposed framework, two priors are given for model coefficients $\hat{\boldsymbol{a}}$ and drift calibration $\boldsymbol{\epsilon}$. Therefore, in order to achieve a better convergence speed and solution quality, the prior means $\boldsymbol{a}$ and $\mathbf{0}$ are used to initialize variables $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$. We continue to update $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$ until convergence. The convergence condition is that the relative difference of drift calibration $\boldsymbol{\epsilon}$ between current and previous iterations is less than a threshold. In summary, our proposed alternating-based method is shown in Algorithm 1. In fact, if the quadratic objective functions (3.7) w.r.t. $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$ are strictly convex with lower-bounded Hessians, the proposed alternating-based method can achieve convergency of Formulation (3.7) [37, 87].

3.4 Estimation of Hyper-Parameters

It is important to induce the aforementioned three hyper-parameters so that drifts can be accurately calibrated and meanwhile the over-fitting can be avoided [91]. In this section, Cross-validation, Gibbs EM and variational Bayesian EM are utilized to induce hyper-parameters, respectively.

3.4.1 Unsupervised Cross-validation

Cross-validation is a simple method to select hyper-parameters [33]. Although there are three hyper-parameters λ, δ_0, $\delta_{\boldsymbol{\epsilon}}$ in Formulation (3.7), instead of determined by cross-validation individually, we design two ratios λ/δ_0 and $\delta_{\boldsymbol{\epsilon}}/\delta_0$ which demand certain values. As shown in Fig. 3.2, we partition temperature measurements during m time-instants into s non-overlapping parts. Given each combination of ratios candidates λ/δ_0 and $\delta_{\boldsymbol{\epsilon}}/\delta_0$, in each run, one of the s parts is exploited to estimate the model error and the rest $s-1$ parts are used to calculate model coefficients and drift calibration. In addition, different groups will be selected for error estimation in different runs. In the same manner, each run gives a model error e_r $(r = 1, 2, \cdots, s)$ estimated from a part of temperature measurements. The model error is defined as follows.

$$e_r \triangleq \sum_{k=\frac{(r-1)m}{s}+1}^{\frac{rm}{s}} [\hat{x}_i^{(k)} + \epsilon_i - \sum_{j=1, j\neq i}^{n} \hat{a}_{i,j}(\hat{x}_j^{(k)} + \epsilon_j) - \hat{a}_{i,0}]^2. \quad (3.12)$$

The final model error is computed as the average $\bar{e} = \sum_{r=1}^{s} e_r/s$. Then two ratios λ/δ_0 and δ_ϵ/δ_0 corresponding to the minimum average model error are chosen.

The pseudo-code of the unsupervised Cross-validation is shown in Algorithm 2. We input sensor measurements $\hat{\boldsymbol{x}}$, the drift-free model coefficients $\boldsymbol{a}$, the number of folds for Cross-validation s and several hyper-parameters candidates. Then the sensor measurements are split into s non-overlapping parts as illustrated in Fig. 3.2. Formulation (3.7) on model training set is optimized by Algorithm 1 and the model error defined in Equation (3.12) on the validation set is calculated for each run and each candidate of hyper-parameters. After the grid search on all hyper-parameters and sensor measurements $\hat{\boldsymbol{x}}$ is finished, the model error is averaged on s runs. At last, the hyper-parameters with the least average model error are chosen to output. The unsupervised Cross-validation flow is summarized in Fig. 3.3.

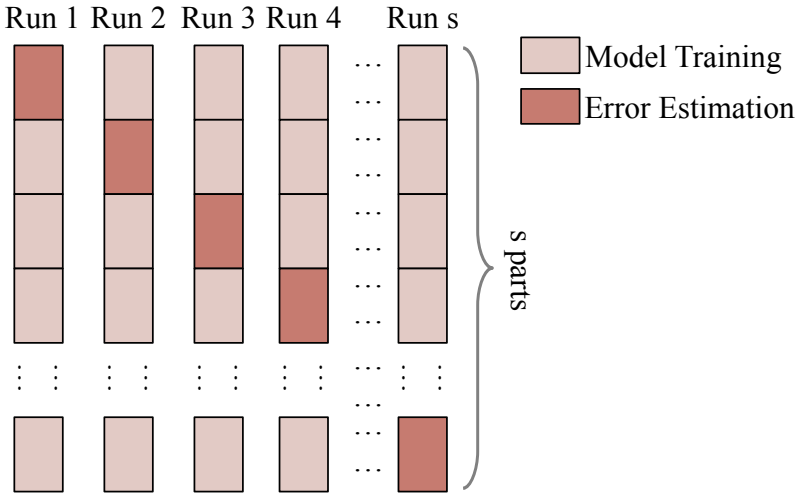


Figure 3.2 The unsupervised cross-validation.

Algorithm 2 The unsupervised cross-validation

Require: Sensor measurements $\hat{\boldsymbol{x}}$, prior $\boldsymbol{a}$, number of folds for cross-validation s, hyper-parameters candidates $\{(\lambda/\delta_0)_1, (\lambda/\delta_0)_2, \cdots, (\lambda/\delta_0)_{d_{\lambda/\delta_0}}\}$ and $\{(\delta_{\boldsymbol{\epsilon}}/\delta_0)_1, (\delta_{\boldsymbol{\epsilon}}/\delta_0)_2 \cdots, (\delta_{\boldsymbol{\epsilon}}/\delta_0)_{d_{\delta_{\boldsymbol{\epsilon}}/\delta_0}}\}$.

1: **for** $r \leftarrow 1$ to s **do**
2: **for** $i \leftarrow 1$ to d_{λ/δ_0} **do**
3: **for** $j \leftarrow 1$ to $d_{\delta_{\boldsymbol{\epsilon}}/\delta_0}$ **do**
4: Obtain the model coefficients $\hat{\boldsymbol{a}}$ and calibration $\boldsymbol{\epsilon}$ by Algorithm 1 from sensor measurements with rth part removed.
5: Compute the model error on rth part of sensor measurements by Equation (3.12).
6: **end for**
7: **end for**
8: **end for**
9: Average the computed modeling error for each pair of hyper-parameters candidates $(\lambda/\delta_0)_i$ and $(\delta_{\boldsymbol{\epsilon}}/\delta_0)_j$, i.e., $\bar{e} \leftarrow \sum_{r=1}^{s} e_k/s$.
10: Select $(\lambda/\delta_0)_i$ and $(\delta_{\boldsymbol{\epsilon}}/\delta_0)_j$ with the smallest modeling error, *i.e.*, $(\lambda/\delta_0)_{opt}, (\delta_{\boldsymbol{\epsilon}}/\delta_0)_{opt} \leftarrow \underset{}{\text{argmin}} \quad \bar{e}$.
11: **return** The optimal hyper-parameters $(\lambda/\delta_0)_{opt}$ and $(\delta_{\boldsymbol{\epsilon}}/\delta_0)_{opt}$.

Note that unlike conventional Cross-validation [25, 26, 51, 52, 73, 118], not any golden values are used in metrics to choose hyper-parameters in model fitting stage. Therefore, in our proposed framework, Cross-validation is adopted in an unsupervised-learning-like fashion.

Cross-validation is time-consuming since Algorithm 1 has to be performed for multiple times. Thus we propose two fast and efficient EM algorithms to determine hyper-parameters in statistical model.

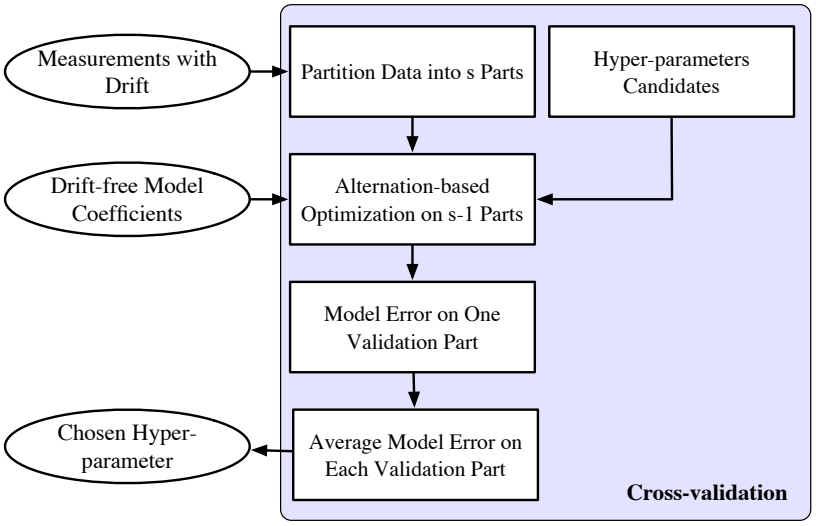


Figure 3.3 The unsupervised cross-validation flow.

3.4.2 Gibbs Expectation Maximization

In this section, MLE is used to determine individual hyper-parameters δ_0, λ and $\delta_{\boldsymbol{\epsilon}}$. MLE of hyper-parameters is formulated as follows:

$$\max_{\delta_{\boldsymbol{\epsilon}},\delta_0,\lambda} \quad \mathcal{P}(\hat{\boldsymbol{x}};\delta_0,\lambda,\delta_{\boldsymbol{\epsilon}}). \tag{3.13}$$

However, in the likelihood function $\mathcal{P}(\hat{\boldsymbol{x}};\delta_0,\lambda,\delta_{\boldsymbol{\epsilon}})$, the integral is intractable. EM algorithm is leveraged to efficiently find a solution to Formulation (3.13) [146]. According to EM algorithm, Formulation (3.13) can be taken the logarithm and transferred to be its auxiliary lower bound function [34]. Then the auxiliary lower-bound function is optimized by **E-step** and **M-step** iteratively after the term independent of hyper-parameters is omitted. The detailed derivation can be found in [100]. For convenience, all hyper-parameters are collected as a set $\Omega \triangleq \{\delta_0,\lambda,\delta_{\boldsymbol{\epsilon}}\}$.

Expectation Step with Gibbs Sampling

Since $\mathcal{P}(\hat{\boldsymbol{x}}; \delta_0, \lambda, \delta_{\boldsymbol{\epsilon}}) = \mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda) + \mathcal{KL}(\mathcal{Q}||\mathcal{P})$ and $\mathcal{KL}(\mathcal{Q}||\mathcal{P}) \geq 0$, $\mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda)$ is a lower-bound function defined as follows:

$$\mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda) = \mathbb{E}_{\mathcal{Q}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon})} \ln \frac{\mathcal{P}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{a}}, \boldsymbol{\epsilon}; \delta_\epsilon, \delta_0, \lambda)}{\mathcal{Q}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon})}, \tag{3.14}$$

where $\mathbb{E}$ indicates the expectation operator. $\mathcal{Q}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon})$ is an arbitrary joint distribution for $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$. Instead of maximizing the marginal likelihood directly, the EM maximizes the lower bound function. Let' s assume that we can find $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$ analytically. Then, we can simply substitute $\mathcal{Q}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}) = \mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$. The lower-bound function can be represented as follows

$$\begin{aligned} \mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda) = \mathbb{E}_{\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})} \ln \mathcal{P}(\hat{\boldsymbol{X}}, \hat{\boldsymbol{a}}, \boldsymbol{\epsilon}; \delta_\epsilon, \delta_0, \lambda) \\ - \mathbb{E}_{\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})} \ln \mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}}). \end{aligned} \tag{3.15}$$

The second term $\mathbb{E}_{\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})} \ln \mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$ is a constant with respect to δ_ϵ and δ_0 and λ, and we do not take the term into account when maximizing the lower-bound function. Therefore, we define a quantity as follows

$$Q(\Omega|\Omega^{\text{old}}) = \mathbb{E}_{\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}, \boldsymbol{\epsilon}; \Omega), \tag{3.16}$$

where Ω^{old} denotes estimated hyper-parameters in the previous iteration.

However, the posterior $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\boldsymbol{x}; \Omega^{\text{old}})$ is intractable since it is hard to calculate the integral in the likelihood function $\mathcal{P}(\hat{\boldsymbol{x}}; \delta_0, \lambda, \delta_{\boldsymbol{\epsilon}})$. There are two main methods to approximate the posterior $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\boldsymbol{x}; \Omega)$: Variational Inference and *Markov Chain Monte Carlo* (MCMC)

[100]. Compared with Variational Inference, MCMC has the advantage of being non-parametric and asymptotically exact [103]. Therefore, Monte Carlo method is utilized to approximate the quantity defined in Equation (3.17) as follows:

$$Q(\Omega|\Omega^{\text{old}}) \approx \frac{1}{L}\sum_{l=1}^{L} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}^{(l)}, \boldsymbol{\epsilon}^{(l)}; \Omega), \tag{3.17}$$

where samples $\hat{\boldsymbol{a}}^{(l)}$ and $\boldsymbol{\epsilon}^{(l)}$ are sampled from the distribution $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$. L is the total number of samples. In MCMC, there are two main algorithms to obtain samples from the desired distribution $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$: Metropolis Hastings algorithm and Gibbs sampling. Since the rejection rate will be high in complex problems, Metropolis Hastings algorithm has a very slow convergence rate [100]. Therefore, Gibbs sampling is harnessed to obtain samples $\hat{\boldsymbol{a}}^{(l)}$ and $\boldsymbol{\epsilon}^{(l)}$.

Gibbs sampling has the behavior that one or batch variables are cyclically and repeatedly updated in some particular order at random from conditional distribution. Sampling order is arranged to be $\hat{a}_{1,0}^{(l)}, \cdots, \hat{a}_{1,n}^{(l)}, \hat{a}_{2,0}^{(l)}, \cdots, \hat{a}_{n,n-1}^{(l)}, \epsilon_1^{(l)}, \cdots, \epsilon_n^{(l)}$. In Gibbs sampling, one of key points is derivation of the conditional distribution for each variable. Note that according to Formulation (3.7), the logarithm conditional distribution w.r.t. individual variable is quadratic. Therefore, the conditional distribution of each variable is Gaussian distribution as follows:

$$\begin{aligned}\hat{a}_{p,q} &\sim \mathcal{P}(\hat{a}_{p,q}|\boldsymbol{\epsilon}, \hat{\boldsymbol{a}}_{/\hat{a}_{p,q}}, \hat{\boldsymbol{x}}; \delta_\epsilon, \delta, \lambda) = \mathcal{N}(\hat{a}_{p,q}; \mu_{\hat{a}_{p,q}}, \sigma^{-1}_{\hat{a}_{p,q}}),\\ \epsilon_t &\sim \mathcal{P}(\epsilon_t|\boldsymbol{\epsilon}_{/\epsilon_t}, \hat{\boldsymbol{a}}, \hat{\boldsymbol{x}}; \delta_\epsilon, \delta, \lambda) = \mathcal{N}(\epsilon_t; \mu_{\epsilon_t}, \sigma^{-1}_{\epsilon_t}),\end{aligned} \tag{3.18}$$

in agreement with Equation (3.4) and Equation (3.5). μ is mean and σ is precision. $\hat{\boldsymbol{a}}_{/\hat{a}_{p,q}}$ and $\boldsymbol{\epsilon}_{/\epsilon_t}$ denote $\hat{\boldsymbol{a}}$ but with $\hat{a}_{p,q}$ omitted and $\boldsymbol{\epsilon}$ but with ϵ_t omitted, respectively. In particular, we define $\hat{a}^{(s)}_{i,i} \triangleq -1$, $\hat{x}^{(k)}_0 + \epsilon^{(s)}_0 \triangleq 1$ and $p \neq q$. The mean and precision of each variable are given as follows:

$$\sigma_{\epsilon_t} = m\delta_0 \sum_{i=1}^{n} \hat{a}^{(s)2}_{i,t} + \delta_{\boldsymbol{\epsilon}}, \tag{3.19}$$

$$\sigma_{\hat{a}_{p,q}} = \delta_0 \sum_{k=1}^{m} (\hat{x}^{(k)}_q + \epsilon^{(s)}_q)^2 + \frac{\lambda}{a^2_{p,q}}, \tag{3.20}$$

$$\mu_{\epsilon_t} = \frac{\delta_0}{\sigma_{\epsilon_t}} \sum_{k=1}^{m} \sum_{i=1}^{n} \hat{a}^{(s)}_{i,t} [\sum_{j=0}^{n} \hat{a}^{(s)}_{i,j} (\hat{x}^{(k)}_j + \epsilon^{(s)}_j) - \hat{a}^{(s)}_{i,t} \epsilon^{(s)}_t], \tag{3.21}$$

and

$$\begin{aligned}\mu_{\hat{a}_{p,q}} = \frac{\delta_0}{\sigma_{\hat{a}_{p,q}}} \sum_{k=1}^{m} (\hat{x}^{(k)}_q + \epsilon^{(s)}_q) [\sum_{j=0}^{n} \hat{a}^{(s)}_{p,j} (\hat{x}^{(k)}_j + \epsilon^{(s)}_j)\\ - \hat{a}^{(s)}_{p,q} (\hat{x}^{(k)}_q + \epsilon^{(s)}_q)] + \frac{\lambda}{\sigma_{\hat{a}_{p,q}} a_{p,q}}.\end{aligned} \tag{3.22}$$

Before Gibbs sampling, in order to converge to the desired posterior, the warm-start has to be performed if there is no reasonable initialization for samples. Furthermore, it is very hard to judge whether the warm-start is enough [100]. In order to waive the warm-start, a reasonable initialization for samples is adopted in Gibbs sampling. Note that Gibbs sampling is used to

obtain samples from the desired posterior $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$ (6.6). As we discussed in Section 5.1, Formulation (3.7) is equivalent to MAP estimation of $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$. Thus, given hyper-parameters Ω^{old} and measurement values $\hat{\boldsymbol{x}}$, Gibbs sampling can be initialized by handling Formulation (3.7) to obtain initial samples $\hat{\boldsymbol{a}}^{(0)}$ and $\boldsymbol{\epsilon}^{(0)}$ which satisfy the distribution $\mathcal{P}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\hat{\boldsymbol{x}}; \Omega^{\text{old}})$. As a result, the warm-start can be totally waived.

Maximization Step

After L samples are obtained by Gibbs sampling, in **M-step**, we will maximize the approximated quantity as follows:

$$\max_{\Omega} \quad \frac{1}{L}\sum_{l=1}^{L} \ln\mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}^{(l)}, \boldsymbol{\epsilon}^{(l)}; \Omega). \tag{3.23}$$

With the first-order optimality condition, namely $dQ/d\Omega = 0$, hyper-parameters λ, δ_0, $\delta_{\boldsymbol{\epsilon}}$ can be updated as follows:

$$\lambda = \frac{n^2 L}{\sum_{i=1}^{n}\sum_{j=0,j\neq i}^{n}\sum_{l=1}^{L}\frac{(\hat{a}_{i,j}^{(l)}-a_{i,j})^2}{a_{i,j}^2}}, \tag{3.24}$$

$$\delta_0 = \frac{Lmn}{\sum_{l=1}^{L}\sum_{i=1}^{n}\sum_{k=1}^{m}[\sum_{j=1}^{n}\hat{a}_{i,j}^{(l)}(\hat{x}_j^{(k)}+\epsilon_j^{(l)})+\hat{a}_{i,0}^{(l)}]^2}, \tag{3.25}$$

$$\delta_{\boldsymbol{\epsilon}} = \frac{nL}{\sum_{l=1}^{L}\sum_{i=1}^{n}\epsilon_i^{(l)2}}. \tag{3.26}$$

Here, $\hat{a}_{i,i}^{(l)} \triangleq -1$ and $\hat{x}_0^{(k)} + \epsilon_0^{(l)} \triangleq 1$. We continue to alternate between E-step and M-step until convergence. The convergence condition is that the relative difference of three hyper-parameters between current and previous iterations is less than a threshold. Then hyper-parameters λ, δ, $\delta_{\boldsymbol{\epsilon}}$ can be determined.

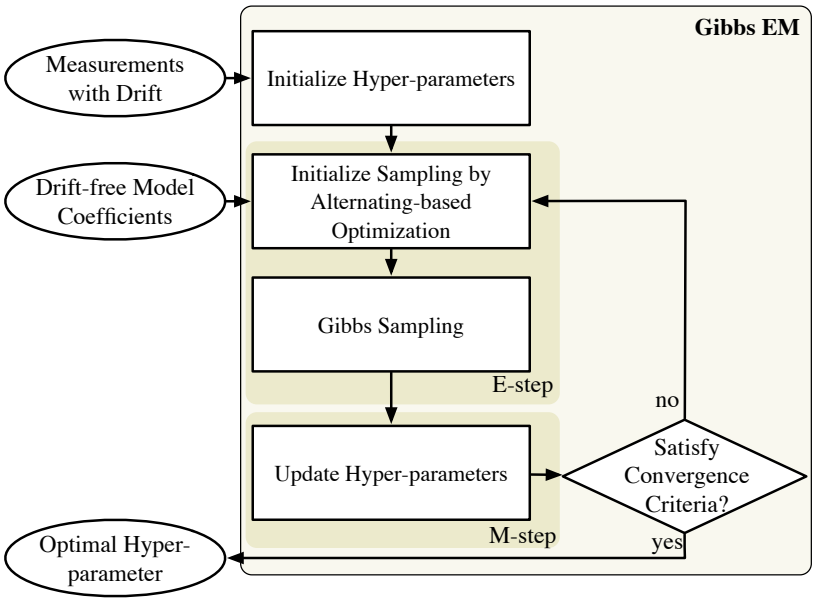


Figure 3.4 The Gibbs EM flow.

For convenience, all variables are collected as a set $\Psi \triangleq \{\psi_1, \psi_2, \ldots, \psi_{n^2+n}\} = \{\hat{a}_{1,0}, \cdots, \hat{a}_{1,n}, \cdots, \hat{a}_{n,n-1}, \epsilon_1, \cdots, \epsilon_n\}$. The pseudo-code of the Gibbs EM is concluded in Algorithm 3. Given $\hat{\boldsymbol{x}}$, $\boldsymbol{a}$, and initialized hyper-parameters as inputs, the Gibbs EM is performed iteratively. In E-step, samples are initialized by Algorithm 1 to waive warm start. Then in Gibbs sampling, each samples are cyclically and repeatedly obtained by the desired conditional distributions defined in Equation (3.18). Once there are enough samples acquired by Gibbs sampling, hyper-parameters are updated by Equations (3.24) to (3.26) in M-step. E-step and M-step are iteratively performed until the convergence criteria is satisfied. At last, we have the optimized hyper-parameters. For a better understanding, our proposed Gibbs EM is displayed in Fig. 3.4.

Algorithm 3 Gibbs EM

Require: Sensor measurements $\hat{\boldsymbol{x}}$, prior $\boldsymbol{a}$;

Initialize hyper-parameters Ω;
repeat
  Initialize samples $\Psi^{(0)}$ by Algorithm 1;
  for $l \leftarrow 1$ to L **do**
    for $i \leftarrow 1$ to $n^2 + n$ **do**
      Sample $\psi_i^{(l)}$ from the desired conditional distribution $\mathcal{N}(\psi_i; \mu_{\psi_i}, \sigma_{\psi_i})$ (3.18) with $\psi_1^{(l)}, \cdots, \psi_{i-1}^{(l)}, \psi_{i+1}^{(l-1)}, \cdots, \psi_{n^2+n}^{(l-1)}$;
    end for
  end for
  Update hyper-parameters Ω by Equations (3.24) to (3.26);
until Convergence
return hyper-parameters Ω.

3.4.3 Variational Bayesian Expectation Maximization

In practice, MCMC sampling method is computationally demanding. As a result, it often limits its use to small-scale problems [100]. Besides, it can be difficult to judge whether the sampling method is generating independent samples from the desired distribution. In this section, we develop a deterministic approximation scheme which scales well to large applications.

We can even enforce full independence between the model parameters $\hat{\boldsymbol{a}}$ and the calibration $\boldsymbol{\epsilon}$ given measurements $\hat{\boldsymbol{x}}$. This assumption, known as the mean field approximation [95], allows us to compute the update rules for $\hat{\boldsymbol{a}}$ and the calibration $\boldsymbol{\epsilon}$ in isolation.

According to the mean field approximation, in Equation (3.14),

let $\mathcal{Q}(\hat{\boldsymbol{a}}, \boldsymbol{\epsilon}) = \mathcal{Q}(\hat{\boldsymbol{a}})\mathcal{Q}(\boldsymbol{\epsilon})$. Then, the lower-bound function $\mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda)$ can be factorized into $\hat{\boldsymbol{a}}$ and $\boldsymbol{\epsilon}$.

$$\begin{aligned}
&\mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda) \\
=& -\mathcal{KL}(\mathcal{Q}(\hat{\boldsymbol{a}})||\tilde{\mathcal{P}}(\boldsymbol{\epsilon})) - \mathbb{E}_{\mathcal{Q}(\boldsymbol{\epsilon})} \ln \mathcal{Q}(\boldsymbol{\epsilon}) + c \qquad (3.27)\\
=& -\mathcal{KL}(\mathcal{Q}(\boldsymbol{\epsilon})||\tilde{\mathcal{P}}(\hat{\boldsymbol{a}})) - \mathbb{E}_{\mathcal{Q}(\hat{\boldsymbol{a}})} \ln \mathcal{Q}(\hat{\boldsymbol{a}}) + c, \qquad (3.28)
\end{aligned}$$

where c is a constant to adjust $\tilde{\mathcal{P}}(\hat{\boldsymbol{a}})$ or $\tilde{\mathcal{P}}(\boldsymbol{\epsilon})$ to become a proper probability density function as follows:

$$\tilde{\mathcal{P}}(\hat{\boldsymbol{a}}) \triangleq \frac{1}{Z} \exp[\mathbb{E}_{\mathcal{Q}(\boldsymbol{\epsilon})} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}; \lambda, \delta_0, \delta_{\boldsymbol{\epsilon}})], \qquad (3.29)$$

$$\tilde{\mathcal{P}}(\boldsymbol{\epsilon}) \triangleq \frac{1}{Z} \exp[\mathbb{E}_{\mathcal{Q}(\hat{\boldsymbol{a}})} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}; \lambda, \delta_0, \delta_{\boldsymbol{\epsilon}})], \qquad (3.30)$$

where Z is a normalized constant. Since Kullback–Leibler divergence $\mathcal{KL}(\mathcal{Q}(\hat{\boldsymbol{a}})||\tilde{\mathcal{P}}(\boldsymbol{\epsilon})) \geq 0$, the lower-bound function $\mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda)$ is maximized w.r.t. $\mathcal{Q}(\hat{\boldsymbol{a}})$ when $\mathcal{KL}(\mathcal{Q}(\hat{\boldsymbol{a}})||\tilde{\mathcal{P}}(\boldsymbol{\epsilon})) = 0$, which happens when $\mathcal{Q}(\hat{\boldsymbol{a}}) = \tilde{\mathcal{P}}(\boldsymbol{\epsilon})$. Similarity, because $\mathcal{KL}(\mathcal{Q}(\boldsymbol{\epsilon})||\tilde{\mathcal{P}}(\hat{\boldsymbol{a}})) \geq 0$, the lower-bound function $\mathcal{L}(\mathcal{Q}; \delta_\epsilon, \delta_0, \lambda)$ is maximized w.r.t. $\mathcal{Q}(\boldsymbol{\epsilon})$ if $\mathcal{KL}(\mathcal{Q}(\boldsymbol{\epsilon})||\tilde{\mathcal{P}}(\hat{\boldsymbol{a}})) = 0$, which happens when $\mathcal{Q}(\boldsymbol{\epsilon}) = \tilde{\mathcal{P}}(\hat{\boldsymbol{a}})$.

Therefore, in **variational E-step**, let

$$\ln \mathcal{Q}(\hat{\boldsymbol{a}}) = \mathbb{E}_{\mathcal{Q}(\boldsymbol{\epsilon})} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\lambda, \delta_0, \delta_{\boldsymbol{\epsilon}}), \qquad (3.31)$$

$$\ln \mathcal{Q}(\boldsymbol{\epsilon}) = \mathbb{E}_{\mathcal{Q}(\hat{\boldsymbol{a}})} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}, \boldsymbol{\epsilon}|\lambda, \delta_0, \delta_{\boldsymbol{\epsilon}}), \qquad (3.32)$$

where according to Formulation (3.7), the logarithm conditional distribution w.r.t. individual variable is quadratic. Therefore, the variational distribution of each variable is Gaussian distri-

bution as follows:

$$\mathcal{Q}(\hat{\boldsymbol{a}}) \sim \mathcal{N}(\hat{\boldsymbol{a}}; \mathbb{E}(\hat{\boldsymbol{a}}), \boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}}), \tag{3.33}$$

$$\mathcal{Q}(\boldsymbol{\epsilon}) \sim \mathcal{N}(\boldsymbol{\epsilon}; \mathbb{E}(\boldsymbol{\epsilon}), \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}), \tag{3.34}$$

where $\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}}$ is the covariance matrix. $\hat{\boldsymbol{a}}_i$ is independent to each other, since Formulation (3.7) w.r.t. $\hat{\boldsymbol{a}}$ can be decomposed into n independent sub-Formulations w.r.t. $\hat{\boldsymbol{a}}_i$. Therefore, $\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}} \triangleq \text{diag}[\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}_1}, \boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}_2}, \cdots, \boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}_n}]$. $\mathbb{E}(\hat{\boldsymbol{a}}) \triangleq [\mathbb{E}(\hat{\boldsymbol{a}}_1^\top), \mathbb{E}(\hat{\boldsymbol{a}}_2)^\top, \cdots, \mathbb{E}(\hat{\boldsymbol{a}}_n)^\top]^\top$. By combining coefficients of $\hat{\boldsymbol{a}}_i$ quadratic term in Equation (3.32), we can obtain

$$\begin{aligned}\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}_i}^{-1} = \delta_0(&\hat{\boldsymbol{X}}_{/i}^\top \hat{\boldsymbol{X}}_{/i} + \mathbb{E}(\boldsymbol{E}_{/i})^\top \hat{\boldsymbol{X}}_{/i} + \hat{\boldsymbol{X}}_{/i}^\top \mathbb{E}(\boldsymbol{E}_{/i}) \\ &+\mathbb{E}(\boldsymbol{E}_{/i}^\top \boldsymbol{E}_{/i})) + \lambda \boldsymbol{A}_i^\top \boldsymbol{A}_i,\end{aligned} \tag{3.35}$$

where $\boldsymbol{A}_i \triangleq \text{diag}[a_{i,0}, a_{i,1}, \cdots, a_{i,i-1}, a_{i,i+1}, \cdots, a_{i,n}]$,

$$\hat{\boldsymbol{X}}_{/i} \triangleq \begin{bmatrix} 1 & \hat{x}_1^{(1)} & \cdots & \hat{x}_{i-1}^{(1)} & \hat{x}_{i+1}^{(1)} & \cdots & \hat{x}_n^{(1)} \\ 1 & \hat{x}_1^{(2)} & \cdots & \hat{x}_{i-1}^{(2)} & \hat{x}_{i+1}^{(2)} & \cdots & \hat{x}_n^{(2)} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 1 & \hat{x}_1^{(m)} & \cdots & \hat{x}_{i-1}^{(m)} & \hat{x}_{i+1}^{(m)} & \cdots & \hat{x}_n^{(m)} \end{bmatrix}, \tag{3.36}$$

$$\boldsymbol{E}_{/i} \triangleq \begin{bmatrix} 0 & \epsilon_1 & \cdots & \epsilon_{i-1} & \epsilon_{i+1} & \cdots & \epsilon_n \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ 0 & \epsilon_1 & \cdots & \epsilon_{i-1} & \epsilon_{i+1} & \cdots & \epsilon_n \end{bmatrix}. \tag{3.37}$$

Hence, according to Equations (3.36) and (3.37), $\mathbb{E}(\boldsymbol{E}_{/i})$ depends on the mean vector $\mathbb{E}(\boldsymbol{\epsilon})$ and $\mathbb{E}(\boldsymbol{E}_{/i}^\top \boldsymbol{E}_{/i})$ depends on the autocorrelation matrix $\mathbb{E}(\boldsymbol{\epsilon}^\top \boldsymbol{\epsilon})$ and the mean vector $\mathbb{E}(\boldsymbol{\epsilon})$.

Combining coefficients of $\hat{\boldsymbol{a}}_i$ linear term in Equation (3.32), we can obtain the mean vector as follows

$$\mathbb{E}(\hat{\boldsymbol{a}}_i)^\top = \delta_0(\hat{\boldsymbol{x}}_i^\top + \mathbb{E}(\boldsymbol{\epsilon}_i)^\top)(\hat{\boldsymbol{X}}_{/i} + \mathbb{E}(\boldsymbol{E}_{/i}))\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}_i}, \tag{3.38}$$

where $\hat{\boldsymbol{x}}_i \triangleq [\hat{x}_i^{(1)}, \hat{x}_i^{(2)}, \cdots, \hat{x}_i^{(m)}]^\top$ and $\mathbb{E}(\boldsymbol{\epsilon}_i) \triangleq [\mathbb{E}(\epsilon_i), \mathbb{E}(\epsilon_i), \cdots, \mathbb{E}(\epsilon_i)]^\top \in \mathbb{R}^m$. Once the covariance matrix $\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}_i}^{-1}$ and mean vector $\mathbb{E}(\hat{\boldsymbol{a}}_i)$ are obtained, we can obtain the autocorrelation matrix $\mathbb{E}(\hat{\mathbf{a}}_i\hat{\mathbf{a}}_i^\top) = \boldsymbol{\Sigma}_{\hat{\mathbf{a}}_i} + \mathbb{E}(\hat{\mathbf{a}}_i)\mathbb{E}(\hat{\mathbf{a}}_i)^\top$.

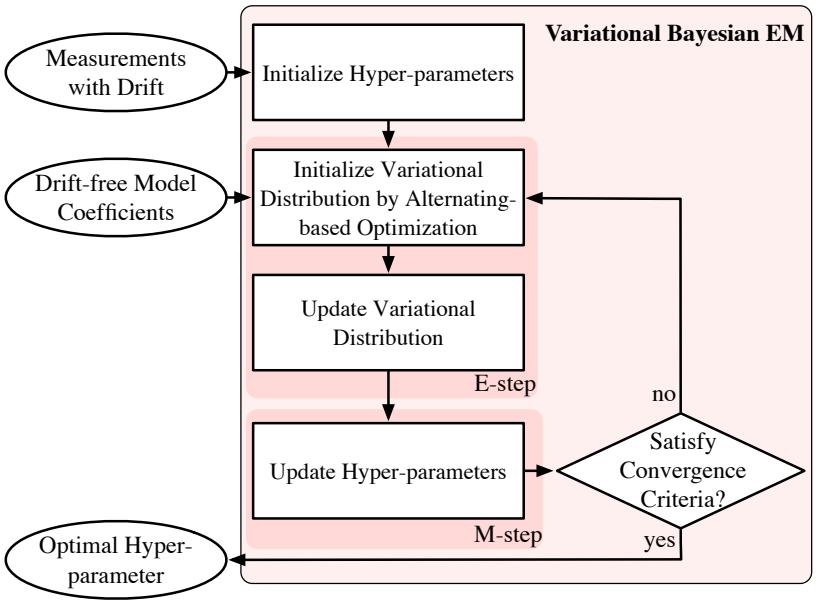


Figure 3.5 The variational Bayesian EM flow.

Likewise, by combining coefficients of $\boldsymbol{\epsilon}$ quadratic term in Equation (3.31), we can get the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$ and the mean vector $\mathbb{E}(\boldsymbol{\epsilon})$ in Equation (3.39) and Equation (3.40), which depend on the mean vector $\mathbb{E}(\hat{\boldsymbol{a}}_i)$ and the autocorrelation matrix $\mathbb{E}(\hat{\mathbf{a}}_i\hat{\mathbf{a}}_i^\top)$. Then the autocorrelation matrix $\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top)$ can be obtained by $\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top) = \boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} + \mathbb{E}(\boldsymbol{\epsilon})\mathbb{E}(\boldsymbol{\epsilon})^\top$.

$$\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1} = m\sum_{i=1}^{n}\begin{bmatrix} \mathbb{E}(\hat{a}_{i,1}\hat{a}_{i,1}) & \cdots & \cdots & \mathbb{E}(\hat{a}_{i,1}\hat{a}_{i,n}) \\ \mathbb{E}(\hat{a}_{i,2}\hat{a}_{i,1}) & \cdots & \cdots & \mathbb{E}(\hat{a}_{i,2}\hat{a}_{i,n}) \\ \vdots & \vdots & \vdots & \cdots \\ \mathbb{E}(\hat{a}_{i,i-1}\hat{a}_{i,1}) & \cdots & \cdots & \mathbb{E}(\hat{a}_{i,i-1}\hat{a}_{i,n}) \\ -\mathbb{E}(\hat{a}_{i,1}) & \cdots & \cdots & -\mathbb{E}(\hat{a}_{i,n}) \\ \mathbb{E}(\hat{a}_{i,i+1}\hat{a}_{i,1}) & \cdots & \cdots & \mathbb{E}(\hat{a}_{i,i+1}\hat{a}_{i,n}) \\ \vdots & \vdots & \vdots & \cdots \\ \mathbb{E}(\hat{a}_{i,n}\hat{a}_{i,1}) & \cdots & \cdots & \mathbb{E}(\hat{a}_{i,n}\hat{a}_{i,n}) \end{bmatrix} \tag{3.39}$$

$$\begin{aligned} \mathbb{E}(\boldsymbol{\epsilon}) =& (m\sum_{i=1}^{n}\Big[\mathbb{E}(\hat{a}_{i,0}\hat{a}_{i,1}) \quad \cdots \quad \mathbb{E}(\hat{a}_{i,0}\hat{a}_{i,i+1})\cdots\mathbb{E}(\hat{a}_{i,0}\hat{a}_{i,n})\Big] \\ &+\frac{1}{m}\sum_{k=1}^{m}(\boldsymbol{x}^{(k)})^{\top}\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}^{-1})\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}} \end{aligned} \tag{3.40}$$

Since model coefficients $\mathbf{a}$ and calibration $\boldsymbol{\epsilon}$ variational distributions are dependent on each other, by iteratively updating them, the variational distributions $\mathcal{Q}(\hat{\mathbf{a}})$ and $\mathcal{Q}(\boldsymbol{\epsilon})$ can be obtained to maximize the lower-bound function defined in Equation (3.28).

Once the optimal variational distributions $\mathcal{Q}(\hat{\mathbf{a}})$ and $\mathcal{Q}(\boldsymbol{\epsilon})$ can be obtained, unlike Formulation (3.23), in **variational M-step**, the objective function is shown as follows:

$$\max_{\Omega} \quad \mathbb{E}_{\mathcal{Q}(\hat{\boldsymbol{a}})\mathcal{Q}(\boldsymbol{\epsilon})} \ln \mathcal{P}(\hat{\boldsymbol{x}}, \hat{\boldsymbol{a}}, \boldsymbol{\epsilon}; \Omega). \tag{3.41}$$

With the first-order optimality condition, that is $dQ/d\Omega = 0$, hyper-parameters λ, δ_0, $\delta_{\boldsymbol{\epsilon}}$ can be updated as follows:

$$\lambda = \frac{n^2}{\sum_{i=1}^{n}\sum_{j=0,j\neq i}^{n}\frac{\mathbb{E}(\hat{a}_{i,j}^2)+a_{i,j}^2-2a_{i,j}\mathbb{E}(\hat{a}_{i,j})}{a_{i,j}^2}}, \tag{3.42}$$

$$\begin{aligned}\delta_0 = mn/\sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{p=0}^{n}\sum_{q=0}^{n}\mathbb{E}(\hat{a}_{i,p}\hat{a}_{i,q})[\hat{x}_p^{(k)}\hat{x}_q^{(k)} \\ + \hat{x}_p^{(k)}\mathbb{E}(\epsilon_q) + \hat{x}_q^{(k)}\mathbb{E}(\epsilon_p) + \mathbb{E}(\epsilon_p\epsilon_q)]\end{aligned} \tag{3.43}$$

$$\delta_{\boldsymbol{\epsilon}} = \frac{n}{\sum_{i=1}^{n}\mathbb{E}(\epsilon_i^2)}. \tag{3.44}$$

Compared with Equations (3.24) to (3.26), we can find in Equations (3.42) to (3.44), the variational distributions' mean vector and autocorrelation matrix replace samples, i.e., $\mathbb{E}(\epsilon_i) = \sum_{l=1}^{L}\epsilon_i^{(l)}/L$, $\mathbb{E}(\hat{a}_{i,j}) = \sum_{l=1}^{L}\hat{a}_{i,j}^{(l)}/L$, $\mathbb{E}(\epsilon_i\epsilon_j) = \sum_{l=1}^{L}\epsilon_i^{(l)}\epsilon_j^{(l)}/L$ and $\mathbb{E}(\hat{a}_{i,p}\hat{a}_{i,q}) = \sum_{l=1}^{L}\hat{a}_{i,p}^{(l)}\hat{a}_{i,q}^{(l)}/L$. Therefore, compared with Gibbs EM, the variational Bayesian EM are more stable since it belongs to deterministic methods while Gibbs EM is stochastic method.

Like Gibbs EM, we continue to alternate between variational E-step and variational M-step until convergence. The convergence condition is that the relative differences of three hyper-parameters between current and previous iterations are less than thresholds. Then hyper-parameters λ, δ, $\delta_{\boldsymbol{\epsilon}}$ can be determined.

The pseudo-code of Variational Bayesian EM is listed in Algorithm 4. Given $\hat{\boldsymbol{x}}$, $\boldsymbol{a}$, and initialized hyper-parameters, variational Bayesian EM is performed iteratively. In variational E-step, to initialize calibration variational distribution, the calibration mean and the autocorrelation matrix are obtained by Algorithm 1. Then calibration and model coefficients variational distributions are iteratively updated by Equations (3.35) and (3.38) to (3.40) until the convergence criteria is satisfied. In

variational M-step, hyper-parameters are updated according to Equations (3.42) to (3.44). The variational E-step and the variational M-step are iteratively performed until the convergence criteria is satisfied. Finally, the optimal hyper-parameters are obtained. For a further illustration, our proposed variational Bayesian EM flow is shown in Fig. 3.5.

Algorithm 4 Variational Bayesian EM

Require: Sensor measurements $\hat{\boldsymbol{x}}$, prior $\boldsymbol{a}$;

Initialize hyper-parameters Ω;
repeat
    Initialize calibration mean $\mathbb{E}(\boldsymbol{\epsilon}) = \boldsymbol{\epsilon}$ obtained by Algorithm 1 and the autocorrelation matrix $\mathbb{E}(\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top) = \mathbb{E}(\boldsymbol{\epsilon})\mathbb{E}(\boldsymbol{\epsilon})^\top$;
    repeat
        Update variational distribution $\mathcal{Q}(\hat{\boldsymbol{a}})$ with mean vector $\mathbb{E}(\hat{\boldsymbol{a}})$ and covariance matrix $\boldsymbol{\Sigma}_{\hat{\boldsymbol{a}}}$ by Equations (3.35) and (3.38);
        Update variational distribution $\mathcal{Q}(\boldsymbol{\epsilon})$ with mean vector $\mathbb{E}(\boldsymbol{\epsilon})$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\epsilon}}$ by Equations (3.39) and (3.40);
    until Convergence
    Update hyper-parameters Ω by Equations (3.42) to (3.44);
until Convergence
return hyper-parameters Ω.

3.5 Overall flow

The overall flow of our proposed sensor drift calibration is shown in Fig. 3.6, which consists of four parts: model optimization, Cross-validation, Gibbs EM and variational Bayesian EM. With drift-free measurements model coefficients and several temper-

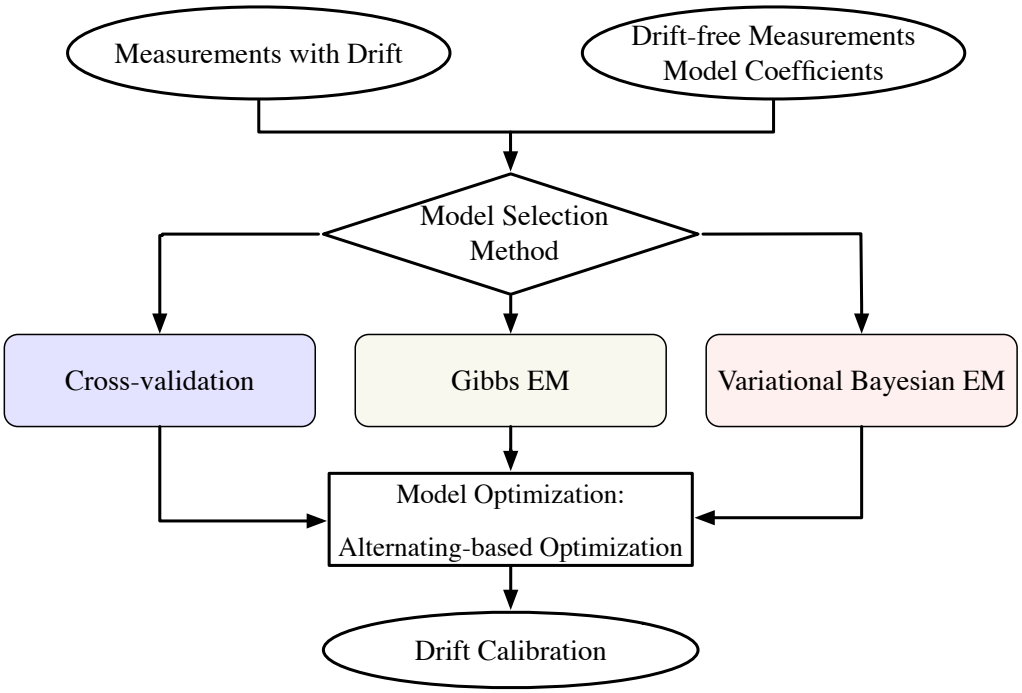


Figure 3.6 The proposed sensor drift calibration flow.

ature measurements with drifts as inputs, an alternating-based optimization algorithm is proposed to resolve sensor drift calibration formulation in model optimization. Additionally, Cross-validation, Gibbs EM and variational Bayesian EM are adopted to induce hyper-parameters, respectively. Based on the aforementioned techniques, the proposed flow is expected to accurately calibrate sensor drifts.

3.6 Experimental Results

The in-building temperature data are selected to test our proposed framework. We use several sensors for calibrate drifts. All data is directly generated from EnergyPlus as shown in Fig. 3.7.

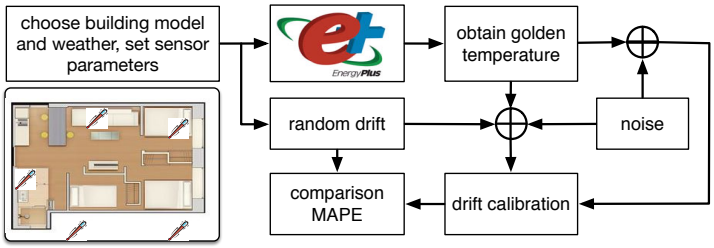
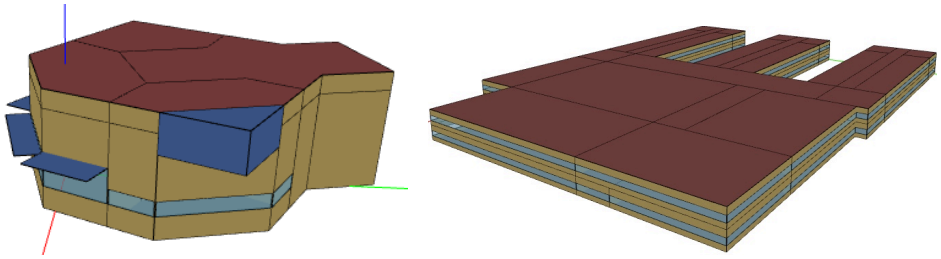


Figure 3.7 The generated simulation data.



(a) (b)

Figure 3.8 Benchmark: (a) Hall; (b) Secondary School.

As illustrated in Fig. 3.8, the two building benchmarks, Hall [13] with Washington, D.C weather and Secondary School [12] with Chicago weather, are simulated by EnergyPlus to generate the ground-truth in-building temperatures, which are used to evaluate our proposed framework. The temperature sampling period is set to be one hour.

Practically, both drift and measurement noises need to be carefully considered and reasonably set to get close to real temperature measurement. Because of a slow-aging effect, time effects on sensor performance is not considered in our experiments. Drift is set to be time-invariant while measurement noise is configured as time-variant. Two low-cost temperature sensors, MCP9509 with accuracy ±4.5 °C and LM335A with accuracy

± 5 °C are chosen to set drift variance, respectively. According to the triple standard deviation, we set two drift variances to be $\sigma^2 = (4.5/3)^2 = 2.25$ and $\sigma^2 = (5/3)^2 = 2.78$. In addition, according to our survey, the noise variance is set to be 0.001 °C. All temperature measurements are generated by adding noise.

The time-instant number needs to be reasonably set to mimic practical application and accurately calibrate sensor drifts. We assume the temperature measurements are drift-free during first $m_0 = 240$ time-instants (first 10 days). And during $m = 60$ time-instants (60 hours), the temperature measurements with drifts are utilized to test our proposed framework.

TSBL [125] and the proposed framework with Cross-validation, Gibbs EM and variational Bayesian EM (VB-EM) are exploited to calibrate sensor drifts, respectively. All methods are implemented by Python on 12-core Linux machine with 256G RAM and 2.80GHz. 100 combinations of hyper-parameters ratios and $s = 5$ folds are set in Cross-validation. In Gibbs EM, since the warm-start is waived in Gibbs sampling, to achieve better trade-off between accuracy and runtime, only $L = 10$ samples are generated to perform Monte Carlo approximation (3.17), and three hyper-parameters λ, δ_0, $\delta_{\boldsymbol{\epsilon}}$ are initialized with 10^3, 10^{-4} and 10^{-3}, respectively. The threshold values of convergence criterion are set as 10^{-8}, 10^{-2} and 10^{-3} in Algorithm 1, Algorithm 3 and Algorithm 4.

To verify the optimization and convergence of our proposed alternating-based algorithm as shown in Algorithm 1 more clearly,

two sensors are used to calibrate sensor drifts and sense school temperatures in the second benchmark. In Fig. 3.9, we can obviously see that, by using our proposed alternating-based algorithm, the cost of Formulation (3.7), estimated calibrations ϵ_1 and ϵ_2 converge quickly to the stationary points.

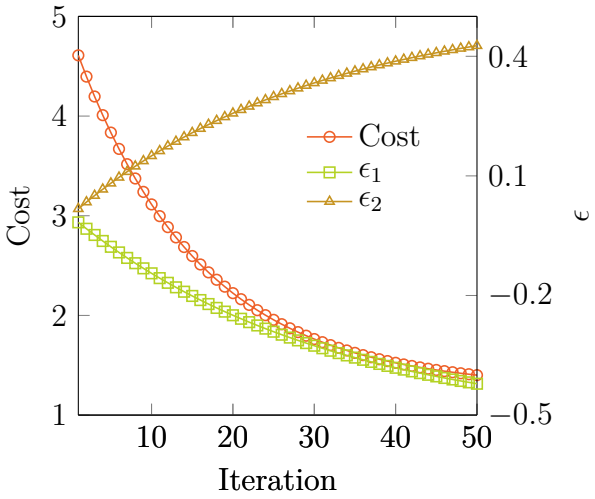


Figure 3.9 Convergence of Alternating-based Method.

The drift calibration accuracy is quantified by mean absolute percent error (MAPE) defined as follows:

$$\text{MAPE} = \frac{1}{nm} \sum_{k=1}^{m} \sum_{i=1}^{n} \left| \frac{\hat{\epsilon}_i^{(k)} - \epsilon_i}{\epsilon_i} \right|, \tag{3.45}$$

where $\hat{\epsilon}_i^{(k)}$ is estimated calibration. Specifically, in our proposed framework, $\hat{\epsilon}_i^{(k)} = \hat{\epsilon}_i$. The sensor drift calibration performances of accuracy and runtime are shown in Fig. 3.10 and Fig. 3.11.

As shown in Fig. 3.11, TSBL has acceptable computational overhead even if its computational complexity is dominated by multiple matrix inversion operations. However, as displayed in

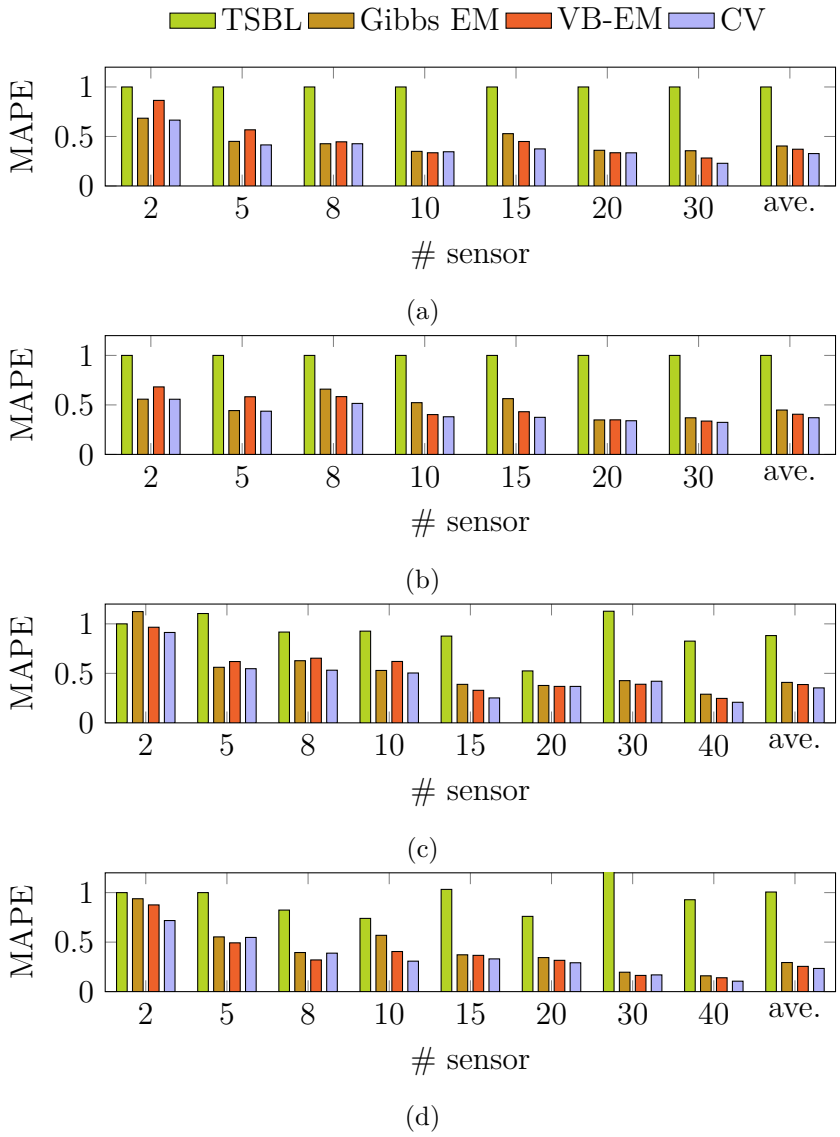


Figure 3.10 Drift variance is set to (a,c) 2.25; (b,d) 2.78; Benchmark: (a,b) Hall; (c,d) Secondary School.

Fig. 3.10, TSBL has the worst performance and robustness for drifts calibration. In fact, temperature signals lie in time-variant subspace since in-building temperatures are influenced by multiple time-variant factors, e.g., weather. As a result, TSBL cannot

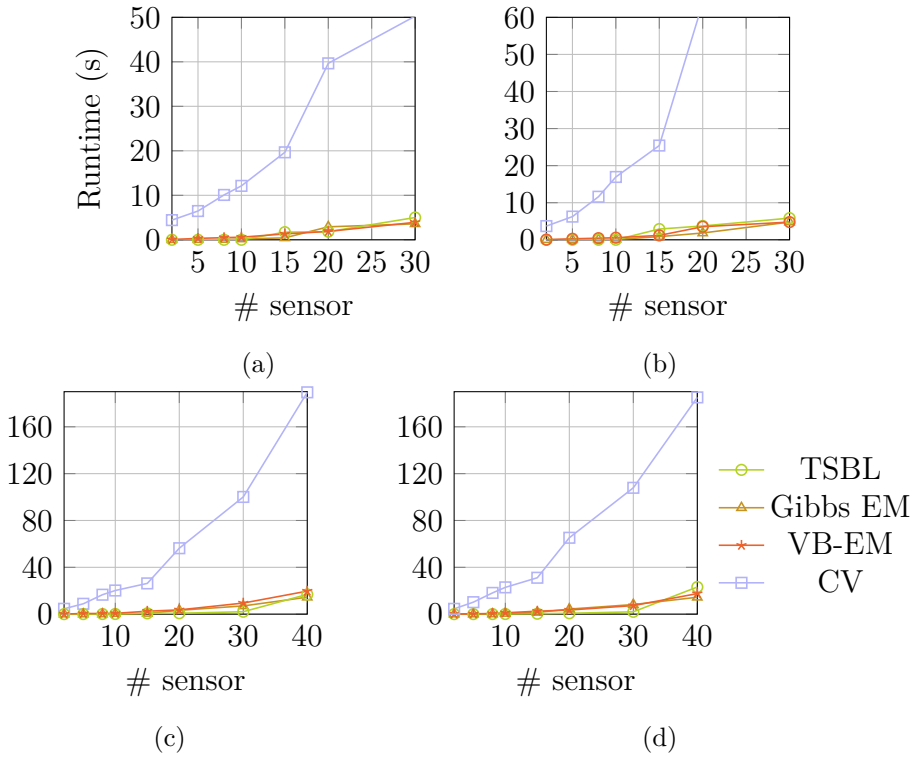


Figure 3.11 Runtime vs. # sensor: (a,c) 2.25; (b,d) 2.78; Benchmark: (a,b) Hall; (c,d) Secondary School.

achieve an obvious drift calibration.

Unlike TSBL, the proposed spatial correlation model can calibrate drifts even if temperature signals lie in time-variant subspace. Therefore, as shown in Fig. 3.10, the proposed framework with Cross-validation or Gibbs EM or VB-EM outperforms TSBL in terms of accuracy. On average, the proposed framework can achieve about 3× accuracy improvement. Moreover, the proposed drift calibration framework with Cross-validation can achieve the best accuracy. However, as shown in Fig. 3.11, Cross-validation has a heavy computational overhead since we

need to run Algorithm 1 for multiple times. Compared with Cross-validation and TSBL, Gibbs EM and VB-EM have lower computation complexity. Furthermore, in order to make a better trade-off between accuracy and runtime, Gibbs EM uses fewer samples to perform Monte Carlo approximation so that its accuracy is worse than VB-EM. However, as shown in Fig. 3.10, the proposed framework with Gibbs EM cannot achieve the best accuracy since Gibbs sampling is a stochastic method and VB-EM ignores the correlations between model coefficients $\mathbf{a}$ and calibration $\boldsymbol{\epsilon}$.

In Fig. 3.10, it can be seen that because of incremental correlation, the more sensors can achieve the higher accuracy of drift calibration by using our proposed framework. In practice, when fewer sensors need to be calibrated, in order to achieve better accuracy, Cross-validation can be used to determine hyper-parameters within a reasonable response time, e.g., 1 minute. While more sensors need to be calibrated, Gibbs EM can be exploited to determine hyper-parameters so that sensor measurement accuracy can be improved to a tolerable level within acceptable runtime. The proposed calibration framework with Gibbs EM can achieve a robust drift calibration and a better trade-off between accuracy and runtime.

In order to illustrate the fact that the warm-start can be waived and the initialization is reasonable for samples in Gibbs sampling, 100 Gibbs sampling traces, including the first 10 sampling traces, are shown in Fig. 3.12. We can observe that the

first 10 samples have the same shape with the 100 samples. It indicates that the first 10 samples have the same statistics with the 100 samples. Consequently, the first 10 samples can be used to represent the 100 samples.

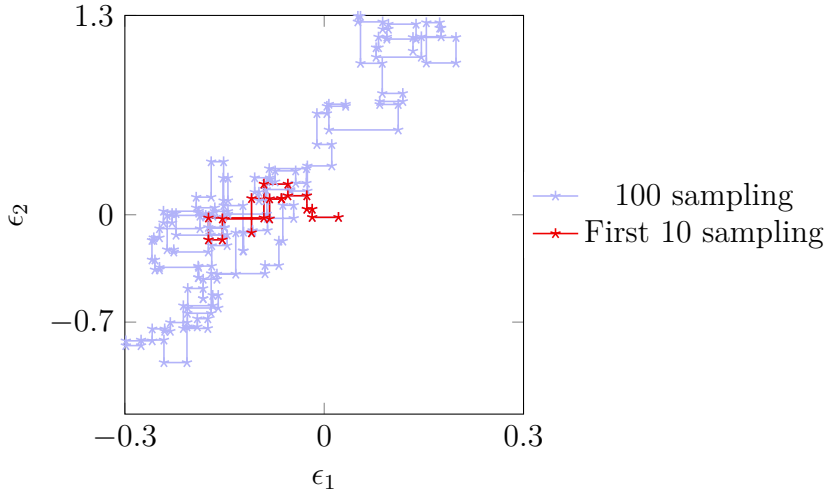


Figure 3.12 Gibbs sampling traces.

In this application, $L = 10$ can make a better trade-off between runtime and accuracy. More samples can improve the calibration accuracy. For illustrating the relationships among the number of samples, accuracy and runtime, we show the experimental results in Fig. 3.13. We set the sampling number to be 10, 100, 1000 and 10000. According to Fig. 3.10, because of incremental correlation discussed above, the drift calibration performance is more stable when the number of sensors is more than 15. Therefore, we average MAPE and runtime on different numbers of sensors when the number of sensors is more than 15. As the experimental results demonstrated in Fig. 3.13, the more

number of samples can improve the calibration accuracy while it has more runtime. In addition, the more number of samples cannot achieve significant accuracy improvement while it leads to a significant runtime cost. Therefore, $L = 10$ is enough to make a better trade-off between runtime and accuracy. Besides, based on the experimental results shown in Fig. 3.13, to achieve the same drift calibration accuracy with VB-EM, Gibbs EM needs 10000 samples, which will incur a 30× runtime overhead. Therefore, our proposed VB-EM can achieve better a trade-off between accuracy and runtime.

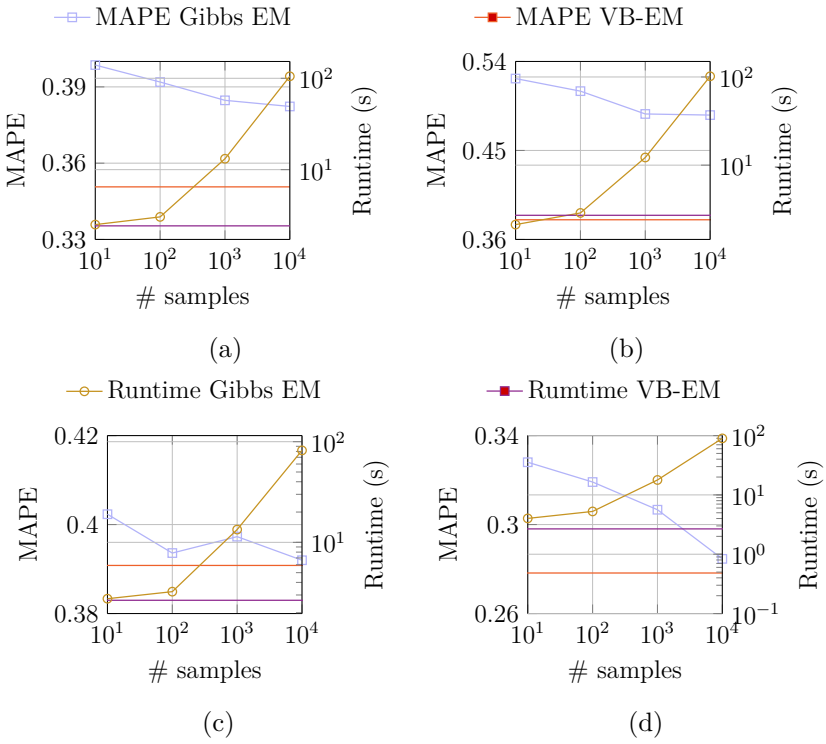


Figure 3.13 # samples vs. acc. vs. runtime: (a, b) hall; (c, d) school; (a, c) drift variance 2.25; (b, d) drift variance 2.78.

Furthermore, considering that the temperature sensors are

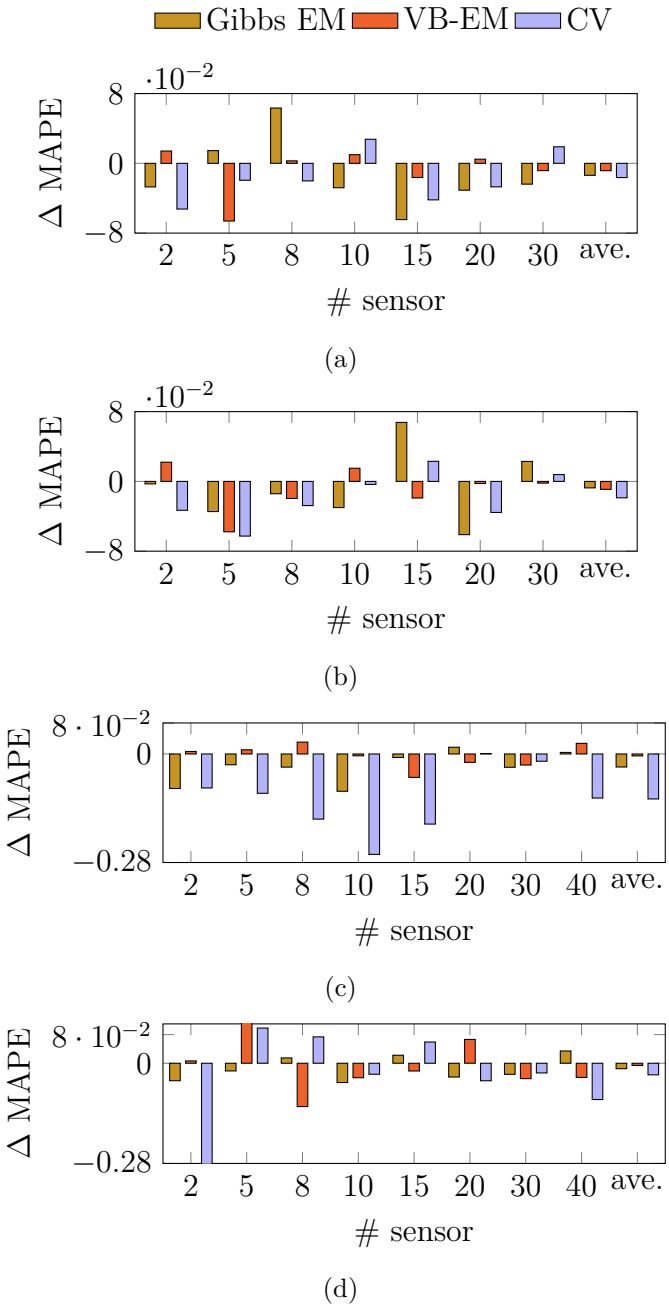


Figure 3.14 Δ MAPE when noise variance is set 0.01, drift variance (a,c) 2.25; (b,d) 2.78; Benchmark: (a,b) Hall; (c,d) Secondary School.

used in a severe environment, in order to verify the robustness for noise, we set another noise variance to be 0.01 °C. We de-

fine $\Delta MAPE = MAPE_{0.001} - MAPE_{0.01}$, where $MAPE_{0.001}$ is mean absolute percent error under 0.001 noise variance and $MAPE_{0.01}$ is mean absolute percent error under 0.01 noise variance. $\Delta MAPE > 0$ means that noise brings a positive effect on drift calibration accuracy while $\Delta MAPE < 0$ means that noise affects negatively on drift calibration accuracy. Experimental results are shown in Fig. 3.14. Although it is uncertain that which effect the noise will bring on drift calibration accuracy under different sensor numbers in each individual experiment, noise brings a negative effect on drift calibration accuracy on average, i.e., $\sum \Delta MAPE$. What's more, the larger noise variance triggers drift calibration degradation by using our proposed methods, i.e., CV, Gibbs EM, and VB-EM. Our experimental results as shown in Fig. 3.14 indicate that compared with CV and Gibbs EM, our proposed framework with VB-EM is robust for noise.

3.7 Summary

In this chapter, a sensor spatial correlation model has been proposed to perform drift calibration. Thanks to spatial correlation, the unknown actual temperature measured by each sensor is linearly expressed by all other sensors. The priors for model coefficients and drift calibration are applied to MAP estimation. MAP estimation is then formulated as a non-convex problem with three hyper-parameters, which is optimized by the pro-

posed alternating-based method. Cross-validation, Gibbs EM and VB-EM are exploited to determine hyper-parameters, respectively. Experimental results demonstrate that on benchmarks simulated from EnergyPlus, the proposed framework with variational Bayesian EM can achieve a robust drift calibration and a better trade-off between accuracy and runtime. Averagely, compared with state-of-the-art, the proposed framework can achieve about 3× accuracy improvement. In order to achieve the same drift calibration accuracy with variational Bayesian EM, Gibbs EM needs 10000 samples, which will result in a 30× runtime overhead.

□ **End of chapter.**

Chapter 4

Fast Aging Degradation Estimation

4.1 Preliminaries

4.1.1 Problem Formulation

Before committing the design to silicon, the aging-induced transistor degradation needs to be accurately estimated in the post-layout simulation to judge circuit reliability. However, the actual degree of the aging-induced transistor degradation is hard to estimate. On one hand, the traditional static aging reliability simulation causes inaccurate judgment on the aging-prone transistors since the dynamic stress conditions are completely ignored. On the other hand, the traditional dynamic simulation is time-consuming since it needs a large number of accepted transient steps. Besides, the accuracy of traditional dynamic aging reliability simulation heavily relies on dynamic stress conditions such as clock speeds and waveform swings. Thus the

traditional aging reliability simulation is hard to make a better trade-off between accuracy and runtime.

In this thesis, we adopt a data-driven approach in a supervised-learning manner to fast estimate aging-induced degradation. As mentioned above, the accuracy of traditional dynamic aging reliability simulation heavily relies on dynamic stress conditions. Thus our data-driven approach decouples dynamic stress conditions. In the training set, in order to achieve accurate estimations, the golden-truth aging-induced degradations are obtained by an industrial aging DFR tool with the static and dynamic stress conditions given by very sophisticated designers. Compared with static stress conditions, it is hard to determine dynamic stress conditions in practice, thus in the inference stage, dynamic stress conditions are not considered in our data-driven approach. This strategy can achieve a better compromise between computational complexity and model accuracy for the model implementation. In the industry, `dvtlin (HCI+BTI,10)` is used to assess the degree of aging-induced transistor degradation [11, 14]. `dvtlin(HCI+BTI,10)` is defined as follows:

Definition 1 (`dvtlin(HCI+BTI,10)`)**.** *The shifting value of threshold voltage of the transistor from fresh to 10 years due to HCI and BTI.*

The larger `dvtlin (HCI+BTI,10)` is, the worse aging-induced transistor degradation is, vice versa. For convenience, in this paper, we shorten `dvtlin(HCI+BTI,10)` as `dvtlin`.

Based on the above description, we define our problem formulation as follows.

Problem 2 (Estimating `dvtlin` in Analog ICs)**.** *Given some analog IC post-layout netlists, their stress conditions and a list containing all transistors with* ***dvtlin*** *obtained by an industrial DFR tool as the training set, our task is training a model on the training set to fast and accurately estimate* ***dvtlin*** *of each transistor on the testing set while minimizing the estimation error.*

4.1.2 Analog ICs Topology

When analog IC netlist files are parsed and flattened, analog ICs are naturally represented as bipartite graphs [88]. There are two disjoint sets in the bipartite graph. One set contains all nets and the other one contains all devices. Each undirected edge connects one net and one device. Fig. 4.1 demonstrates an analog circuit and the corresponding bipartite graph representation.

4.1.3 Graph Convolutional Networks

Since the graph can reveal structural information, graph learning provides a powerful data-driven approach for modeling irregular grid-based data in machine learning tasks [40, 60]. Node embedding generation is one of the most important technologies in graph learning [132]. The representation of each node is obtained by embedding and then fed into the traditional machine

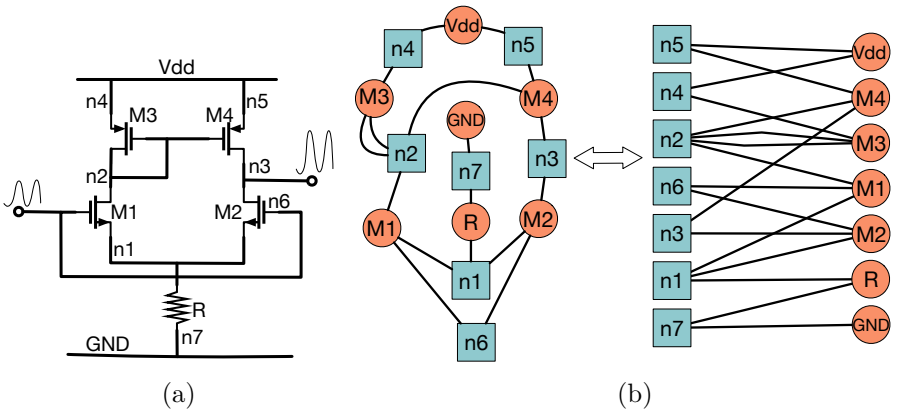


Figure 4.1 Analog circuit is represented as a bipartite graph: (a) An analog circuit; (b) The corresponding bipartite representation.

learning models for estimation and prediction. Recently, some data-driven methods were proposed to learn node embeddings automatically [40, 60].

One of the data-driven embedding methods is GCNs, whose main idea is automatically learning how to aggregate information iteratively from neighborhoods on graphs by using neural networks. The embedding generation is equivalent to using a filter on each node, whose accessible domain contains the node itself and its neighboring nodes. The neighboring nodes are a set of all nodes which are adjacent to the node itself. Compared with traditional CNNs, the graph topology is also treated as a feature to perform machine learning tasks so that GCNs can achieve higher accuracy on graph-based datasets and applications.

4.2 Heterogeneous Graph Representation

To adopt a GCN-based framework, it is fundamental to perform an embedding generation algorithm on graphs to encode the nodes and edges as feature vectors. Although the graph representations of analog ICs have been studied in many works of literature [88, 111, 162, 163], the embedding generation algorithm cannot be directly performed on the graph representations of analog ICs since not all nodes have design parameters as features. Next, we will propose a directed heterogeneous graph representation method to guarantee that each node has features.

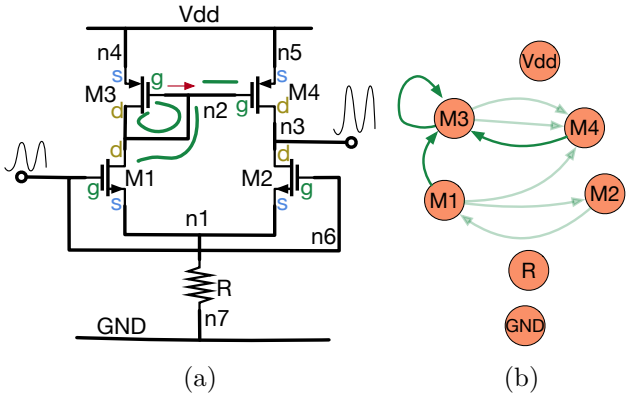
To perform an embedding generation algorithm, we treat each device, direct current (DC) voltage source or ground as a node. An analog IC post-layout netlist can be flattened as a bipartite graph as illustrated in Fig. 4.1. To guarantee that each node has features, a naïve method is constructing a homogeneous multigraph to represent the topology of analog IC netlist [71]. In this homogeneous multigraph, each edge represents one path from one device to another. It allows multiple edges between any two nodes since there may be multiple paths from one device to the other one. Note that dynamic stress conditions do not be considered in this graph representation so that the GCN-based framework is independent of them and can replace the traditional static aging reliability simulation.

The typical analog IC netlists have heterogeneity since they contain multi-typed basic devices and multi-typed connection

pins [98]. An inevitable problem of the homogeneous representation method is that it fails to characterize the diversities among pins, devices, connections, and relative sequential relationships. In order to express these diversities, we propose a heterogeneous directed multigraph representation, where the type of the edges is treated as the type of the pins to which it connects. The heterogeneous directed multigraph is defined as follows.

Definition 2 (Heterogeneous directed multigraph)**.** *A heterogeneous directed multigraph is defined as a graph* $\mathcal{HMG}_d(\mathcal{V}_{hmg}, \mathcal{E}_{hmg}, \mathcal{O}_{\mathcal{V}_{hmg}}, \mathcal{R}_{\mathcal{E}_{hmg}}, \varphi_{hmg})$*, where* $\mathcal{V}_{hmg}$ *is the set of nodes,* $\mathcal{E}_{hmg}$ *is the multiset of edges.* $\mathcal{O}_{\mathcal{V}_{hmg}}$ *and* $\mathcal{R}_{\mathcal{E}_{hmg}}$ *represent the sets of node types and edge types, respectively.* φ_{hmg} *is adopted to assign each node in* $\mathcal{V}_{hmg}$ *to a node type, i.e.,* $\varphi_{hmg}(\upsilon_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$ *for* $\forall \upsilon_i \in \mathcal{V}_{hmg}$*.* r *indicates the edge type, such that* $r \in \mathcal{R}_{\mathcal{E}_{hmg}}$*. Each instance in* $\mathcal{E}_{hmg}$ *is* $((\upsilon_i, \upsilon_j), r)$ *and the ordered pair* (υ_i, υ_j) *satisfies* $\upsilon_i, \upsilon_j \in \mathcal{V}_{hmg}$*.*

We use an example to show this heterogeneous directed multigraph representation. As shown in Fig. 4.2(a), if we stand at the gate of the transistor `M3` and lookout, we will see transistors `M1`, `M3` and `M4`. Thus there are three directed edges (denoted by dark green) with gate connections from `M1`, `M3` and `M4` to the transistor `M3` as shown in Fig. 4.2(b). This method is inspired by circuit analysis, where the input impedance is obtained in the same fashion [98]. In the same manner, we can obtain the directed multigraph corresponding to the gate connections as shown in



(a) (b)

Figure 4.2 The heterogeneous directed multigraph representation: (a) The differential amplifier; (b) The gate connection.

Fig. 4.2(b), where all green edges denote the gate connection. Moreover, the circuit netlist in Fig. 4.2(a) is finally transformed into the multigraph in Fig. 4.3. In particular, considering that the electrical characteristics of direct current voltage sources and grounds are not influenced by other devices, we set them to be predecessors.

After the topology of different connections is obtained, we use one adjacency matrix to encode the topology of each connection in the heterogeneous multigraph. The adjacency matrix of the type-r connection is defined as $\boldsymbol{A}_r$, where each element $A_r(i,j)$ is the number of the instance $((v_i, v_j), r)$ in the multiset $\mathcal{E}_{dmg}$ and $r \in \mathcal{R}_{\mathcal{E}_{hmg}}$. As an example, the adjacency matrix $\boldsymbol{A}_g$ corresponding to the gate connection in Fig. 4.2(b) is shown in Matrix (4.1).

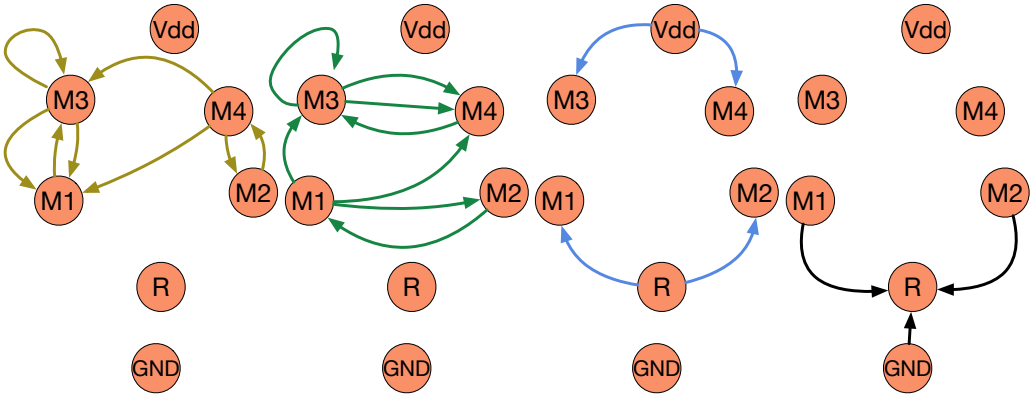


Figure 4.3 A heterogeneous directed multigraph with multi-typed edges. From left to right: drain, gate, source and other connections.

$$
\begin{array}{c}
 \\
M1 \\
M2 \\
M3 \\
M4 \\
R \\
Vdd \\
GND
\end{array}
\begin{array}{c}
\begin{array}{ccccccc}
M1 & M2 & M3 & M4 & R & Vdd & GND
\end{array} \\
\left(\begin{array}{ccccccc}
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)
\end{array}. \quad (4.1)
$$

In summary, an analog IC netlist can be represented as a heterogeneous directed multigraph. The topology of heterogeneous directed multigraph is encoded as multiple adjacency matrices, where each one encodes one type of connection. In the next section, we will propose an H-GCN model on the heterogeneous directed multigraph to estimate `dvtlin` by taking multiple adjacency matrices and the design parameters of all typed devices

as inputs.

4.3 Heterogeneous GCN

In this section, an H-GCN model on the heterogeneous directed multigraph is developed to estimate `dvtlin`. The features of all types of devices are firstly mapped into a unified latent space then our proposed H-GCN model sequentially conducts multiple embedding generations on these features.

4.3.1 Notations

As discussed in Section 4.2, given an analog IC netlist, we parse and flatten it to be a bipartite graph as shown in Fig. 4.1. Then we transform to be a heterogeneous directed multigraph $\mathcal{HMG}_d(\mathcal{V}_{hmg}, \mathcal{E}_{hmg}, \mathcal{O}_{\mathcal{V}_{hmg}}, \mathcal{R}_{\mathcal{E}_{hmg}}, \varphi_{hmg})$ as shown in Fig. 4.2 and Fig. 4.3. Then according to the heterogeneous directed multigraph, we can obtain $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ adjacency matrices, *i.e.*, $\boldsymbol{A}_r$ for $\forall r \in \mathcal{R}_{\mathcal{E}_{hmg}}$, where $|\cdot|$ is the cardinality of set. Besides, the device $v_i \in \mathcal{V}_{hmg}$ has $h_{\varphi_{hmg}(v_i)}$ design parameters for $\varphi_{hmg}(v_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$. We use these design parameters as the features of each node, *i.e.*, $\boldsymbol{x}_{v_i} \in \mathbb{R}^{1 \times h_{\varphi_{hmg}(v_i)}}$. Note that the number of design parameters relies on the type of device. Next, we will propose a unified latent space mapping method to map the features of each node into a unified latent space so that the embedding generation can be performed meanwhile the features of multi-typed devices can be well effectively exploited and encoded.

4.3.2 Unified Latent Space Mapping

A typical analog IC netlist contains multi-typed devices, which have different design parameters. We regard each device as a node and use its design parameters as the node features vector. To perform an embedding generation algorithm and aggregate information from the node itself and its multi-typed neighboring nodes, a straightforward method is concatenating all of these design parameters as a long feature vector with one-hot encoding [71]. However, it will miss some structural information among multi-typed nodes as well as unstructured content associated with each node [20]. In this work, a latent space mapping method is used to transform the features vectors of all types of nodes into a unified latent space.

We propose to use $|\mathcal{O}_{\mathcal{V}_{hmg}}|$ node-type-related mapping matrices to map the original feature vectors. For a node $v_i \in \mathcal{V}_{hmg}$ whose node type is $t = \varphi_{hmg}(v_i) \in \mathcal{O}_{\mathcal{V}_{hmg}}$ and feature vector is $\boldsymbol{x}_{v_i} \in \mathbb{R}^{1\times h_t}$, we define a node-typed-related matrix $\boldsymbol{U}_t \in \mathbb{R}^{h_t\times\tau}$ to map the feature vector with length h_t into a unified τ-dimension latent space, i.e., $\boldsymbol{f}_{v_i}^{(0)} = \boldsymbol{x}_{v_i} \cdot \boldsymbol{U}_t \in \mathbb{R}^{1\times\tau}$. In order to ease the model training on GPUs, we extend it to be a matrix-matrix multiplication as follows:

$$\boldsymbol{F}^{(0)} = \sum_{t\in\mathcal{O}_{\mathcal{V}_{hmg}}} \boldsymbol{X}_t \cdot \boldsymbol{U}_t \in \mathbb{R}^{|\mathcal{V}_{hmg}|\times\tau}, \tag{4.2}$$

where $\boldsymbol{X}_t \in \mathbb{R}^{|\mathcal{V}_{hmg}|\times h_t}$ is a feature matrix stacking the feature vectors of all type-t nodes. In particular, a node feature vector is

$\mathbf{0} \in \mathbb{R}^{1\times h_t}$ if the type of the node is not t. $\boldsymbol{F}^{(0)}$ is the feature matrix, where the feature vectors of all nodes are in a unified latent space. We take the circuit in Fig. 4.2(a) as an example. There are three types of nodes (devices), i.e., $\mathcal{O}_{\mathcal{V}_{hmg}} = \{vs, trans, res\}$. Thus there are three feature matrices:

$$\boldsymbol{X}_{vs} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{x}_{Vdd} \\ \boldsymbol{x}_{GND} \end{bmatrix}, \boldsymbol{X}_{trans} = \begin{bmatrix} \boldsymbol{x}_{M1} \\ \boldsymbol{x}_{M2} \\ \boldsymbol{x}_{M3} \\ \boldsymbol{x}_{M4} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \boldsymbol{X}_{res} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{x}_{R} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}. \tag{4.3}$$

It is noted that the concatenated features representation [71] is a special case of latent space representation. Compared with the concatenated features representation, our latent space representation is general enough and can effectively exploit and encode features. Furthermore, the proposed latent space mapping method can be extended to multiple layers in the same fashion with multilayer perceptron (MLP) to extract high-order features.

4.3.3 Embedding Generation

After applying the proposed latent space mapping method, the features of all nodes share the same representation forms. An H-GCN model is proposed as the skeleton of the estimation model.

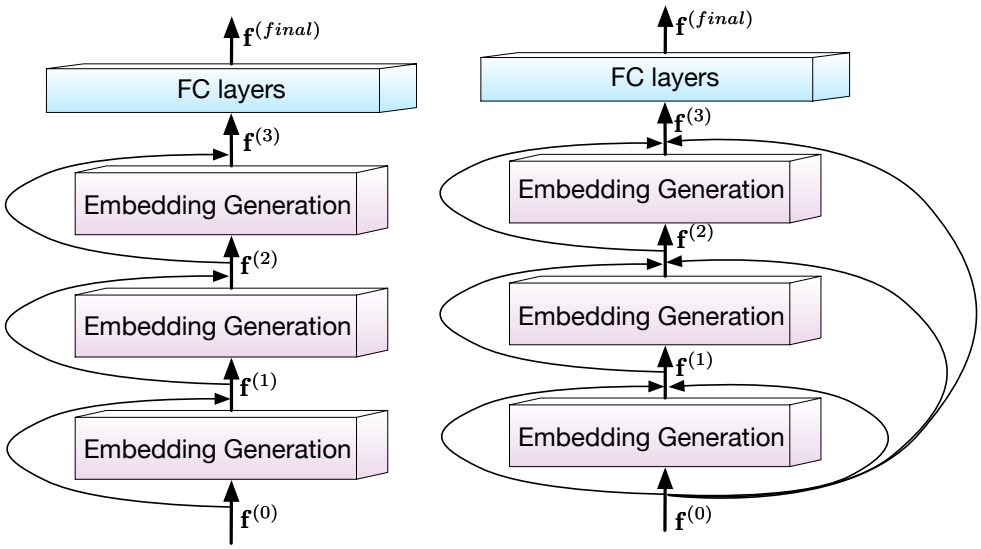


Figure 4.4 H-GCN and deep H-GCN: (a) H-GCN; (b) deep H-GCN with initial residual connections.

The model takes the unified latent feature representations of all nodes $\boldsymbol{F}^{(0)}$ and $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ adjacency matrices $\boldsymbol{A}_r$ $(r \in \mathcal{R}_{\mathcal{E}_{hmg}})$ as inputs.

As discussed in Section 4.2, we use one adjacency matrix to represent one type of connection in the heterogeneous directed multigraph. In order to distinguish among different connection pins, inspired by [86], we assign a learnable model coefficient to each adjacency matrix. Then the overall topology of the heterogeneous directed multigraph can be expressed as the summation of the adjacency matrices of all types of connections with these model coefficients. For example, as shown in Fig. 4.3, there are four adjacency matrices $\boldsymbol{A}_g$, $\boldsymbol{A}_s$, $\boldsymbol{A}_d$ and $\boldsymbol{A}_{otr}$ in the heterogeneous directed multigraph. We assign four model coefficients

w_g, w_s, w_d and w_{otr} to distinguish them. To improve numerical stability, we normalize all adjacency matrices. Therefore, the topology of the heterogeneous directed multigraph can be encoded as $w_g\tilde{\boldsymbol{A}}_g + w_s\tilde{\boldsymbol{A}}_s + w_d\tilde{\boldsymbol{A}}_d + w_{otr}\tilde{\boldsymbol{A}}_{otr}$, where the normalized adjacency matrix $\tilde{\boldsymbol{A}}_r = \boldsymbol{A}_r\boldsymbol{D}_r^{-1}$, $\boldsymbol{D}_r$ is a diagonal matrix with $D_r(i,i) = \sum_{j=1}^{|\mathcal{V}_{hmg}|} A_r(i,j)$ and $r \in \mathcal{R}_{\mathcal{E}_{hmg}} = \{g, s, d, otr\}$. For the sake of convenience, we denote the heterogeneous adjacency matrix as follows:

$$\tilde{\boldsymbol{A}} \triangleq \sum_{r \in \mathcal{R}_{\mathcal{E}_{hmg}}} w_r \tilde{\boldsymbol{A}}_r, \tag{4.4}$$

where w_r is a model coefficient. $\tilde{\boldsymbol{A}}_r$ is the normalized adjacency matrix. r denotes the connection type.

Typically, there are two popular GCN models. One is vanilla GCN [60], where the information of the node itself and that of its neighboring nodes are aggregated by adding one identity matrix to the normalized graph adjacency matrix. The other is GraphSAGE [40], where the feature of the node itself is concatenated with that of its neighboring nodes. Since there are self-loops in our graph representation as shown in Figs. 4.2 and 4.3. Compared with the vanilla GCN, GraphSAGE has the stronger ability to distinguish between the self-loop edge and the node itself. Thus we adopt GraphSAGE as our basic backbone. In our proposed H-GCN, the developed heterogeneous embedding generation is shown as follows:

$$\boldsymbol{F}^{(l)} = \sigma\left(\text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}\right) \cdot \boldsymbol{W}^{(l)}\right), \tag{4.5}$$

where CONCAT$(\cdot)$ denotes the concatenation operation. $\sigma(\cdot)$ is a nonlinear activation function. In this work, we use ReLU as our nonlinear activation function. $\boldsymbol{W}^{(l)}$ denotes the learnable model coefficients in the l-th embedding generation layer.

The proposed H-GCN is shown in Fig. 4.4(a), where the embedding generation as shown in Equation (4.5) is recursively and sequentially performed several times, the local topology and node features are extracted to represent the feature vector of each node. Then the extracted feature vector of each node is fed into a traditional machine learning model to estimate `dvtlin`.

4.4 Going to Deep H-GCN

Despite GCN can achieve enormous successes in graph learning task, it is very shallow. Such a shallow model limits its ability to extract information from multi-hop devices so that it cannot achieve a more accurate estimation for complex and large-scale circuit netlists. Besides, stacking more layers and adding nonlinearity tend to degrade the performance of this model, due to the over-smoothing issue [70]. In this section, based on GCNII [22], we further extend the proposed H-GCN to be a deep version via initial residual connections and identity mappings to alleviate the over-smoothing issue.

4.4.1 Embedding Generation with Initial Residual Connections and Identity Mappings

The primary reason behind the over-smoothing issue is that the representations of the nodes as shown in Equation (4.5) become indistinguishable as the number of heterogeneous embedding generation layers increases. In order to alleviate the over-smoothing issue, according to GCNII [22], the final representation of each node retains the input layer even if many layers are stacked. According to this manner, we extend the heterogeneous embedding generation as shown in Equation (4.5) by concatenating with the input of the first layer as follows:

$$\boldsymbol{F}^{(l)} = \sigma\left(\text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}\right) \cdot \boldsymbol{W}^{(l)}\right), \quad (4.6)$$

where $\boldsymbol{F}^{(0)}$ is the unified feature matrix defined in Equation (4.2). The heterogeneous adjacency matrix $\tilde{\boldsymbol{A}}$ is defined in Equation (4.4). As shown in Fig. 4.4(b), our extended deep H-GCN leverages the embedding generation with the initial residual connections. Thus the final representation of each node retains the input layer even if many layers are stacked.

According to [22, 61], performing multiple non-linearity operations to the feature matrix still causes the over-smoothing issue. In our deep H-GCN model, inspired by ResNet [43], to further alleviate the over-smoothing issue, we leverage an identity matrix $\boldsymbol{I}$ with a hyper-parameter β_l to the model coefficient

matrix $\boldsymbol{W}^{(l)}$ in Equation (4.6) as follows:

$$\boldsymbol{W}^{(l)} \leftarrow (1-\beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)}. \tag{4.7}$$

Note that $\boldsymbol{I}$ is easy to extend as an augmented matrix of an identity matrix with a zero matrix if $\boldsymbol{W}^{(l)}$ is not a square matrix. By adding the matrix $\boldsymbol{I}$, we have two advantages: Similar to the motivation of ResNet [43], the matrix $\boldsymbol{I}$ ensures that a deep H-GCN model achieves the same performance as its shallow version does; Since the optimal model coefficient matrix $\boldsymbol{W}^{(l)}$ has small norm, it allows to put strong regularization. Chen et. al suggest $\beta_l = \log(1/l + 1)$ [22].

Like our proposed H-GCN, our extended deep H-GCN recursively and sequentially performs the embedding generation with initial residual connections and identity mappings several times to extract the information of local topology and node features to represent the feature vector of each node. Then the extracted feature vector is fed into a traditional machine learning model to estimate `dvtlin` of each transistor.

Unlikes JKNet [137], our extended deep H-GCN can facilitate several layers stacking since the output of each layer does not be uniformly combined to the last layer. Compared with GCNII [22], our extended deep H-GCN has the stronger ability to distinguish between the self-loop edge and the node itself since it concatenates the features of neighboring nodes and the node itself. By using the extended deep H-GCN, it is expected to achieve more accurate estimations of aging-induced transistor

degradation.

Algorithm 5 Embedding Generation of Deep H-GCN

Require: $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ normalized adjacency matrices $\tilde{\boldsymbol{A}}_r$ and connection-type-related model coefficients w_r with $\forall r \in \mathcal{R}_{\mathcal{E}_{hmg}}$; feature matrices of all nodes corresponding to each type of nodes $\boldsymbol{X}_t$ and node-type-related latent space mapping matrices $\boldsymbol{U}_t$ with $\forall t \in \mathcal{O}_{\mathcal{V}_{hmg}}$; Search depth D; Model coefficients matrices $\boldsymbol{W}^{(l)}$ for $l = 1, 2, \cdots, D$.

1: $\boldsymbol{F}^{(0)} \leftarrow \sum_{t \in \mathcal{O}_{\mathcal{V}_{hmg}}} \boldsymbol{X}_t \boldsymbol{U}_t$; ▷ map to a unified latent space

2: $\tilde{\boldsymbol{A}} \leftarrow \sum_{r \in \mathcal{R}_{\mathcal{E}_{hmg}}} w_r \tilde{\boldsymbol{A}}_r$; ▷ encode the topology of the heterogeneous graph

3: **for** $l = 1$ to D **do**

4: $\quad \beta_l \leftarrow \log(1/l + 1)$;

5: $\quad \boldsymbol{F}^{(l)} \leftarrow \sigma(\text{CONCAT}(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}) \cdot (1 - \beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)})$; ▷ embedding generation

6: **end for**

7: **return** Embedding feature matrix $\boldsymbol{F}^{(D)}$.

Our proposed deep embedding generation algorithm (i.e., forward propagation) is summarized in Algorithm 5. The connection-type-related normalized adjacency matrices $\tilde{\boldsymbol{A}}_r$, connection-type-related model coefficients w_r, feature matrices of all nodes $\boldsymbol{X}_t$, node-type-related latent space mapping matrices $\boldsymbol{U}_t$, the search depth D and model coefficients matrices $\boldsymbol{W}^{(l)}$ are provided as inputs. According to Equation (4.2), the latent space mapping matrices $\boldsymbol{U}_t$ are used to map features of all nodes $\boldsymbol{X}_t$ into a unified feature representation $\boldsymbol{F}^{(0)}$. Then the heterogeneous adjacency matrix $\tilde{\boldsymbol{A}}$ is obtained by Equation (4.4). Let l denotes the current step in the loop (the depth of the search) and $\boldsymbol{F}^{(l)}$ denotes the feature representation of all nodes at this step. Then

each step in the loop of Algorithm 5 proceeds as follows: Firstly, the hyper-parameter β_l can determined by $\log(1/l+1)$. Meanwhile, all nodes aggregate the feature representations of their neighboring nodes into a matrix $\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}$. Then the aggregated neighboring feature matrix $\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}$ is concatenated with the node's current representation $\boldsymbol{F}^{(l)}$ and the initial unified feature matrix $\boldsymbol{F}^{(0)}$. And this concatenated feature matrix is fed into a fully connected (FC) layer with a nonlinear activation function $\sigma(\cdot)$ and model coefficients $(1-\beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)}$, which makes the model more robust. The outputs are used at the next step of the algorithm. Finally, the proposed deep embedding generation algorithm takes the embedding feature matrix $\boldsymbol{F}^{(D)}$ as an output.

Note that only transistors are prone to have aging degradations in analog IC netlists while features of all devices need to be fed into our proposed deep H-GCN to perform the embedding generation algorithm. Thus a mask matrix with the size $\#transistors \times \#devices$ is used to extract the embedding features of each transistor in the last embedding generation layer. Finally, several traditional FC layers are adopted to estimate `dvtlin` of each transistor by inputting the embedding features. The Mean Square Error (MSE) function is used as the loss function to compute the errors between the ground-truth and our estimated `dvtlin`. All model coefficients, including w_r, $\boldsymbol{U}_t$ and $\boldsymbol{W}^{(l)}$ in Algorithm 5, are updated by the backpropagation and the gradient-based method in an end-to-end fashion in each

training epoch.

4.4.2 Over-smoothing Analysis

Here we provide an over-smoothing analysis for the proposed H-GCN and the extended deep H-GCN. For simplicity, we assume connection-type-related model coefficients w_r for $\forall r \in \mathcal{R}_{\mathcal{E}_{hmg}}$ and the unified feature matrix $\boldsymbol{F}^{(0)}$ to be non-negative, that is $w_r \geq 0$ and $\boldsymbol{F}^{(0)} \geq \boldsymbol{O}$. Note that we can convert w_r and $\boldsymbol{F}^{(0)}$ to be nonnegative by a linear transformation if they are negative [22].

Over-smoothing analysis of multi-layer H-GCN

To illustrate that the over-smoothing issue in the proposed H-GCN, we will show that the directed multigraph as shown in Section 4.2 is strongly connected and aperiodic, and the embedding generation as shown in Equation (4.5) is equivalently expressed as the formulation of Laplacian smoothing.

To show strong connectivity and aperiodicity for the directed multigraph, we first see the bipartite graph representation of analog IC as shown in Fig. 4.1. The bipartite graph is connectivity since there is a path between every pair of nodes [29]. Then it is easily proved in contrapositive form that the directed multigraph is connectivity if it is transferred from a connected bipartite. Moreover, the directed multigraph is strongly connected since any two devices are paired up as a connection in each net to transfer the connected bipartite as the directed multi-

graph. Besides, the directed multigraph is aperiodic since there is no integer $\kappa > 1$ that divides the length of every cycle of the graph [55].

Here we will analyze whether the embedding generation as shown in Equation (4.5) satisfies the formulation of Laplacian smoothing. Since $\boldsymbol{F}^{(0)}$ is non-negative, like [22], we remove the nonlinear activation function (We use ReLU as the nonlinear activation function.). Thus Equation (4.5) can be transferred as follows:

$$\boldsymbol{F}^{(l)} = \text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}\right) \cdot \boldsymbol{W}^{(l)}. \tag{4.8}$$

We set $\boldsymbol{W}^{(l)} = [\lambda \boldsymbol{I}, (1-\lambda)\boldsymbol{I}]^\top \cdot \hat{\boldsymbol{W}}^{(l)}$. Then Equation (4.8) can be transferred as follows:

$$\begin{aligned} &\boldsymbol{F}^{(l)} \\ =&\text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}\right) \cdot [\lambda \boldsymbol{I}, (1-\lambda)\boldsymbol{I}]^\top \cdot \hat{\boldsymbol{W}}^{(l)} \\ =&\left((\boldsymbol{I} - \lambda \boldsymbol{I}) \cdot \boldsymbol{F}^{(l-1)} + \lambda \tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}\right) \cdot \hat{\boldsymbol{W}}^{(l)}, \\ =&(\boldsymbol{I} - \lambda \boldsymbol{D}^{-1}\boldsymbol{L})\boldsymbol{F}^{(l-1)} \cdot \hat{\boldsymbol{W}}^{(l)}, \end{aligned} \tag{4.9}$$

where $0 < \lambda \leq 1$ is a parameter which controls the weighting between the features of the current node and its neighbors. $\boldsymbol{L} = \boldsymbol{D} - \tilde{\boldsymbol{A}}$ is the graph Laplacian matrix, where $\boldsymbol{D}$ is a diagonal matrix with $D(i,i) = \sum_j \tilde{A}(i,j)$. According to Equation (4.9), the embedding generation $\boldsymbol{F}^{(l)}$ as shown in Equation (4.5) is equivalently expressed as the formulation of Laplacian smoothing, i.e., $(\boldsymbol{I} - \lambda \boldsymbol{D}^{-1}\boldsymbol{L})\boldsymbol{F}^{(l-1)} \cdot \hat{\boldsymbol{W}}^{(l)}$ [70].

According to the fundamental Theorem of Markov Chains

[97], since the proposed directed multigraph is strongly connected and aperiodic, and the embedding generation can be equivalently expressed as the formulation of Laplacian smoothing, the random walk converges to a unique stationary distribution $\boldsymbol{\pi}$, i.e., $\lim_{D\to\infty} \boldsymbol{f}_i^{(D)} = \boldsymbol{\pi}$, where the i-th row in $\boldsymbol{F}^{(D)}$ is represented as $\boldsymbol{f}_i^{(D)}$, which is also the embedding feature of the i-th node. Thus as the number of layers increases, the representations of the nodes in H-GCN are inclined to converge to a certain value and become indistinguishable, which is the over-smoothing issue.

Over-smoothing analysis of multi-layer deep H-GCN

Here we give an over-smoothing analysis for the extended deep H-GCN to illustrate it can preventing the over-smoothing even if the number of layers goes to infinity.

We consider the embedding generation with initial residual connections and identity mappings as shown in Equations (4.6) and (4.7) in the extended deep H-GCN. Since $\boldsymbol{F}^{(0)}$ is non-negative, we remove the nonlinear operation. Moreover, we fix $(1-\beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)}$ to be $\gamma_l \boldsymbol{W}^{(l)}$, where γ_l is a parameter. Thus Equations (4.6) and (4.7) can be transferred as follows:

$$\boldsymbol{F}^{(l)} = \text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}\right) \gamma_l \boldsymbol{W}^{(l)}. \tag{4.10}$$

In the same manner, we set $\boldsymbol{W}^{(l)} = [\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I}]^\top \cdot \hat{\boldsymbol{W}}^{(l)}$. Then

Equation (4.10) can be transferred as follows:

$$\begin{aligned}
&\boldsymbol{F}^{(l)} \\
&= \text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}\right) \cdot [\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I}]^{\top} \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)} \\
&= \left((\tilde{\boldsymbol{A}} + \boldsymbol{I})\boldsymbol{F}^{(l-1)} + \boldsymbol{F}^{(0)}\right) \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)}.
\end{aligned} \tag{4.11}$$

Furthermore, we set $\boldsymbol{F}^{(0)} = (\tilde{\boldsymbol{A}} + \boldsymbol{I}) \cdot \hat{\boldsymbol{x}}$. Thus Equation (4.11) can be further transferred as follows:

$$\begin{aligned}
\boldsymbol{F}^{(l)} &= \left((\tilde{\boldsymbol{A}} + \boldsymbol{I})(\boldsymbol{F}^{(l-1)} + \hat{\boldsymbol{x}})\right) \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)} \\
&= \left((\boldsymbol{I} - \boldsymbol{L})(\boldsymbol{F}^{(l-1)} + \hat{\boldsymbol{x}})\right) \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)}.
\end{aligned} \tag{4.12}$$

For simplicity, like [22], we further set $\hat{\boldsymbol{W}}^{(l)}$ as an identity matrix. Consequently, according to Equation (4.12) and [22], we can express the final embedding feature representation as follows:

$$\boldsymbol{F}^{(D)} = \left(\sum_{l=0}^{D} \left(\prod_{k=D-l}^{D} \gamma_k\right) (\boldsymbol{I} - \boldsymbol{L})^l\right) \cdot \hat{\boldsymbol{x}}. \tag{4.13}$$

This expression suggests that deep H-GCN converges to a distribution that carries information from both the input feature $\hat{\boldsymbol{x}}$ ($\boldsymbol{F}^{(0)}$) and the graph structure $\boldsymbol{L}$ ($\tilde{\boldsymbol{A}}$). This property ensures that deep H-GCN will not suffer from an over-smoothing issue even if the number of layers goes to infinity.

4.5 Large Scale Graph Training via Neighborhood Sampling

With the fast development of semiconductors, more and more devices and connections are integrated into an analog IC, which brings great challenges to the model training because of the iterative embedding generations among neighboring nodes. For example, to compute the gradients of node v_i in the l-th layer, the embedded features of the neighboring nodes $\mathcal{N}(v_i)$ of v_i are required, i.e., some features in the $(l-1)$-th layer. Moreover, the features of $\mathcal{N}(\mathcal{N}(v_i))$ are also needed, i.e., some features in the $(l-2)$-th layer. As the search depth D (the number of embedding generation layers) increases, the kind of neighborhood explosion will bring the much more node number in each training epoch so that it causes lots of training time and memory resources even through out-of-memory on GPU. Fig. 4.5(a) is taken as an example to show the neighborhood explosion.

To achieve enough scalability, a straightforward method is using the traditional graph partitioning algorithms [17] to partition a large-scale graph to be several isolated small size subgraphs to be trained. However, the features of the neighboring node (device) cannot be aggregated to the node itself in all training epochs if the neighboring node and the node itself are partitioned into two isolated subgraphs. As a result, these graph partitioning algorithms cause performance degradation.

To alleviate the performance degradation, like the dropout

technique in CNNs [109], a probability-based sampling method is leveraged to obtain different subgraphs among different training epochs. Considering that the runtime is dominated by graph transformation in our proposed flow, we perform a probability-based sampling method on the bipartite graph. In our proposed probability-based sampling method, the sampling probability relies on the number of edges (degree) in each node, as shown in Equation (4.14).

$$\mathcal{P}(v_i) = \frac{D_b(i,i)}{|\boldsymbol{D}_b|}, \tag{4.14}$$

where $\boldsymbol{D}_b$ is a diagonal matrix. Its each diagonal element $D_b(i,i)$ $= \sum_j A_b(i,j)$, where $\boldsymbol{A}_b$ is the adjacency matrix of the bipartite graph, which is constructed by directly parsing and flattening the circuit netlist as shown in Fig. 4.1. $|\boldsymbol{D}_b|$ is summation of all of the elements in the matrix $\boldsymbol{D}_b$. According to Equation (4.14), the node with more edges is sampled with a higher probability. Note that although the analog IC bipartite graph contains device nodes and net nodes, our proposed probability-based neighborhood sampling algorithm is performed to only sample device nodes while the net nodes will be automatically removed by our proposed heterogeneous multigraph representation as shown in Section 4.2.

The basic idea behind our proposed probability-based sampling method is that the device with more connections has a more important influence on the aging degradation of its neighboring transistors. Unlike traditional graph partitioning algo-

rithms [17], our proposed probability-based algorithm can obtain different subgraphs among different training epochs. In other words, even though the neighboring node and the node itself are partitioned into two isolated subgraphs in the current training epoch, they may be partitioned into the same subgraph in other training epochs. Thus our proposed probability-based neighborhood sampling algorithm can achieve more robust performance and good scalability.

Our proposed probability-based neighborhood sampling algorithm is summarized in Algorithm 6. We provide the search depth D, the transistor node set $\mathcal{V}_s$, the number of device nodes to be trained T and analog IC netlist bipartite graph $\mathcal{BG}$ as inputs. Firstly, we need to find a device node set $\mathcal{V}^{(D)}$ and a net node set $\mathcal{U}^{(D)}$, whose each element is equal or less $2D$ hops away from one of the transistor nodes in $\mathcal{V}_s$ (from line 2 to line 13). As mentioned above, the deep H-GCN model brings much more elements in the node set $\mathcal{V}^{(D)}$. In order to sample $T - |\mathcal{V}_s|$ elements from the node set $\mathcal{V}^{(D)}$, the sampling probability of each node is calculated by Equation (4.14) (from line 15 to line 17). Then all sampled nodes are combined with the transistor node set $\mathcal{V}_s$. At last, the algorithm outputs the bipartite subgraph with the device node set $\mathcal{V}_b$ with T device nodes and the net node set $\mathcal{U}^{(D)}$. By performing Algorithm 6 in each training epoch, different bipartite subgraphs can be trained so that our deep H-GCN model can achieve more robust performance. Besides, like the dropout technique in CNNs [109], our proposed

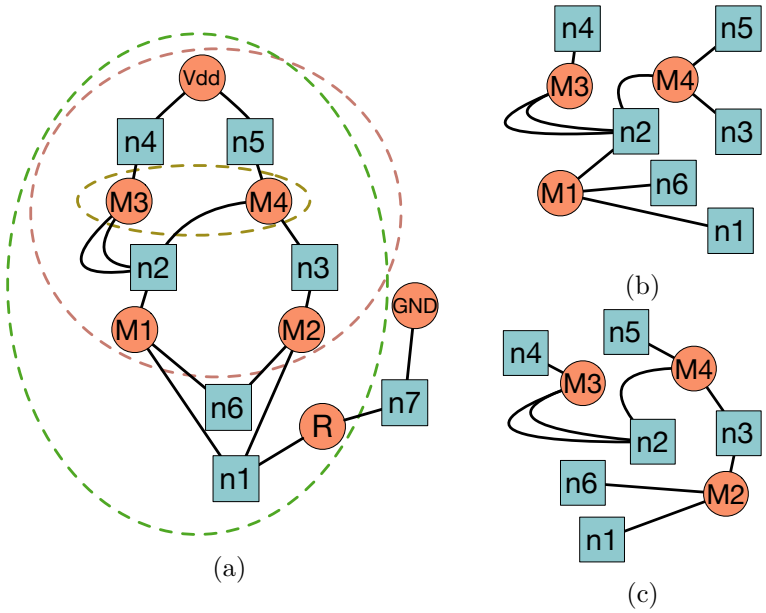


Figure 4.5 The neighborhood sampling.

probability-based neighborhood sampling algorithm can alleviate overfitting since it randomly drops embedding features from neighboring nodes during the training stage.

We take Fig. 4.5 as an example to illustrate the proposed probability-based neighborhood sampling algorithm. In this example, we set the search depth as $D = 3$ and the number of device nodes as $T = 3$. The initial input node set is $\mathcal{V}_s = \{\texttt{M3}, \texttt{M4}\}$. As shown in Fig. 4.5(a), the 6-hop device node set is $\mathcal{V}^{(3)} = \{\texttt{M1}, \texttt{M2}, \texttt{M3}, \texttt{M4}, \texttt{Vdd}, \texttt{R}\}$ and the 6-hop net node set is $\mathcal{U}^{(3)} = \{\texttt{n1}, \texttt{n2}, \texttt{n3}, \texttt{n4}, \texttt{n5}, \texttt{n6}\}$. Except for M3 and M4 in $\mathcal{V}_s$, we only need to sample one more node from $\mathcal{V}^{(3)} \backslash \mathcal{V}_s$. According to Equation (4.14), M1 and M2 have the largest probability to be sampled since they have the most edge number. If M1 is sampled, Algorithm 6 takes the subgraph with the device node

Algorithm 6 The Neighborhood Sampling Algorithm

Require: Search depth D, the input transistor node set $\mathcal{V}_s$, the number of device nodes to be trained T, analog IC netlist bipartite graph $\mathcal{BG}$;

Initialize device node set $\mathcal{V}^{(0)} \leftarrow \mathcal{V}_s$;
for $k = 1$ to D **do**
    $\mathcal{U}^{(k)} \leftarrow \emptyset$;
    for all $v \in \mathcal{V}^{(k-1)}$ **do**
        $\mathcal{U}^{(k)} \leftarrow \mathcal{U}^{(k)} \cup \mathcal{N}(v)$; ▷ expand the net node set.
    end for
    $\mathcal{V}^{(k)} \leftarrow \emptyset$;
    for all $\mu \in \mathcal{U}^{(k)}$ **do**
        $\mathcal{V}^{(k)} \leftarrow \mathcal{V}^{(k)} \cup \mathcal{N}(\mu)$; ▷ expand the device node set.
    end for
end for
$\mathcal{U}^{(D)} \leftarrow \cup_{k=1}^{D} \mathcal{U}^{(k)}$; ▷ combine the net node sets with $1, 3, \cdots, 2D-1$ hops distance away from one node of $\mathcal{V}_s$.
$\mathcal{V}^{(D)} \leftarrow \cup_{k=0}^{D} \mathcal{V}^{(k)}$; ▷ combine the device node sets with $0, 2, \cdots, 2D$ hops distance away from one node of $\mathcal{V}_s$.
if $|\mathcal{V}^{(D)}| > T$ **then**
    for all $v_i \in \mathcal{V}^{(D)} \backslash \mathcal{V}_s$ **do**
        Calculate $\mathcal{P}(v_i)$ according to Equation (4.14);
    end for
    Sample $T - |\mathcal{V}_s|$ nodes according to the probability $\mathcal{P}$ to the set $\mathcal{V}_b$;
    $\mathcal{V}_b \leftarrow \mathcal{V}_b \cup \mathcal{V}_s$;
else
    $\mathcal{V}_b \leftarrow \mathcal{V}^{(D)}$;
end if
return The bipartite subgraph with $\mathcal{V}_b \cup \mathcal{U}^{(D)}$ of $\mathcal{BG}$.

$\mathcal{V}_b = \{\texttt{M1}, \texttt{M3}, \texttt{M4}\}$ and net node $\mathcal{U}^{(3)}$ shown in Fig. 4.5(b) as an output. If `M2` is sampled, Algorithm 6 takes the subgraph with the device node $\mathcal{V}_b = \{\texttt{M2}, \texttt{M3}, \texttt{M4}\}$ and net node $\mathcal{U}^{(3)}$ shown in Fig. 4.5(c) as an output. The two subgraphs are trained in different training epochs.

4.6 Overall flow

The overall flow of our proposed fast analog IC aging-induced degradation estimation framework is illustrated in Fig. 4.6, which consists of four parts: parsing and flattening netlist, probability-based neighborhood sampling, heterogeneous graph representation and deep H-GCN inference. Our flow takes the netlist with static stress conditions as an input. Then netlist is parsed and flattened to be a bipartite graph. In order to achieve enough scalability, the probability-based neighborhood sampling as shown in section 4.5 is adopted to obtain a subgraph. Moreover, the subgraph is transferred to be the heterogeneous graph to efficiently represent the topology of analog ICs as shown in Section 4.2. The heterogeneous graph is input into the deep H-GCN to perform `dvtlin` estimation. At last, the overall flow takes the estimated `dvtlin` values of the target transistors as outputs. Based on the aforementioned techniques, the proposed flow is expected to achieve fast and accurate `dvtlin` estimation.

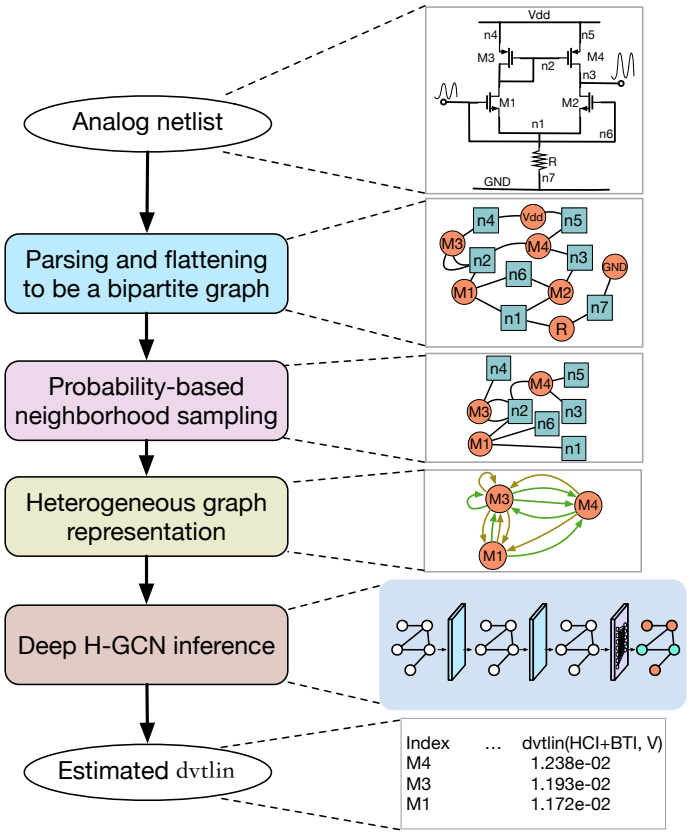


Figure 4.6 Overall flow.

4.7 Experimental Results

4.7.1 Benchmarks and Experimental Setting

Considering that the aging-related wear-out is very severe at the $5nm$ technology node in the industry, our experiments are illustrated on eight different industrial phase-locked loop designs implemented at the very advanced $5nm$ technology node. It should be noted that our proposed method is very general that it does not rely on any technology information or parameters, so it can be adopted in other technologies. These designs are represented

with post-layout netlists. The post-layout netlist files consist of Spectre and SPICE formats [65], as inputs in our models. We run an industrial aging DFR tool to obtain the `dvtlin` value of each transistor. Note that in each design, the stress conditions are given by very sophisticated designers to obtain `dvtlin` as a golden-truth value. To evaluate the estimation performance, each time, we use seven designs for training and the remaining design for testing.

Table 4.1 Statistics of Designs

Design	#trans.	#device	#net
1	4,348	99,009	18,155
2	4,382	99,696	18,299
3	3,999	179,758	31,303
4	3,998	185,480	33,819
5	4,980	692,480	111,308
6	523	31,279	6,002
7	6,398	452,109	76,807
8	1,998	96,749	16,006

Table 4.2 Device Type

Type	#Param.
MOS	51
MOS spice	75
DIO/ESD	8
Cap	12
R	6
VSource	1

The statistics of the eight industrial phase-locked loop designs

Table 4.3 Graph learning models

Method	Feature		Graph representation	
	Concat.	Latent	Homo.	Heter.
GCN	✓		✓	
GCNII	✓		✓	
H-GCN-concat	✓			✓
H-GCN		✓		✓
Deep H-GCN-concat	✓			✓
Deep H-GCN		✓		✓

Table 4.4 MAE (mV) and r^2 Score Comparisons.

Design	DFR tool (static)		GCN [40]		GCNII [22]		H-GCN		Deep H-GCN	
	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score
1	4.009	0.181	1.316	0.703	1.332	0.691	0.914	0.821	**0.824**	**0.843**
2	4.072	0.194	1.389	0.596	1.339	0.619	0.893	0.814	**0.856**	**0.839**
3	4.543	0.327	4.070	0.588	4.166	0.599	2.302	0.817	**2.012**	**0.840**
4	4.515	0.332	4.111	0.588	4.177	0.551	2.575	0.746	**2.350**	**0.815**
5	4.160	0.277	3.750	0.521	4.021	0.354	2.525	0.787	**2.454**	**0.816**
6	3.962	0.395	2.077	0.802	2.092	0.802	1.661	0.834	**1.541**	**0.865**
7	4.319	0.266	3.166	0.685	3.168	0.689	2.889	0.826	**2.704**	**0.874**
8	4.594	0.224	3.491	0.637	3.748	0.610	2.670	0.786	**2.503**	**0.840**

are shown in TABLE 4.1, where the netlist (Design 5) has up to 692K devices. The netlist of each design is parsed and flattened. Then all alternating current (AC) voltage sources (dynamic conditions), such as behavioral signal (Verilog-A description), sine wave, pulse and piecewise linear (pwl), are ignored while all DC voltage sources are considered so that our model can be independent of the dynamic stress conditions. Except for AC voltage sources, there are 32 types of basic devices in all designs. According to their parameters, all of these devices are divided into 6 categories as listed in TABLE 4.2. Their feature vectors have

different lengths, i.e., the #Param. Correspondingly, 6 matrices are adopted to map feature vectors into a unified latent space.

According to the domain knowledge, the four pins of transistors and two pins of diodes play an important role in circuit analysis [98]. Consequently, in the heterogeneous graph representation, all of the edges connecting to these six types of pins are emphasized by categorizing them into six types of edges. All of the edges connecting to other pins are uniformly treated as an individual type of edges. In total, seven adjacency matrices are used to specify various connection pins, i.e., gate, drain, source, substrate, anode, cathode and others.

Two traditional graph learning models, our proposed H-GCN and the extended deep H-GCN are implemented in the experiments, as listed in TABLE 4.3. GCN [40] and GCNII [22] models use the concatenated features representation and treat the analog IC netlist as homogeneous undirected multigraphs mentioned in Section 4.2. While our H-GCN and deep H-GCN use the proposed heterogeneous directed multigraph representation and the unified latent space mapping algorithm. In our H-GCN and deep H-GCN models, ReLU function is used as the activation function. We use MSE between the ground-truth and our estimated `dvtlin` as loss function with weight decay hyperparameter 10^{-7} and stochastic gradient descent for optimization. We set the embedding generation with search depth $D = 2, 4, 6, 8$ and 3 FC layers in the four graph learning models. And we set the device sampling size $T = 1000$ in Algorithm 6.

In order to illustrate the performance of graph learning models, all inferences are repeatedly performed ten times to report averages. Each output feature size of embedding generation layers and the latent space mapping layer is 512. The output feature sizes of three FC layers are set as 4096, 1024 and 1, respectively. The batch size is 32 and the number of the training epoch is 600. The same settings and configurations are used in GCN [40] and GCNII [22]. To improve numerical stability, we normalize all feature vectors and adjacency matrices. In order to illustrate the performance of these data-driven graph learning models, we also run dynamic and static aging reliability simulations by using the industrial aging DFR tool.

The proposed graph representation is implemented with Python and Networkx library. All graph learning models are implemented with TensorFlow, and are trained and tested on a Linux machine with 18 cores and NVIDIA Tesla V100 GPU with 32GB memory.

Mean absolute error (MAE) defined in Equation (4.15) is used as a metric to evaluate the absolute accuracy on the testing set.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|, \tag{4.15}$$

where i indicates the i-th transistor. y_i is the i-th transistor's `dvtlin`, as the golden-truth value, obtained by the industrial aging DFR tool with the appropriate stress given by very sophisticated designers. $\hat{y}_i$ is the estimated `dvtlin` of the i-th transistor by graph learning methods or the static reliability

simulation. n is the number of transistors on the testing set. In addition, we use r^2 score defined in Equation (4.16) as a metric to evaluate the relative accuracy on the testing set.

$$r^2\text{score} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{4.16}$$

where $\bar{y}$ denotes the mean of the golden-truth values y_i on the testing set. r^2 score indicates how the regression prediction perfectly fits the data. The higher r^2 and lower MAE mean the better accuracy.

4.7.2 Accuracy and Runtime

We train the four graph learning models as shown in TABLE 4.3 with the embedding generation with search depth $D = 2, 4, 6, 8$ and list their best accuracies in TABLE 4.4. Besides, we also list the accuracies of the static aging reliability simulation by using the industrial aging DFR tool. Compared with graph learning models, the static aging reliability simulation causes the worst accuracy since it completely ignores dynamic stress conditions. Thus our proposed data-driven approach with a supervised learning manner is effective to estimate aging-induced degradation.

According to TABLE 4.4, the typical GCN [40] and GCNII [22] have the worse accuracy, since they do not consider the heterogeneity of the analog circuit. Compared with our proposed H-GCN, our extended deep H-GCN can achieve a more

accurate estimation since it can extract information from multi-hop devices without an over-smoothing issue. The same reason can be used to explain why GCNII [22] can beat the typical GCN [40].

We compare the runtime of our extended deep H-GCN with static and dynamic aging simulations by using the industrial DFR tool as shown in Fig. 4.7, where DFR-D and DFR-S denote the dynamic and static aging reliability simulations, respectively. The static and dynamic aging reliability simulations need to perform fresh simulation, stress simulation, aging simulation as shown in Fig. 2.2. Thus they consume the longest time. Moreover, compared with static aging reliability simulations, the dynamic aging reliability simulation consumes a longer runtime since it needs a large number of accepted transient steps. Compared with the traditional dynamic and static aging reliability simulations, our proposed methods do not need computationally expensive fresh simulation, stress simulation or aging simulation. Thus our extended deep H-GCN can achieve significant speedup. According to Fig. 4.7, our extended deep H-GCN can achieve 241× and 39× speedup on average, respectively, comparing with the dynamic and static aging reliability simulations.

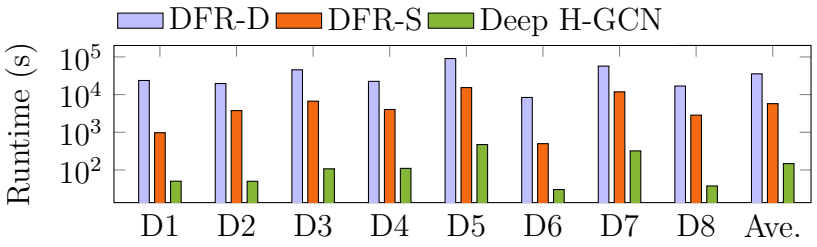


Figure 4.7 Runtime comparisons.

4.7.3 Ablation study

In this section, we give two ablation studies to illustrate the effectiveness of the deep structures and our proposed the unified latent space mapping algorithm.

The effectiveness of the deep structures and over-smoothing

The accuracies w.r.t. the search depth of embedding generation $D = 2, 4, 6, 8$ among different graph learning models are shown in Fig. 4.8. As the search depth of embedding generation increases, both GCN [40] and our H-GCN bring performance degradation because of the over-smoothing issue. In other words, they achieve the best accuracy with a shallow structure. While as the search depth of embedding generation increases, GCNII [22] and our deep H-GCN can achieve a more accurate estimation. Thus GCNII [22] and our extended deep H-GCN can alleviate the over-smoothing issue. However, compared with GCNII [22], our extended deep H-GCN can characterize the heterogeneity of the analog circuit so that it can achieve a more accurate estimation.

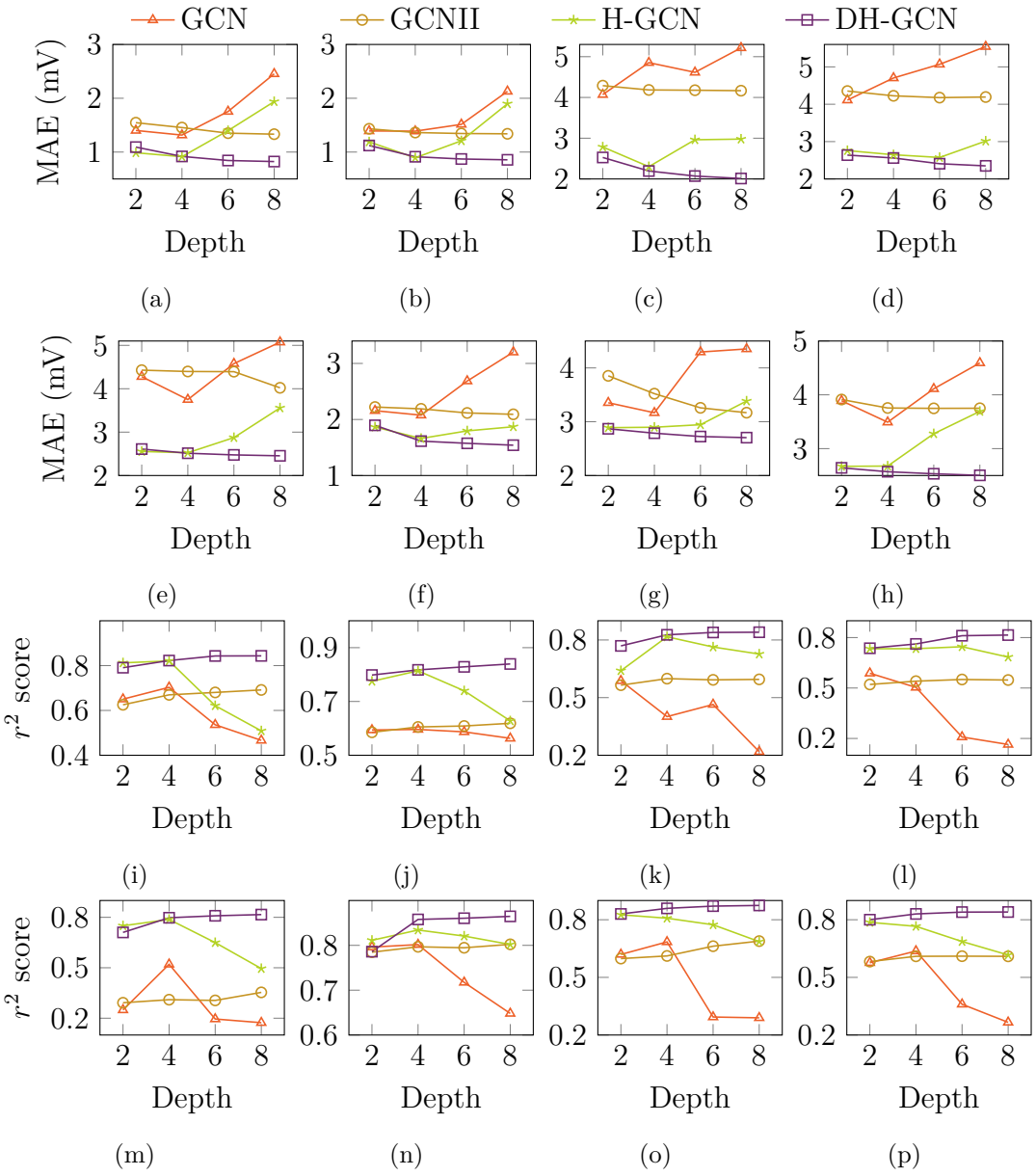


Figure 4.8 The embedding generation depth *vs* accuracy: (a-h): MAE on Design 1 to Design 8; (i-p): r^2 Score on Design 1 to Design 8 (DH-GCN denotes our extended deep H-GCN).

The effectiveness of the unified latent space mapping algorithm

We compare the accuracies of H-GCN and deep H-GCN with H-GCN-concat and deep H-GCN-concat, respectively. Here H-

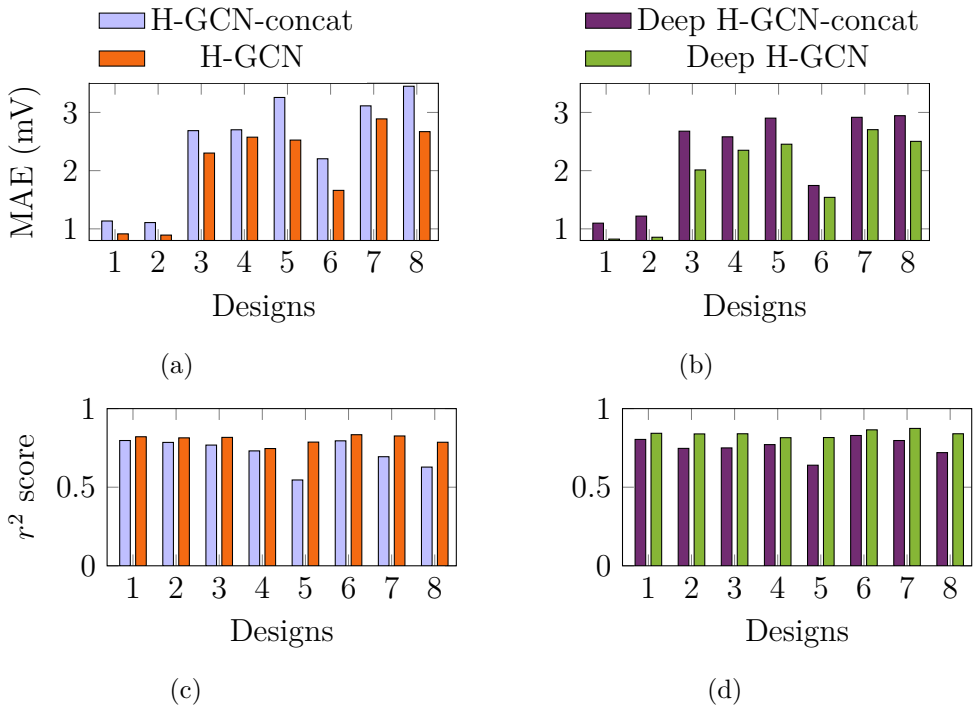


Figure 4.9 MAE and r^2 Score between HGCN and HGCN-concat and between deep H-GCN and deep H-GCN-concat: (a) MAE between H-GCN and H-GCN-concat; (b) MAE between Deep H-GCN and Deep H-GCN-concat; (c) r^2 Score between H-GCN and H-GCN-concat; (d) r^2 Score between Deep H-GCN and Deep H-GCN-concat.

GCN-concat and deep H-GCN-concat adopt the concatenated feature representation as listed in TABLE 4.3. For comparisons, according to Fig. 4.8, we choose the best configurations (the search depth of embedding generation D) for each model on each design. MAE and r^2 score among HGCN, HGCN-concat, deep H-GCN and deep H-GCN-concat are shown in Fig. 4.9. Our proposed unified latent space mapping algorithm can improve accuracy on both HGCN and deep H-GCN since it can effectively exploit and encode features.

4.8 Summary

In this chapter, we propose an H-GCN to fast estimate aging-induced transistor degradation in analog ICs. To characterize the multi-typed devices and connection pins, a heterogeneous directed multigraph is adopted to efficiently represent the topology of analog ICs. A latent space mapping method is used to transform the feature vector of all typed devices into a unified latent space. We further extend the proposed H-GCN to be a deep version via initial residual connections and identity mappings. The extended deep H-GCN can extract information from multi-hop devices without an over-smoothing issue. A probability-based neighborhood sampling method on the bipartite graph is adopted to ease the model training on large-scale graphs and achieve good scalability. We conduct experiments on very advanced $5nm$ industrial benchmarks. Compared with traditional graph learning methods and the static aging reliability simulations by using an industrial DFR tool, our proposed deep H-GCN can achieve more accurate estimations of aging-induced transistor degradation. Compared with the dynamic and static aging reliability simulations, our extended deep H-GCN can achieve on average 241× and 39× speedup, respectively. Thus our proposed deep H-GCN can significantly improve the efficiency of aging verification.

□ **End of chapter.**

Chapter 5

Thermal-driven PCB Routing

5.1 Preliminaries

5.1.1 Problem Formulation

Like the typical PCB routing frameworks, our TRouter takes post-placement layout and design rules as inputs. All components have been placed in the post-placement layout and these components have solder-mask-defined (SMD) pads and/or through-hole pads. SMD pad is a top- or bottom-layer pad that has a predefined assignment within the netlist. While the through-hole pad is a pad that punched through all routing layers that has a predefined assignment within the netlist. The netlist defines the connections among these SMD and/or through-hole pads. Note that different from the traditional escape routing and the area routing, our objective is to route more practical PCB, where there are typical BGAs and non-BGAs, such as SMD pads and through-hole pads scattered around the

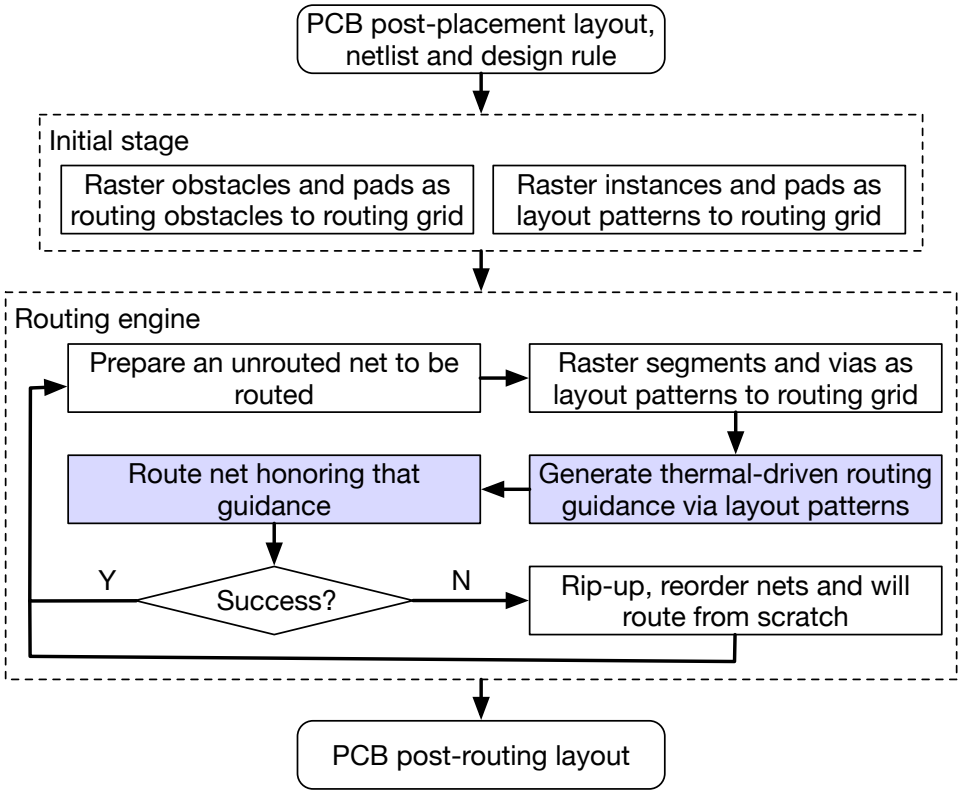


Figure 5.1 Our TRouter overview. The purple boxes denote our contributions.

routing plane.

Specifically, one of our objectives is to minimize the thermal values on the post-routing layout. Another objective is to minimize wirelength, like the typical routing objective. We provide the formal definition of our thermal-driven PCB routing problem.

Problem 3 (Thermal-driven PCB routing)**.** *Given a set of SMD pads, a set of through-hole pads, a netlist, and design rules, connect all the nets so that there is no design rule violation, the total wirelength is minimized while minimizing the thermal*

values on the post-routing layout.

In order to handle Problem 3, we further decompose it into two sub-problems: thermal-driven routing guide generation and guided detailed routing defined as follows.

Problem 4 (Thermal-driven routing guide generation)**.** *Given the placement of all components with pads, some vias and routing segments (traces), predict a cost in each grid cell while minimizing the thermal values on the post-routing layout.*

Problem 5 (Guided detailed routing)**.** *Guided detailed routing takes placement of all components with pads, nets, design rules and a set of generated routing guidance as input, and routes all the nets while honoring routing guidance and satisfying design rules.*

5.1.2 Overview

In order to handle Problem 3, we propose TRouter, a thermal-driven PCB routing framework via CNNs. We devise our TRouter based on the grid-based model since it has a simple data structure, typically constructs routing models quickly, and allows fast neighbor identification and one-step move [108]. Initially, the routing obstacles are captured by rasterizing the arbitrary shapes of obstacles and pads to the routing grid. Meanwhile the layout patterns containing instances and pads are captured by rasterizing the arbitrary shapes of instances and pads to the

routing grid. Then PCB routing is iteratively performed. In each iteration, routing segments and vias are captured by rasterizing them to the routing grid. Then the thermal-driven routing guidance (Problem 4 task) is generated at each grid cell before one net is routed by honoring that guidance (Problem 5 task). Besides, typical tricks such as ripping-up, reordering nets and routing from scratch are leveraged to improve routability. After all nets are well routed, our TRouter takes the post-routing layout as an output. Our TRouter overview is shown in Fig. 5.1.

5.2 Thermal-driven routing guidance generation

In this section, we present a thermal-driven routing guidance generation. Considering that it is difficult to directly formulate the relationship between routing layout and thermal distribution via analytic models, we adopt CNNs to predict thermal distribution by inputting the routing layout since they have a more powerful ability to fit a complex relationship via learning from historical data. In order to bridge the gap between CNNs and the routing engine, and avoid time-consuming and tricky interaction with the CAD tool, the PCB layout needs to be mapped into a routing grid. This routing grid can handle irregular BGA packages [74] and take layout patterns (i.e., instances, pads, vias and segments) as features to input into CNNs. Unlike the previous work [161] where the inference is only performed once when

the post-placement layout is input into the router, our TRouter needs interaction between the routing engine and CNNs for each one or several nets to be routed. Besides, after CNNs are well-trained, unlike the traditional work [71] where the inference has to be performed in each search step, our TRouter adopts gradient values in each grid cell obtained from the backpropagation of CNNs to guide routing.

According to the discussion above, our thermal-driven routing guidance generation consists of three tasks: (1) feature extraction from routing layout; (2) CNNs structure and training; (3) gradient value generation in each routing grid cell.

5.2.1 Feature extraction from routing layout

In order to extract feature patterns from the layout and perform grid-based routing, we rasterize the whole layout as shown in Fig. 5.2(a). The rasterizing can facilitate routing on the layout since these arbitrary shapes along with the pads are rasterized to a 3-D routing grid as obstacles to facilitate later stages to derive a design-rule-violation-free routing solution. In addition, rasterizing can help extract feature patterns from the layout without the help of other CAD tools. We rasterize all instances, pads, vias and segments as features since their position and shape play an important role in thermal distribution. Note that in this example as shown in Fig. 5.2(a), the PCB has two layers: top and bottom. Thus based on the 3-D routing grid, we can directly

extract these patterns as shown in Figs. 5.2(b) to 5.2(g). Each pattern can be naturally encoded as a binary 3-D tensor, where the value is 1 if the region is occupied. Otherwise, the value is 0. Unlike the previous work [161] where a CAD tool is used to obtain a post-placement layout image as features, our TRouter can avoid time-consuming and tricky interaction with any CAD tool to improve routing speed.

5.2.2 CNNs structure and training

After the pattern features are extracted, CNNs need to be built to predict the thermal distribution by inputting these pattern features. Since there are PCB layouts of arbitrary size in different designs and the thermal distribution map has the same size as the PCB layer in each design as shown in Fig. 5.3, we need to adopt fully convolutional networks (FCNs). Compared with classification-architectures-based CNNs, FCNs discard the final classifier layer, convert all fully-connected layers to convolutions and devise a deconvolution layer to bilinearly upsample the coarse outputs to pixel-dense outputs [81]. By these customized structures, FCNs can take input of the arbitrary size and produce correspondingly-sized output with efficient inference and training.

Typical FCNs have the limited ability to perform feature fusion with low-level features and learn from very little available training data. Thus the typical FCNs are developed to be U-

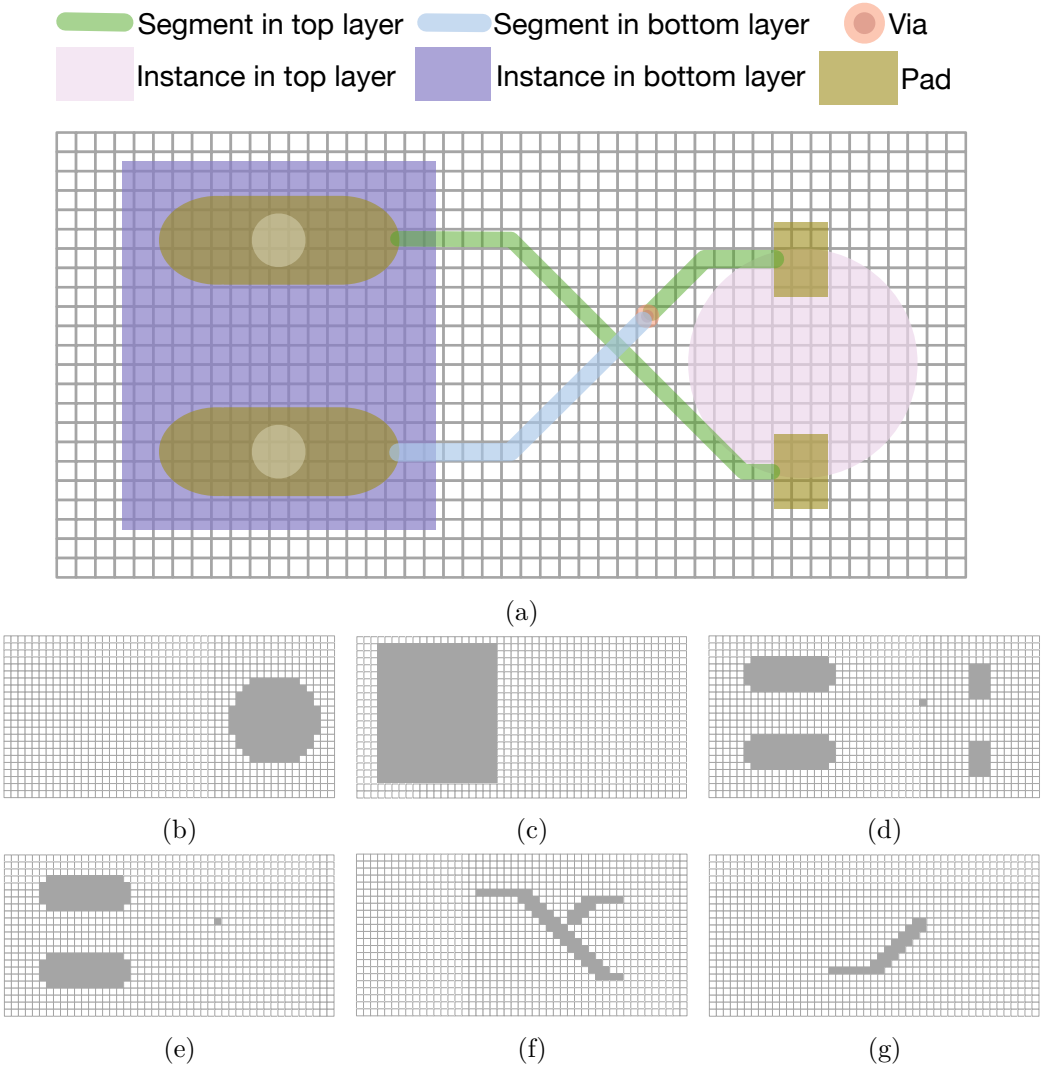


Figure 5.2 A two-layer layout and feature patterns. (a) routing grid; (b) instance pattern on the top layer; (c) instance pattern on the bottom layer; (d) pad/via pattern on the top layer; (e) pad/via pattern on the bottom layer; (f) segment pattern on the top layer; (g) segment pattern on the bottom layer. Grey cell means that the cell value is 1 and white cell means that cell value is 0.

net by adopting skip-connections to perform feature fusion with low-level features in up-sampling parts [102]. Thus it can effi-

ciently learn from very little available training data. Thus we adopt U-net as our basic backbone as shown in Fig. 5.3. The channel number of the model output relies on the thermal simulator. Here we uniformity set it as 1. Besides, according to the feature patterns as shown in Fig. 5.2, the l-layer layout is encoded as a tensor with $2 + 2l$ input channels. Thus we adopt different configurations to predict thermal distribution for the layout with different number of layers as shown in Fig. 5.3.

Formally, the U-net-based thermal distribution prediction model $\phi_{\boldsymbol{W}}$ takes instance pattern tensor $\boldsymbol{\mathcal{X}}_{in}$, pad/via pattern tensor $\boldsymbol{\mathcal{X}}_{pv}$ and segment pattern tensor $\boldsymbol{\mathcal{X}}_{sg}$ as inputs then outputs the predicted thermal distribution $\hat{\boldsymbol{Y}}$ as follows.

$$\hat{\boldsymbol{Y}} = \phi_{\boldsymbol{W}}(\boldsymbol{\mathcal{X}}_{in}, \boldsymbol{\mathcal{X}}_{pv}, \boldsymbol{\mathcal{X}}_{sg}), \tag{5.1}$$

where $\boldsymbol{W}$ denotes the trainable model coefficients in the U-net-based thermal distribution prediction model. They are determined by minimizing MSE, as an empirical loss function, between the predicted thermal distribution and ground-truth as follows.

$$\mathcal{L}(\boldsymbol{W}) = ||\boldsymbol{Y} - \phi_{\boldsymbol{W}}(\boldsymbol{\mathcal{X}}_{in}, \boldsymbol{\mathcal{X}}_{pv}, \boldsymbol{\mathcal{X}}_{sg})||_2^2, \tag{5.2}$$

where $\boldsymbol{Y}$ represents the ground-truth thermal distribution matrix. Minimizing Equation (5.2) can be handled by SGD in the training stage.

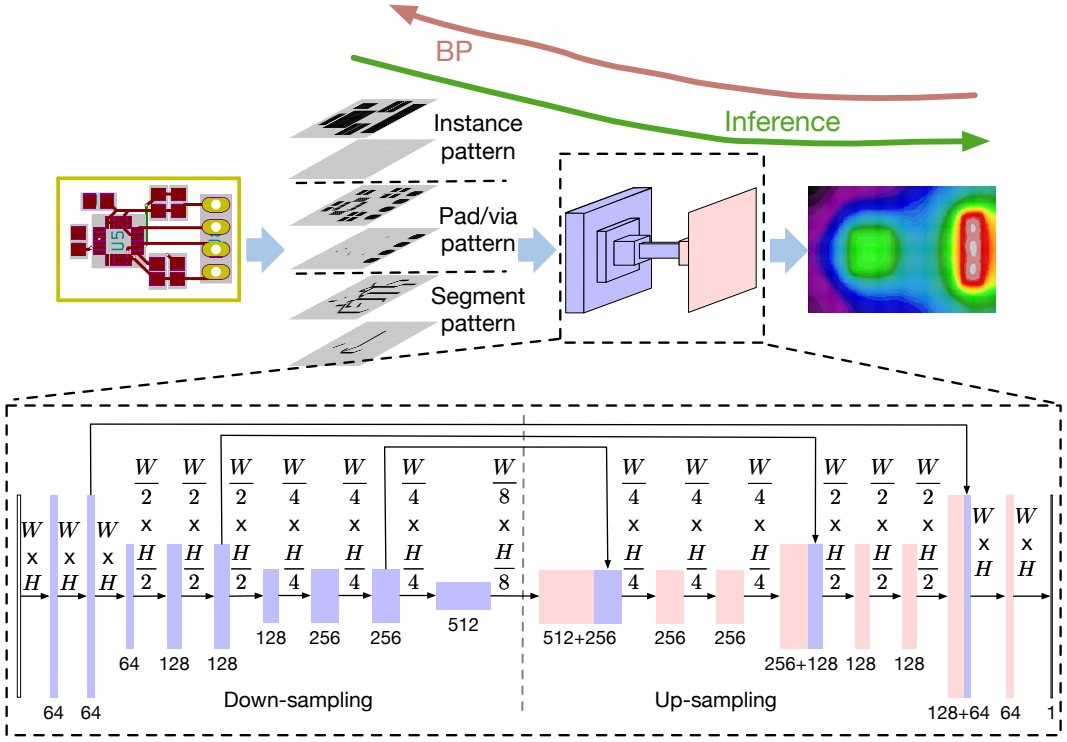


Figure 5.3 U-net-based thermal distribution prediction model. W and H denote the width and height of input feature tensor, respectively.

5.2.3 Gradient value generation in routing grid cell

Until now, we have proposed a U-net-based model to predict thermal distribution. However, the grand challenge is how to use the well-trained U-net-based model to guide routing and achieve high quality and thermal well layout. The typical method is that each routing step is evaluated by performing the inference of the U-net-based model [71]. However, this strategy is very time-consuming. We need to adopt a more efficient strategy to guide routing via the well-trained U-net-based model.

In this work, we propose to use gradient values obtained from

the backpropagation of the well-trained U-net-based model as a cost at each routing grid cell. The main idea behind the strategy is that the gradient of input features can provide guidance for reducing thermal. One of our routing objectives is to minimize the thermal values on the post-routing layout, that is

$$\min_{\boldsymbol{\mathcal{X}}_{pv},\boldsymbol{\mathcal{X}}_{sg}} \quad ||\phi_{\boldsymbol{W}}(\boldsymbol{\mathcal{X}}_{in},\boldsymbol{\mathcal{X}}_{pv},\boldsymbol{\mathcal{X}}_{sg})||_2^2. \tag{5.3}$$

We denote $\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv},\boldsymbol{x}_{sg}) \triangleq ||\phi_{\boldsymbol{W}}(\boldsymbol{\mathcal{X}}_{in},\boldsymbol{\mathcal{X}}_{pv},\boldsymbol{\mathcal{X}}_{sg})||_2^2$, where the tensor $\boldsymbol{\mathcal{X}}_{\cdot}$ is vectorized as $\boldsymbol{x}_{\cdot}$. Then the first-order Taylor expansion of $\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv},\boldsymbol{x}_{sg})$ can be expressed as follows.

$$\begin{aligned}\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv}^{(i)}+\Delta\boldsymbol{x}_{pv}^{(i)},\boldsymbol{x}_{sg}^{(i)}+\Delta\boldsymbol{x}_{sg}^{(i)}) &\approx \psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv}^{(i)},\boldsymbol{x}_{sg}^{(i)})\\ +(\frac{\partial\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv}^{(i)},\boldsymbol{x}_{sg}^{(i)})}{\partial\boldsymbol{x}_{pv}})^\top\Delta\boldsymbol{x}_{pv}^{(i)} &+ (\frac{\partial\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv}^{(i)},\boldsymbol{x}_{sg}^{(i)})}{\partial\boldsymbol{x}_{sg}})^\top\Delta\boldsymbol{x}_{sg}^{(i)},\end{aligned} \tag{5.4}$$

where i denotes that the i-th routing step. $\Delta\boldsymbol{x}_{pv}^{(i)}$ and $\Delta\boldsymbol{x}_{sg}^{(i)}$ mean additional vias and segments for routing in the next step, respectively. According to our feature pattern definition, routing is the process where all unoccupied regions (values are 0 in pattern tensor) are chosen to deploy segments or vias (values are 1 in pattern tensor). Thus $\Delta\boldsymbol{x}_{sg}^{(i)},\Delta\boldsymbol{x}_{pv}^{(i)} \in \{0,1\}^n$, where n is total number of cells in the routing grid. Thus the smaller gradient values $\partial\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv}^{(i)},\boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{pv}$ and $\partial\psi(\boldsymbol{x}_{in},\boldsymbol{x}_{pv}^{(i)},\boldsymbol{x}_{sg}^{(i)})/\partial\boldsymbol{x}_{sg}$ have negative effects on temperature increasing in the routing step. As a result, in each or several routing steps, the layout patterns as shown in Fig. 5.3 are input into the U-net-based

thermal distribution prediction model then the back propagation is performed to obtain gradient value $r(\boldsymbol{c})$ as a cost in each routing grid cell, where $\boldsymbol{c}$ is the coordinate in layout or pattern tensor.

5.3 Guided detailed routing

5.3.1 Detailed routing

After the thermal-driven routing guidance is obtained by U-net-based model, our TRouter adopts the grid-based A* searching algorithm for detailed routing. Different from traditional routing algorithm, where the searching costs only contain from the source to the current location and the current location to the target, our routing scheme will honor thermal-driven routing guidance. Thus our A* searching cost function is defined as follows.

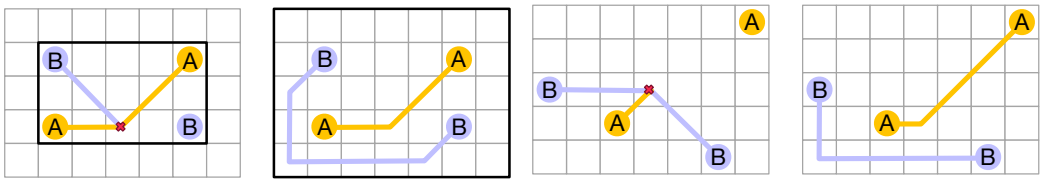
$$f(\boldsymbol{c}) = g(\boldsymbol{s}, \boldsymbol{c}) + h(\boldsymbol{c}, \boldsymbol{t}) + \lambda r(\boldsymbol{c}), \tag{5.5}$$

where $g(\boldsymbol{s}, \boldsymbol{c})$ is the cost from the source $\boldsymbol{s}$ to the current location $\boldsymbol{c}$ and $h(\boldsymbol{c}, \boldsymbol{t})$ is the estimated cost from the current location $\boldsymbol{c}$ to the target $\boldsymbol{t}$. $r(\boldsymbol{c})$ is the cost in the current location $\boldsymbol{c}$ generated by thermal-driven routing guidance as shown in Section 5.2. λ is a hyper-parameter controlling the trade-off between thermal distribution and routing wirelength. Like [74], $h(\boldsymbol{c}, \boldsymbol{t})$ is defined as $\max D - \min D + \sqrt{2} \min D$ to support the routing angle of 135° and 90° degree, where $\min D$ is the minimum difference of a

location $\boldsymbol{c}$ and a target along with x-axis or y-axis and max D is the maximum one. $D = \{|c.x - t.x|, |c.y - t.y|\}$. By performing A* searching algorithm on the whole PCB layout region to route net one by one, all nets can be routed on the PCB layout.

5.3.2 Speedup via an adaptive bounding-box

In the traditional detailed routing, the A* searching algorithm is used to search the next routing location on the whole PCB layout region. The large space brings time-consuming searching steps. In order to reduce searching space, we propose a routing scheme by using an adaptive-size bounding-box. The A* searching is limited in a bounding-box, which is initialized to cover source and target. The size of the bounding-box will be gradually enlarged if there is no legal path (no crossing to segments of other nets) from source to target. In this routing scheme, traditional soft obstacles may cause segment crossing among different nets. Thus, in our routing scheme, hard obstacles are used to avoid segment crossing. In other words, A* searching algorithm does not search the regions occupied by segments, vias and pads of other nets. As shown in Fig. 5.4(a), after net A is routed, some regions are occupied. To reduce search space, a bounding-box is initialized to cover the source and target before net B is routed. However, there is no legal path to route net B within the initial bounding-box. Then the bounding-box is enlarged until there is a legal path to route net B. In addition, the traditional fixed



(a) (b)

Figure 5.4 (a) The adaptive-size bounding-box. Left: a bounding-box is initialized to cover the source and target before net B is routed; Right: the bounding-box is gradually enlarged since there is no legal path to route net B within the initial bounding-box. (b) The reordering routing from scratch. Left: There is no legal path to route net A after net B is routed; Right: All segments are removed and net A is routed firstly.

nets order may result in a routing failure. In order to enhance routability, a scratch routing mechanism is adopted. Once one net cannot be routed successfully, it is firstly routed after all segments of routed nets are removed (ripping-up mechanism). As shown in Fig. 5.4(b), after the net B is routed, there is no legal path to route net A. By using our ripping-up mechanism, all segments are removed and net A is routed firstly.

After the detailed routing is finished, the routed layout is exported into kicad and OrCAD DSN formats. By using the routing scheme via an adaptive-size bounding-box with hard obstacles and ripping-up mechanism, it is expected to achieve significant routing speedup.

5.4 Experimental Results

We implemented our algorithm in the C++ programming language. Boost C++ library [1] is used to calculate the geometric computation within our proposed algorithm. Pytorch library is used to train our CNNs and C++ libtorch library is used to perform inference and BP [8]. Routing is performed on Linux machine with 10 cores and NVIDIA TITAN XP GPU with 12GB memory. In our TRouter, inference and BP are performed on GPU while other operations are performed on CPU. The state-of-the-art PCBRouter [74], as baseline, is performed on CPU. Our benchmarks are come from open-source designs [6,7]. Their statistics is shown in TABLE 5.1, where $W \times H$ denotes the dimension of width and height for the designs. $|L|$, $|C|$, $|P|$ and $|N|$ denote the numbers of PCB routing layers, components, pads, and nets, respectively. Siemens HyperLynx software [4] is used to perform the thermal simulation.

Since there are two kinds of layers in our benchmarks, we need to build two models to predict the thermal distribution in our benchmarks. In other words, one model is built for PCB13, PCB14 and PCB15 and the other is built for rest designs. In each design, the layout of each routing step and its thermal distribution are collected as data set. Each layout and its thermal distribution are routed 90°, 180° and 270° for data augmentation. In order to evaluate the prediction performance for each model, we use one of designs as test case, and others as training

Table 5.1 Benchmark PCB Design statistics.

Designs	W×H (mm)	$\|L\|$	$\|C\|$	$\|P\|$	$\|N\|$
PCB1	21×14	2	8	40	15
PCB2	51×23	2	18	77	34
PCB3	55×28	2	34	138	38
PCB4	23×60	2	28	140	52
PCB5	41×42	2	48	163	54
PCB6	65×55	2	48	190	62
PCB7	51×23	2	46	207	69
PCB8	57×87	2	36	188	70
PCB9	44×36	2	58	229	80
PCB10	102×54	2	57	319	99
PCB11	89×58	2	64	401	134
PCB12	44×45	2	13	118	50
PCB13	58×60	4	58	233	35
PCB14	86×72	4	61	314	63
PCB15	94×63	4	276	1510	272

Table 5.2 Routing results.

Design	PCBRouter [74]					Our TRouter				
	WL (mm)	#Via	RT (s)	max($\boldsymbol{Y}$) (°C)	$\|\|\boldsymbol{Y}\|\|/\#gcell$	WL (mm)	#Via	RT (s)	max($\boldsymbol{Y}$) (°C)	$\|\|\boldsymbol{Y}\|\|/\#gcell$
PCB1	83.300	9	3	26.6	0.149	84.653	8	7	26.2 (↓0.4)	0.144 (↓0.005)
PCB2	273.917	15	1	62.0	0.343	271.780	16	5	60.9 (↓1.1)	0.337 (↓0.006)
PCB3	505.534	38	35	29.1	0.138	497.655	39	18	26.5 (↓2.6)	0.137 (↓0.001)
PCB4	754.216	56	7	58.9	0.327	775.199	71	16	56.3 (↓2.6)	0.318 (↓0.009)
PCB5	980.395	84	50	36.3	0.165	915.117	79	89	33.9 (↓2.4)	0.150 (↓0.015)
PCB6	956.218	49	146	28.2	0.144	961.246	56	151	26.3 (↓1.9)	0.139 (↓0.005)
PCB7	1093.068	166	35	53.4	0.300	1108.766	158	63	52.2 (↓1.2)	0.292 (↓0.008)
PCB8	1233.115	17	17	85.2	0.386	1142.796	20	36	84.5 (↓0.7)	0.382 (↓0.004)
PCB9	1178.936	117	127	31.9	0.161	1071.152	100	43	30.3 (↓1.6)	0.159 (↓0.002)
PCB10	5095.385	595	246	39.9	0.187	4528.277	464	337	37.3 (↓2.6)	0.183 (↓0.004)
PCB11	4140.010	381	401	31.3	0.154	3769.905	361	315	28.4 (↓2.9)	0.149 (↓0.005)
PCB12	1039.218	5	46	49.4	0.227	1039.422	5	15	49.4 (↓0.0)	0.226 (↓0.001)
PCB13	1536.219	88	62	39.3	0.177	1559.212	88	62	36.6 (↓2.7)	0.165 (↓0.012)
PCB14	3129.609	191	743	49.2	0.219	3139.936	194	282	46.2 (↓3.0)	0.207 (↓0.012)
PCB15	7954.524	1278	3627	51.1	0.287	7646.568	1222	1645	48.9 (↓2.2)	0.274 (↓0.013)
Ave.	1996.911	206	370 (1.80)	44.8	0.224	1900.779	192	206 (1.00)	42.9 (↓1.9)	0.217 (↓0.007)

data. The root mean squared error (RMSE) is used to evaluate the error between prediction and ground truth as shown in

Fig. 5.5. We can see that our thermal distribution prediction models can achieve an accurate predictions.

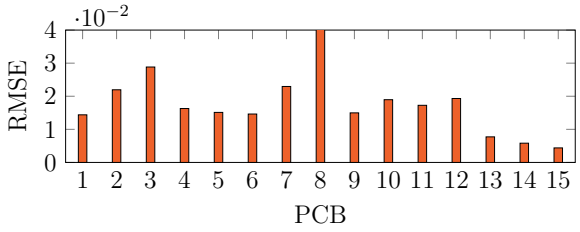


Figure 5.5 Model prediction performance.

We use the thermal distribution prediction models to guide PCB routing in our TRouter. BP is performed for routing each five nets. We compare our TRouter with state-of-the-art PCBRouter [74] as shown in TABLE 5.2. $\max(\boldsymbol{Y})$ and $||\boldsymbol{Y}||/\#gcell$ are used to measure thermal performance, where $||\cdot||$ denotes ℓ_2 norm. We can see that our TRouter can beat PCBRouter [74]. In particular, our TRouter can achieve 3°C maximum temperature reduction at most and 1.8× speedup on average. Besides, thanks to our proposed adaptive bounding-box and scratch routing mechanism, our TRouter can reduce the wirelength (WL) and the number of vias even if they both achieve 0 design rule violation and 100% routability for each design.

In order to illustrate the runtime performance, we profile our TRouter's runtime on one of small designs (PCB1) and one of large designs (PCB15) as shown in Fig. 5.6. When the small design is routed, the runtime is dominated by loading model as shown in Fig. 5.6(a). While the large design routed, the runtime

of loading model can be totally ignored ($\leq 1\%$) as shown in Fig. 5.6(b). Besides, thanks to performing BP on GPU, it does not dominate the routing runtime.

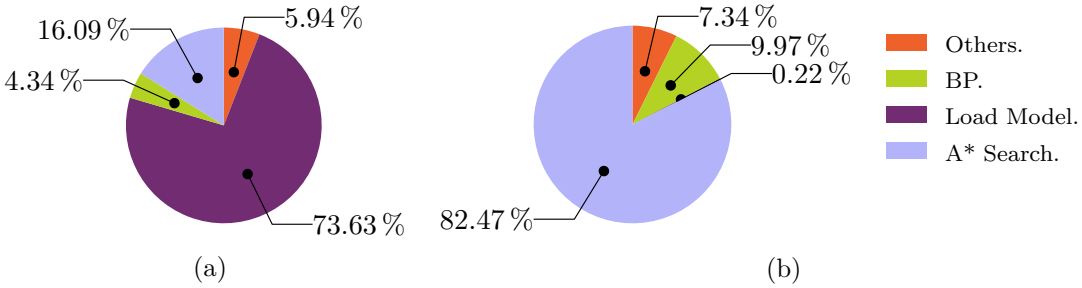


Figure 5.6 Runtime profiling: (a) PCB1; (b) PCB15.

5.5 Summary

In this chapter, we propose TRouter, a thermal-driven PCB routing framework via convolutional neural networks (CNNs). A U-net-based model are leveraged to predict thermal distribution by taking the routing layout as an input. A gradient in each routing grid cell obtained from the backpropagation of the U-net-based model is integrated into a full-board routing algorithm to guide thermal-driven routing. To achieve a significant speedup, an adaptive-size bounding-box is adopted to reduce routing search space. We conduct experiments on open-source benchmarks to illustrate our TRouter can achieve significant speedup and thermal well layouts, comparing with state-of-the-art PCB routing algorithm. Moreover, the methodology of our TRouter can be easily extended to other performance- or

reliability-driven routing frameworks.

□ **End of chapter.**

Chapter 6

Bayesian Sharing Grouped Convolution

6.1 Sharing Grouped Convolution

In this chapter, a sharing grouped convolution structure is proposed to reduce parameter redundancy, improve parameter efficiency. Then the number of parameters in it is analyzed to illustrate the compression performance.

To demonstrate the vanilla grouped convolution, the variation from ResNet to ResNeXt [134, 155] is taken as an example. Fig. 6.1 shows their basic block, which is repeatedly stacked with different configurations to the whole model. The basic block contains a shortcut and three convolutional layers, whose all kernels are represented by a box in each layer. In ResNet, the basic block is shown in Fig. 6.1(a), where there are three traditional convolutional layers. In order to transfer ResNet to ResNeXt, in each block, the second convolutional layer is trans-

ferred as the vanilla grouped convolution by dividing the 64 channels into 16 groups, and each group has 4 channels for this example as shown in Fig. 6.1(b). Compared with the traditional convolution, the vanilla grouped convolution adopts the sparse convolution connections between input and output channels, by dividing the input channels, output channels, and their connections into several groups. According to Fig. 6.1, the parameter number for the second convolutional layer is reduced from $64 \times 3 \times 3 \times 64 = 36864$ of ResNet to $16 \times 4 \times 3 \times 3 \times 4 = 2304$ of ResNeXt in the convolutional layer.

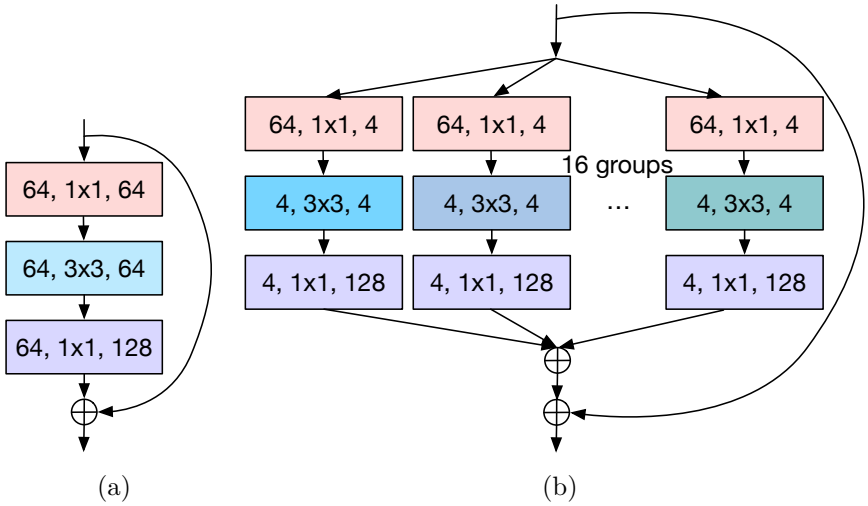


Figure 6.1 The basic block contains a shortcut and three convolutional layers (the boxes indicate the convolutional kernels [#input channel, kernel size, #output channel] for each layer): (a) the three convolutional layers in ResNet; (b) the two convolutional layers and one vanilla grouped convolutional layer with 16 groups in ResNeXt.

In order to further reduce the parameter number and improve parameter efficiency in the vanilla grouped convolution, we pro-

pose a sharing grouped convolution structure. Specifically, all groups share the same parameters so that the same parameters can be used to extract features and pass information among different groups. It has the same manner with [149] to improve parameter efficiency. Then in each basic block, the vanilla grouped convolution (the second layer) as shown in Fig. 6.1(b) will be transferred as the sharing grouped convolution as shown in Fig. 6.2. The parameter number for the second convolutional layer is reduced from $16\times4\times3\times3\times4 = 2304$ to $4\times3\times3\times4 = 144$. Note that compared with the vanilla grouped convolution, our proposed sharing grouped convolution does not reduce computational complexity. However, as shown in Fig. 6.2, the parameters are shared among different groups. Therefore, the efficiency of parameters is improved and the parameter redundancy can be reduced. Besides, the sharing grouped convolution can facilitate the weights reusing strategy in the hardware level implementations so that the actual number of memory accesses decreases significantly and the inference runtime reduces.

As comparison, we show the numbers of parameters of basic blocks in ResNet, ResNext and the sharing ResNeXt in TABLE 6.1. Compared with ResNet, ResNeXt has fewer parameters, and the proposed sharing ResNeXt can further reduce the number of parameters.

Although the proposed sharing grouped convolution structure can reduce parameters and improve the efficiency of parameters, directly training models with the group convolution

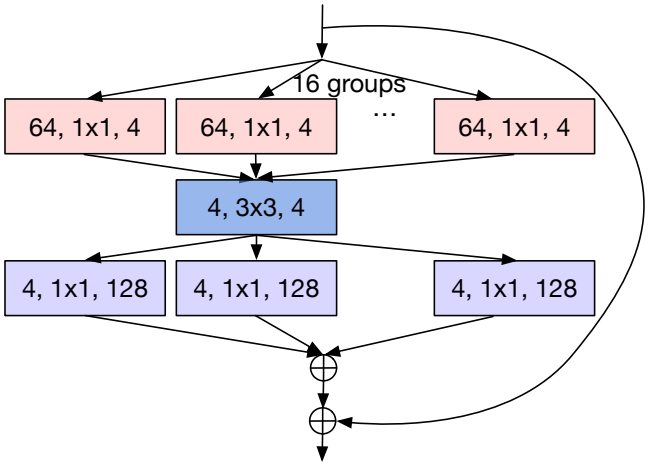


Figure 6.2 The basic block contains a shortcut, two convolutional layers and one sharing grouped convolutional layer with 16 groups in the sharing ResNeXt (the boxes indicate the convolutional kernels [#input channel, kernel size, #output channel] for each layer).

Table 6.1 The numbers of parameters of basic blocks in ResNet, ResNext and the sharing ResNeXt with $g = 16$.

Type	ResNet	ResNeXt	Sharing ResNeXt
conv	$64 \times 1 \times 1 \times 64$	$64 \times 1 \times 1 \times 64$	$64 \times 1 \times 1 \times 64$
(g)conv	$64 \times 3 \times 3 \times 64$	$16 \times 4 \times 3 \times 3 \times 4$	$4 \times 3 \times 3 \times 4$
conv	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$
shortcut	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$	$64 \times 1 \times 1 \times 128$
total	57344	22784	**20624**

structure may cause performance degradation since the correlation among parameters and groups does not be considered.

6.2 Bayesian Sharing Framework

To transfer the vanilla grouped convolution into the sharing structure, a naïve method is directly constructing a network

with the proposed sharing grouped convolution structure then training it. However, this method may cause performance degradation. To avoid performance degradation, we adopt a separate-merge methodology [44], that is updating independently parameters among all groups in the back-propagation stage and computing loss function value by the shared parameters in the forward propagation stage. Based on the separate-merge methodology, a typical method is indiscriminately averaging the parameters among different groups in the forward propagation stage. Given a pre-trained model, we can directly average these parameters among different groups. This is quite straightforward but it ignores the diversities of different groups.

In this section, to efficiently eliminate parameter redundancy and improve model performance, we introduce the **intra-group correlation** and **inter-group importance** of parameters. Then we propose a Bayesian sharing framework. Some notations used in this paper are listed in TABLE 6.2.

6.2.1 Intra-group Correlation and Inter-group Importance

To introduce **intra-group correlation** and **inter-group importance**, a prior distribution on model parameters $\mathcal{P}(\boldsymbol{w})$ is firstly introduced in our framework. Following previous arts [58,82,122], $\mathcal{P}(\boldsymbol{w})$ is defined to be a multivariate Gaussian distribution. For the convenience of expressions, network parameters

Table 6.2 List of Notations.

Name	Definition
$\boldsymbol{w}$	parameters in one grouped convolutional layer
g	# of groups in one grouped convolutional layer
$\boldsymbol{B}_i$	intra-group correlation of the group i
γ_i	inter-group importance of the group i
$\boldsymbol{w}_i$	parameters of the group i
$\boldsymbol{w}_b$	the sharing parameters of g groups
k	kernel size
Ci'	# of input channels in each group
Co'	# of output channels in each group
H	height of input feature
W	width of input feature

are reshaped to be vectors. As shown in Fig. 6.3, in each group, the kernels in each channel are flattened to be a vector. Some notations are explained in TABLE 6.2. Then they are concatenated sequentially to be a vector. Considering that the features are extracted independently from different groups in the vanilla grouped convolution as shown in Fig. 2.4(a), we assume any two network parameters from different groups are independent, i.e., $\mathcal{P}(\boldsymbol{w}) = \prod_i^g \mathcal{P}(\boldsymbol{w}_i)$. Therefore, the prior distribution of network parameters in the group i is defined as follows:

$$\mathcal{P}(\boldsymbol{w}_i; \gamma_i, \boldsymbol{B}_i) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{w}_i}, \boldsymbol{\Sigma}_{\boldsymbol{w}_i}), \ \boldsymbol{\Sigma}_{\boldsymbol{w}_i} \triangleq \gamma_i \boldsymbol{B}_i, \tag{6.1}$$

where $\boldsymbol{w}_i \in \mathbb{R}^{NCo'}$, and $\boldsymbol{B}_i \in \mathbb{R}^{NCo' \times NCo'}$ with $N \triangleq k^2 Ci'$. $\boldsymbol{B}_i$ is a positive definite matrix which captures the correlations of the

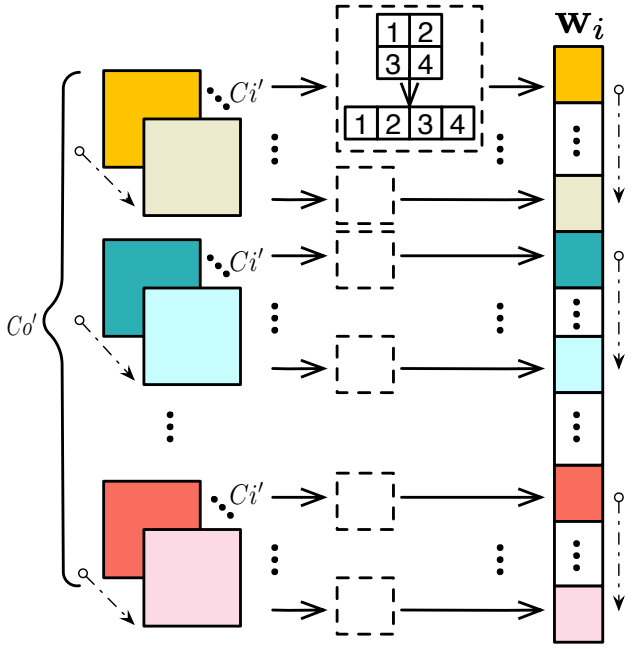


Figure 6.3 Reshape the parameter tensor of one group as a vector, with Ci' input channels and Co' output channels. The kernel size is 2×2. The arrows show the flattening order.

parameters in group i, termed as **intra-group correlation**. γ_i is a coefficient reflecting the relative importance of group i in comparison with other groups, termed as **inter-group importance**. γ_i also indicates the importance of the group i while passing messages or knowledges in the model during inference. $\boldsymbol{\mu}_{\boldsymbol{w}_i}$ is the mean vector of the network parameters $\boldsymbol{w}_i$ in the group i. And $\boldsymbol{\Sigma}_{\boldsymbol{w}_i}$ is the covariance matrix of the network parameters $\boldsymbol{w}_i$ in the group i. For the convolutional layer with g groups, the prior distribution of network parameters is

$$\mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{w}}, \boldsymbol{\Sigma}_{\boldsymbol{w}}), \tag{6.2}$$

where $\boldsymbol{\mu}_{\boldsymbol{w}} = [\boldsymbol{\mu}_{\boldsymbol{w}_1}^\top, \boldsymbol{\mu}_{\boldsymbol{w}_2}^\top, \cdots, \boldsymbol{\mu}_{\boldsymbol{w}_g}^\top]^\top$ is the mean vector of the network parameters $\boldsymbol{w}$. $\boldsymbol{\Sigma}_{\boldsymbol{w}} = \text{diag}[\boldsymbol{\Sigma}_{\boldsymbol{w}_1}, \boldsymbol{\Sigma}_{\boldsymbol{w}_2}, \cdots, \boldsymbol{\Sigma}_{\boldsymbol{w}_g}]$ is the covariance matrix of $\boldsymbol{w}$, which is a block diagonal matrix with principal diagonal blocks being $\boldsymbol{\Sigma}_{\boldsymbol{w}_1}, \boldsymbol{\Sigma}_{\boldsymbol{w}_2}, \cdots, \boldsymbol{\Sigma}_{\boldsymbol{w}_g}$. $\boldsymbol{B} \triangleq \{\boldsymbol{B}_1, \boldsymbol{B}_2, \cdots, \boldsymbol{B}_g\}$ and $\boldsymbol{\gamma} \triangleq [\gamma_1, \gamma_2, \cdots, \gamma_g]^\top$. The intra-group correlation $\boldsymbol{B}_i$ and the inter-group importance γ_i are determined by maximizing Type II likelihood [100] as shown in Formulation (6.3).

$$\max_{\boldsymbol{B},\gamma} \quad \ln \int \mathcal{P}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}}, \boldsymbol{w}) \mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma}) \mathrm{d}\boldsymbol{w}, \tag{6.3}$$

where $\boldsymbol{\mathcal{Y}}$ and $\boldsymbol{\mathcal{X}}$ are the output and input features, respectively. $\mathcal{P}(\boldsymbol{w}; \boldsymbol{B}, \boldsymbol{\gamma})$ satisfies multivariate Gaussian distribution with hyper-parameters $\boldsymbol{B}$ and $\boldsymbol{\gamma}$ defined in Equation (6.2).

According to Formulation (6.3), to obtain the intra-group correlation $\boldsymbol{B}_i$ and the inter-group importance γ_i, we need give a concrete form of $\mathcal{P}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}}, \boldsymbol{w})$. Moreover, the concrete form of $\mathcal{P}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}}, \boldsymbol{w})$ relies on the relationships among input features $\boldsymbol{\mathcal{X}}$, output features $\boldsymbol{\mathcal{Y}}$ and model parameters $\boldsymbol{w}$ in one grouped convolutional layer.

Nevertheless, in practice, because of non-linear operations in CNN models, it is hard to obtain the closed form of the likelihood function $\mathcal{P}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}}, \boldsymbol{w})$ and the integral of the marginal likelihood in Formulation (6.3) is intractable in neural networks [59]. Like in [58], we consider the linear relationship between the input and the output features of each layer before a nonlinearity is applied.

6.2.2 Maximum Type II Likelihood Estimation

In this subsection, the Type II likelihood in Equation (6.3) is transformed for the ease of the computations of the intra-group correlation $\boldsymbol{B}_i$ and inter-group importance γ_i.

As mentioned above, we consider the linear relationship between the input and the output features of each layer before a nonlinearity is applied. To represent the vanilla grouped convolution in the form of matrix-vector multiplication, we reshape the input features as shown in Fig. 6.4:

- In Step 1, we reshape the input features of one group to be a block-diagonal matrix. As the parameter kernel window slides on the input feature, the corresponding features are flattened to be a vector with length k^2. Therefore, we flatten the feature in one channel to be an $HW \times k^2$ matrix. Then the feature matrices of all Ci' channels are reshaped to be a block-diagonal matrix.

- In Step 2, we duplicate the input feature block matrix by Co' times to generate a larger block-diagonal matrix.

- In Step 3, we place the block-diagonal matrices of the g groups at the diagonal of the final feature matrix. The parameters are also reshaped in the same manner, as mentioned above in Fig. 6.3.

For each group, the matrix-vector multiplication with model error $\boldsymbol{v}_i$ can be represented as $\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{w}_i + \boldsymbol{v}_i$, where $\boldsymbol{y}_i \in \mathbb{R}^{MCo'}$

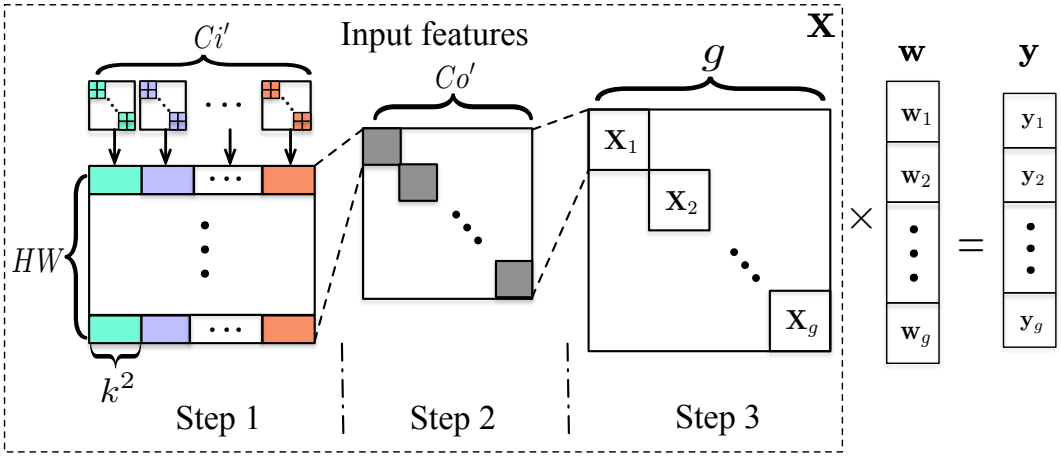


Figure 6.4 Reshape input features. The vanilla grouped convolution operation is transformed as matrix-vector multiplication.

and $\boldsymbol{X}_i \in \mathbb{R}^{MCo' \times NCo'}$ represent the reshaped outputs and inputs respectively, with $M \triangleq HW$. For a layer with g groups, the vanilla grouped convolution is:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{v}, \tag{6.4}$$

where $\boldsymbol{y} = [\boldsymbol{y}_1^\top, \cdots, \boldsymbol{y}_g^\top]^\top$, $\boldsymbol{X} = \text{diag}[\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_g]$, and $\boldsymbol{v} = [\boldsymbol{v}_1^\top, \cdots, \boldsymbol{v}_g^\top]^\top$. The model error $\boldsymbol{v}$ is assumed to follow independent identical Gaussian distribution, i.e., $\mathcal{P}(\boldsymbol{v}) \sim \mathcal{N}(\boldsymbol{0}, \lambda \boldsymbol{I})$, where λ is a hyper-parameter controling the precision of model error. $\boldsymbol{I}$ is an identity matrix. The concrete form of the likelihood function in Formulation (6.3) can be obtained as follows:

$$\mathcal{P}(\boldsymbol{\mathcal{Y}}|\boldsymbol{\mathcal{X}}, \boldsymbol{w}) = \mathcal{P}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}; \lambda) \sim \mathcal{N}(\boldsymbol{X}\boldsymbol{w}, \lambda \boldsymbol{I}). \tag{6.5}$$

According to Fig. 6.4, the size of matrix $\boldsymbol{X}$ is $MCo'g \times NCo'g$. However, on one hand, note that the vanilla grouped convolution adopts the sparse convolution connections between input

and output channels, by dividing the input channels, output channels, and their connections into several groups. Thus the matrix-vector multiplication can be performed group by group, i.e., $\boldsymbol{y}_i = \boldsymbol{X}_i\boldsymbol{w}_i + \boldsymbol{v}_i$, where $\boldsymbol{X}_i$'s size is $MCo' \times NCo'$. Moreover, $\boldsymbol{X}_i$ is also a block diagonal matrix, whose each block size is $M \times N$, which can facilitate the matrix-vector multiplication output channel by output channel. Besides, in popular vanilla grouped convolutional neural networks, for example, ResNeXt, small kernels (e.g., $k = 3$ or $k = 1$) are widely adopted. On the other hand, our proposed sharing framework is performed layer by layer in grouped convolutional layers, instead of all convolutional layers, in popular CNN models. Thus in practice, this reshaping input features does not bring a horrible memory footprint. In addition, this reshaping is only performed in the training stage while the memory footprint does not increase in the inference stage.

According to the network parameters prior $\mathcal{P}(\boldsymbol{w}_i; \gamma_i, \boldsymbol{B}_i)$ defined in Equation (6.1) and the likelihood function $\mathcal{P}(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$ defined in Equation (6.5), the posterior of network parameters also follows multivariate Gaussian distribution $\mathcal{P}(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}; \boldsymbol{\gamma}, \boldsymbol{B}, \lambda)$ $\sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where the mean $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$ are represented as follows [100]:

$$\begin{aligned} \boldsymbol{\mu} &= \boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}), \\ \boldsymbol{\Sigma} &= (\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1} + \frac{1}{\lambda}\boldsymbol{X}^\top\boldsymbol{X})^{-1}, \end{aligned} \tag{6.6}$$

where $\boldsymbol{\mu} \triangleq [\boldsymbol{\mu}_1^\top, \cdots, \boldsymbol{\mu}_g^\top]^\top$, and $\boldsymbol{\Sigma} \triangleq \text{diag}[\boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_g]$. $\boldsymbol{\mu}_i$ and

$\boldsymbol{\Sigma}_i$ are the posterior mean and the covariance matrix of network parameters in the group i, respectively.

Now to determine the intra-group correlation $\boldsymbol{B}_i$ and the inter-group importance γ_i, we can transform Formulation (6.3) as follows:

$$\max_{\boldsymbol{B},\boldsymbol{\gamma},\lambda} \quad \ln \mathcal{P}(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{B},\boldsymbol{\gamma},\lambda), \tag{6.7}$$

where the marginal likelihood function $\mathcal{P}(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{B},\boldsymbol{\gamma},\lambda)$ is defined as follows:

$$\mathcal{P}(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{B},\boldsymbol{\gamma},\lambda) = \int \mathcal{P}(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w};\lambda)\mathcal{P}(\boldsymbol{w};\boldsymbol{B},\boldsymbol{\gamma})\mathrm{d}\boldsymbol{w}. \tag{6.8}$$

Then Formulation (6.7) can be transformed to the equivalent formulation as follows:

$$\min_{\boldsymbol{B},\boldsymbol{\gamma},\lambda} \quad \mathcal{L}(\boldsymbol{B},\boldsymbol{\gamma},\lambda), \tag{6.9}$$

where

$$\begin{aligned} &\mathcal{L}(\boldsymbol{B},\boldsymbol{\gamma},\lambda) \triangleq -2\ln \mathcal{P}(\boldsymbol{y}|\boldsymbol{X};\boldsymbol{B},\boldsymbol{\gamma},\lambda) \\ &= \ln|\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top| \\ &+ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top(\lambda\boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}). \end{aligned} \tag{6.10}$$

Since it has the ability to adaptively learn and exploit intra-group correlation for better performance and only takes few iterations, in next section, we illustrate how to use a group LASSO type method to handle Formulation (6.9) so that the intra-group correlation $\boldsymbol{B}_i$, the inter-group importance γ_i and the hyper-parameter λ can be well determined.

6.2.3 Optimization via Group LASSO Type Algorithm

In this subsection, we follow the work [157] and use a group LASSO type algorithm to determine hyper-parameters so that it can achieve fast convergence. The main idea is shown as follows: Firstly, we find the upper-bound of the cost function $\mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$ defined in Equation (6.10). Then the upper-bound can be transformed to be a group LASSO problem. As a result, we can solve it with a typical group LASSO solver more efficiently.

In order to find an appropriate upper-bound of $\mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$, we introduce a temporary function $h(\boldsymbol{\alpha}) \triangleq [\frac{1}{\lambda}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \boldsymbol{\alpha}^\top \boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}]$. $\boldsymbol{\alpha}$ is defined as a temporary variable, which is different from the model parameters $\boldsymbol{w}$. Note that the function $h(\boldsymbol{\alpha})$ is convex. Therefore, there is a global minimum $\boldsymbol{\alpha}_0$, i.e., $h(\boldsymbol{\alpha}_0) \leq h(\boldsymbol{\alpha})$, with the first derivative $h(\boldsymbol{\alpha}_0)' = \boldsymbol{0}$, where

$$\boldsymbol{\alpha}_0 = (\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1} + \boldsymbol{X}^\top \boldsymbol{X})^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top \boldsymbol{X}. \tag{6.11}$$

Substituting Equation (6.11) into the function $h(\boldsymbol{\alpha})$ and using Woodbury matrix identity [96] lead to

$$h(\boldsymbol{\alpha}_0) = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top (\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}). \tag{6.12}$$

Thus for the right term in Equation (6.10), we have

$$\begin{aligned} &(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}})^\top (\lambda \boldsymbol{I} + \boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^\top)^{-1}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}) \\ \equiv\ &\min_{\boldsymbol{\alpha}} [\frac{1}{\lambda}||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \boldsymbol{\alpha}^\top \boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}]. \end{aligned} \tag{6.13}$$

With this, Equation (6.10) is upper-bounded by:

$$\begin{aligned}\mathcal{UL}(\boldsymbol{\alpha},\boldsymbol{\gamma},\boldsymbol{B},\lambda) &= \ln|\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^{\top}| \\ &\quad + \frac{1}{\lambda}||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}-\boldsymbol{X}\boldsymbol{\alpha}||_2^2+\boldsymbol{\alpha}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}. \end{aligned} \tag{6.14}$$

Here, we temporarily fix $\boldsymbol{B}$ and λ. Then instead of directly optimizing Formulation (6.9), we minimize the upper-bound in Equation (6.14) w.r.t. $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ as follows:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\gamma}} \quad \mathcal{UL}(\boldsymbol{\alpha},\boldsymbol{\gamma}). \tag{6.15}$$

Furthermore, considering the term $(1/\lambda)||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}-\boldsymbol{X}\boldsymbol{\alpha}||_2^2$ is independent of $\boldsymbol{\gamma}$ in Equation (6.14), Formulation (6.15) can be handled in two steps alternatively and iteratively. In the first step, we optimize Formulation (6.15) w.r.t. $\boldsymbol{\gamma}$ as follows:

$$f(\boldsymbol{\alpha}) \triangleq \min_{\boldsymbol{\gamma}\geq\boldsymbol{0}}[\ln|\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^{\top}|+\boldsymbol{\alpha}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha}], \tag{6.16}$$

In the second step, we optimize Formulation (6.15) w.r.t. $\boldsymbol{\alpha}$ as follows:

$$\min_{\boldsymbol{\alpha}} \quad ||\boldsymbol{y}-\boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}-\boldsymbol{X}\boldsymbol{\alpha}||_2^2+\lambda f(\boldsymbol{\alpha}). \tag{6.17}$$

In the first step, since $g(\boldsymbol{\gamma}) \triangleq \ln|\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^{\top}|$ is non-decreasing and concave, $g(\boldsymbol{\gamma})$ can be expressed with its concave conjugate $g^*(\boldsymbol{z})$ as follows [19]:

$$g(\boldsymbol{\gamma}) \triangleq \ln|\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^{\top}| = \min_{\boldsymbol{z}\geq\boldsymbol{0}} \boldsymbol{z}^{\top}\boldsymbol{\gamma}-g^*(\boldsymbol{z}), \tag{6.18}$$

where the concave conjugate is given as follows:

$$g^*(\boldsymbol{z}) = \min_{\boldsymbol{\gamma}\geq\boldsymbol{0}} \boldsymbol{z}^{\top}\boldsymbol{\gamma}-\ln|\lambda\boldsymbol{I}+\boldsymbol{X}\boldsymbol{\Sigma}_{\boldsymbol{w}}\boldsymbol{X}^{\top}|. \tag{6.19}$$

Therefore, Equation (6.16) can be transformed as Equation (6.20).

$$\begin{aligned} f(\boldsymbol{\alpha}) &= \min_{\boldsymbol{\gamma},\boldsymbol{z}\geq\mathbf{0}} \quad \boldsymbol{\alpha}^\top \boldsymbol{\Sigma}_{\boldsymbol{w}}^{-1}\boldsymbol{\alpha} + \boldsymbol{z}^\top\boldsymbol{\gamma} - g^*(\boldsymbol{z}) \\ &= \min_{\boldsymbol{\gamma},\boldsymbol{z}\geq\mathbf{0}} \quad \sum_{i=0}^{g}\left(\frac{\boldsymbol{\alpha}_i^\top \boldsymbol{B}_i^{-1}\boldsymbol{\alpha}_i}{\gamma_i} + z_i\gamma_i\right) - g^*(\boldsymbol{z}), \end{aligned} \tag{6.20}$$

where $\boldsymbol{z} = [z_1, z_2, \cdots, z_g]^\top$. Minimizing Equation (6.20) w.r.t. $\boldsymbol{\gamma}$, we have

$$\gamma_i = z_i^{-\frac{1}{2}}\sqrt{\boldsymbol{\alpha}_i^\top \boldsymbol{B}_i^{-1}\boldsymbol{\alpha}_i}, \quad i = 1, 2, \cdots, g. \tag{6.21}$$

However, γ_i relies on z_i. According to Equation (6.18) and the duality property [19], we can obtain

$$z_i = \mathrm{Tr}[\boldsymbol{B}_i\boldsymbol{X}_i^\top(\lambda\boldsymbol{I} + \boldsymbol{X}_i\boldsymbol{\Sigma}_{\boldsymbol{w}_i}\boldsymbol{X}_i^\top)^{-1}\boldsymbol{X}_i]. \tag{6.22}$$

According to Equation (6.21) and Equation (6.22), $\boldsymbol{\gamma}$ relies on $\boldsymbol{z}$ and $\boldsymbol{z}$ relies on $\boldsymbol{\gamma}$ ($\boldsymbol{\Sigma}_{\boldsymbol{w}}$). Therefore, in the first step, we optimize Formulation (6.16) by updating $\boldsymbol{\gamma}$ and $\boldsymbol{z}$, alternatively.

In the second step, after $\boldsymbol{\gamma}$ and $\boldsymbol{z}$ are determined, Formulation (6.17) is transformed as follows:

$$\min_{\boldsymbol{\alpha}} \quad ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}} - \boldsymbol{X}\boldsymbol{\alpha}||_2^2 + \lambda\sum_{i=1}^{g} 2z_i^{\frac{1}{2}}\sqrt{\boldsymbol{\alpha}_i^\top \boldsymbol{B}_i^{-1}\boldsymbol{\alpha}_i}. \tag{6.23}$$

Formulation (6.23) is an implicit group LASSO formulation. To make it more clear, we transform it to be Formulation (6.24) as follows:

$$\min_{\boldsymbol{d}} \quad ||\boldsymbol{t} - \boldsymbol{H}\boldsymbol{d}||_2^2 + \lambda\sum_{i=1}^{g}||\boldsymbol{d}_i||_2, \tag{6.24}$$

where $\boldsymbol{d}_i = 2z_i^{1/2}\boldsymbol{B}_i^{-1/2}\boldsymbol{\alpha}_i$, $\boldsymbol{d} = [\boldsymbol{d}_1^\top, \boldsymbol{d}_2^\top, \cdots, \boldsymbol{d}_g^\top]^\top$, $\boldsymbol{t} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}_{\boldsymbol{w}}$, $\boldsymbol{H} = \boldsymbol{X}\cdot\text{diag}[\boldsymbol{B}_1^{1/2}/(2z_1^{1/2}), \boldsymbol{B}_2^{1/2}/(2z_2^{1/2}), \cdots, \boldsymbol{B}_g^{1/2}/(2z_g^{1/2})]$. Formulation (6.24) is a standard group LASSO formulation, which can handled by calling classical group LASSO solver (e.g., [3]) to determine $\boldsymbol{\alpha}$.

Note that during the above process, we fix the intra-group correlation $\boldsymbol{B}$ and the hyper-parameter λ. In fact, the hyper-parameter λ can be automatically determined by a group LASSO solver [3]. Besides, according to [157], since $\boldsymbol{\alpha}$ has the approximate covariance with $\boldsymbol{w}$, $\boldsymbol{B}$ can be approximately estimated by $\boldsymbol{\alpha}$ from the previous iteration, that is

$$\boldsymbol{B}_i = \frac{1}{\gamma_i}\boldsymbol{\Sigma}_{\boldsymbol{w}_i} \approx \frac{1}{\gamma_i}\mathbb{E}[(\boldsymbol{\alpha}_i - \mathbb{E}(\boldsymbol{\alpha}_i))(\boldsymbol{\alpha}_i - \mathbb{E}(\boldsymbol{\alpha}_i))^\top]. \tag{6.25}$$

In particular, according to [50], the first-order auto-regressive process corresponding to the Toeplitz matrix is more sufficient to capture intra-group correlation. Therefore, the intra-group correlation matrix $\boldsymbol{B}_i$ is replaced by $\hat{\boldsymbol{B}}_i$ as follows:

$$\hat{\boldsymbol{B}}_i = \text{Toeplitz}([1, \text{r}, \cdots, \text{r}^{\text{NCo}'-1}]), \tag{6.26}$$

where $r = \bar{m}_1/\bar{m}_0$, $\bar{m}_0$ and $\bar{m}_1$ are the averages of elements along the main diagonal and the main sub-diagonal of $\boldsymbol{B}_i$, respectively.

In summary, the developed group LASSO type algorithm flow is shown in Algorithm 7. We do not directly optimize Formulation (6.9). Instead, we find the upper-bound of the function $\mathcal{L}(\boldsymbol{B}, \boldsymbol{\gamma}, \lambda)$ as shown in Equation (6.14). Then the upper-bound function $\mathcal{UL}(\boldsymbol{\alpha}, \boldsymbol{\gamma}, \boldsymbol{B}, \lambda)$ is minimized by two steps. In the first

step, we alternatively optimize Formulation (6.16) to obtain the hyper-parameter $\boldsymbol{\gamma}$ in Equation (6.21) and $\boldsymbol{z}$ in Equation (6.22). In the second step, we transform Formulation (6.17) as an equivalent group LASSO formulation as shown in Formulation (6.24). After calling the group LASSO solver, we can determine $\boldsymbol{d}$ ($\boldsymbol{\alpha}$) and λ, simultaneously. In addition, we use Equations (6.25) and (6.26) to update the hyper-parameters $\boldsymbol{B}$ and $\hat{\boldsymbol{B}}_i$. The above process is iteratively performed until convergence. Then the intra-group correlation $\boldsymbol{B}$, the inter-group importance $\boldsymbol{\gamma}$ and the hyper-parameter λ are determined. In practice, it only takes few iterations in Algorithm 7 (2 to 5 iterations). In each iteration, any efficient group LASSO algorithm can be used, which bring much faster and suitable to the sharing parameters in the grouped convolution.

Algorithm 7 Group LASSO to handle Formulation (6.9).

Require: $\boldsymbol{X}$, $\boldsymbol{y}$ from one grouped convolutional layer, network parameters $\boldsymbol{w}$.

1: Initialize $\boldsymbol{B}$, $\boldsymbol{\gamma}$, $\boldsymbol{z}$ and λ.
2: **repeat**
3: Update $\boldsymbol{\gamma}$ by Equation (6.21);
4: Update $\boldsymbol{z}$ by Equation (6.22);
5: Solve Formulation (6.24) (Formulation (6.23)) to obtain $\boldsymbol{d}$ ($\boldsymbol{\alpha}$) and λ;
6: Update $\boldsymbol{B}_i = \hat{\boldsymbol{B}}_i$ by Equations (6.25) and (6.26);
7: **until** Convergence
8: **return** hyper-parameters $\boldsymbol{B}$, $\boldsymbol{\gamma}$ and λ.

6.2.4 Overall flow

In this subsection, we will give an overall flow about how to share parameters among different groups so that the vanilla grouped convolution can be transferred as the sharing structure.

After $\boldsymbol{B}$, $\boldsymbol{\gamma}$ and λ are determined by Algorithm 7, in each group, model parameters $\boldsymbol{w}_i$ can be determined by the posterior mean as shown in Equation (6.6), that is $\boldsymbol{w}_i = \boldsymbol{\mu}_i$. To share the parameters among different groups in one grouped convolutional layer, the mean of the sharing parameters $\boldsymbol{\mu}_{\boldsymbol{w}_b}$ is defined as a prior mean as follows:

$$\boldsymbol{\mu}_{\boldsymbol{w}_b} = \frac{\sum_i^g \gamma_i \boldsymbol{w}_i}{\sum_i^g \gamma_i}. \tag{6.27}$$

The mean is the weighted average of all network parameters obtained in the last iteration, with the inter-group importance γ_i. Then in Equations (6.10) and (6.14), the prior mean is $\boldsymbol{\mu}_{\boldsymbol{w}} = \mathbf{1}_g \otimes \boldsymbol{\mu}_{\boldsymbol{w}_b} = [\boldsymbol{\mu}_{\boldsymbol{w}_b}^\top, \boldsymbol{\mu}_{\boldsymbol{w}_b}^\top, \cdots, \boldsymbol{\mu}_{\boldsymbol{w}_b}^\top]^\top$, and $\mathbf{1}_g \in \mathbb{R}^g$ is a vector whose all elements are 1. $\otimes$ represents the Kronecker product. The sharing process is shown in Algorithm 8. As shown in Fig. 6.5, initially, all groups have different parameters. After few iterations, parameters will gradually become the same by our proposed Bayesian sharing framework. In particular, the mean sharing method is a special case of our proposed Bayesian sharing method, i.e., $\boldsymbol{\gamma} \equiv \mathbf{1}_g$.

For the whole CNN model, we adopt a separate-merge methodology [44] to share weights in all grouped convolutional layers, that is separately updating parameters by loss function in the

Algorithm 8 Bayesian sharing framework

Require: $\boldsymbol{X}$, $\boldsymbol{y}$ from one grouped convolutional layer, network parameters $\boldsymbol{w}$.

1: Initialize $\boldsymbol{\mu}_{\boldsymbol{w}_b} = \sum_i^g \boldsymbol{w}_i/g$, $\boldsymbol{\mu}_{\boldsymbol{w}} = \mathbf{1}_g \otimes \boldsymbol{\mu}_{\boldsymbol{w}_b}$;
2: **repeat**
3: Update $\boldsymbol{B}$, $\boldsymbol{\gamma}$ and λ by Algorithm 7;
4: Update model parameters $\boldsymbol{w}_i$ by the posterior mean in Equation (6.6);
5: Update the sharing model parameters $\boldsymbol{\mu}_{\boldsymbol{w}_b}$ by Equation (6.27) and $\boldsymbol{\mu}_{\boldsymbol{w}} = \mathbf{1}_g \otimes \boldsymbol{\mu}_{\boldsymbol{w}_b}$;
6: **until** Convergence
7: **return** The sharing weights $\boldsymbol{w}_b = \boldsymbol{\mu}_{\boldsymbol{w}_b}$.

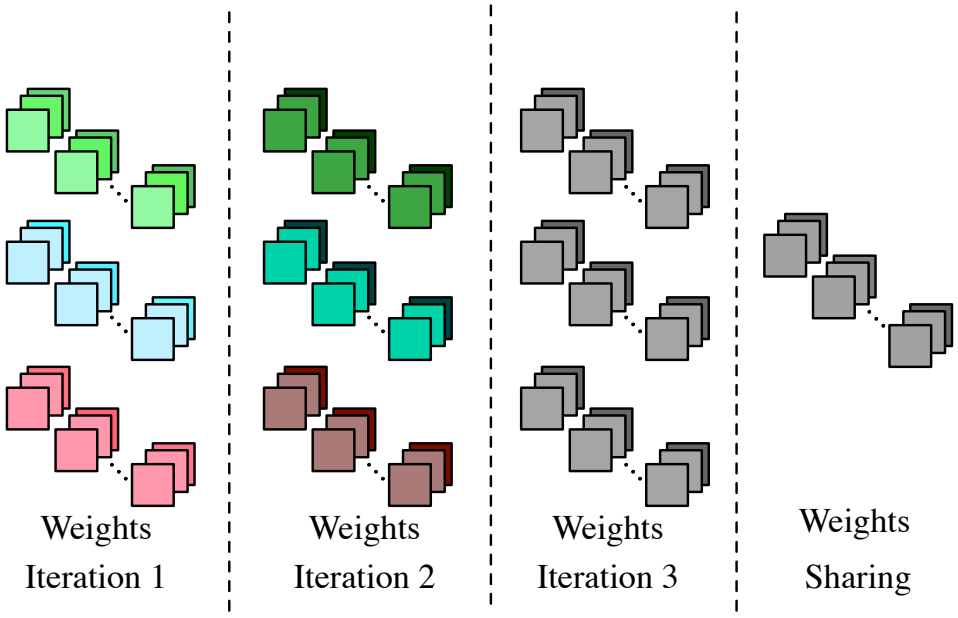


Figure 6.5 The sharing process of grouped convolution parameters. Green, blue and red boxes represent parameters (kernels) in three groups. After few iterations, all groups have the same kernels (grey boxes), which are shared.

back-propagation stage and computing loss function value in the forward propagation stage. Given a pre-trained CNN model, we fix model parameters in non-grouped convolutional layers and

update model parameters in all grouped convolutional layers by our proposed Bayesian sharing method as shown in Algorithm 8 from front layers to back layers sequentially in the forward propagation stage.

The loss value is calculated by all updated shared grouped convolution parameters and other fixed model parameters. Then the loss value is used to updated all model parameters. By performing this sharing process for few epochs, the final sharing model can be obtained.

Note that instead of all layers in CNN model, our proposed Bayesian sharing framework is performed in grouped convolutional layers to transfer to be the sharing structure.

It is worth mentioning that our proposed Bayesian sharing framework is not only compressing models but also regularizing parameters. The regularization technique can encourage learning a more simple model to avoid the risk of overfitting so that the accuracy can be improved [100]. The most common regularization techniques are Ride Regression and Lasso regression, which can force the model parameters to decay towards zeros so that model complexity is reduced and the model generalizes better [100]. Note that directly learning a sharing grouped convolution structure has the same manner with directly forcing the model parameters to zeros, which brings performance degradation. In the proposed Bayesian sharing framework, the model parameters are imposed to be the same among different groups by adaptively learning the intra-group correlation $\boldsymbol{B}$ and

Table 6.3 ResNeXt on CIFAR Dataset.

Model	g	RexNeXt-35					RexNeXt-50					GCR
		#P	Acc. (%)		FLOPs	Time	#P	Acc. (%)		FLOPs	Time	
		(M)	C-10	C-100	(M)	(ms)	(M)	C-10	C-100	(M)	(ms)	(%)
ResNeXt (baseline)	4	1.29	92.87	72.91	202	33.4	2.01	93.67	73.08	279	42.8	100.00
ResNeXt-D	4	1.19 (↓)	92.02 (↓)	72.17 (↓)	202	26.2	1.58 (↓)	92.89 (↓)	72.86 (↓)	279	33.3	25.00
ResNeXt-M	4	1.19 (↓)	93.31 (↑)	73.44 (↑)	202	26.2	1.58 (↓)	93.90 (↑)	73.55 (↑)	279	33.3	25.00
ResNeXt-B	4	**1.19** (↓)	**94.15** (↑)	**74.56** (↑)	**202**	**26.2**	**1.58** (↓)	**94.93** (↑)	**75.46** (↑)	**279**	**33.3**	**25.00**
ResNeXt (baseline)	8	1.31	93.00	73.07	214	43.3	2.04	93.22	73.16	291	47.3	100.00
ResNeXt-D	8	1.18 (↓)	92.32 (↓)	72.51 (↓)	214	38.5	1.70 (↓)	93.14 (↓)	72.71 (↓)	291	42.3	25.00
ResNeXt-M	8	1.18 (↓)	93.42 (↑)	73.62 (↑)	214	38.5	1.70 (↓)	93.78 (↑)	73.93 (↑)	291	42.3	12.50
ResNeXt-B	8	**1.18** (↓)	**94.08** (↑)	**74.97** (↑)	**214**	**38.5**	**1.70** (↓)	**95.04** (↑)	**76.11** (↑)	**291**	**42.3**	**12.50**
ResNeXt (baseline)	16	1.35	92.93 (↑)	73.23 (↑)	222	48.7	2.12	93.23 (↑)	73.31 (↑)	309	55.0	100.00
ResNeXt-D	16	1.26 (↓)	92.48 (↓)	72.57 (↓)	222	43.8	1.88 (↓)	93.22 (↓)	72.93 (↓)	309	52.6	6.25
ResNeXt-M	16	1.26 (↓)	93.38 (↑)	73.64 (↑)	222	43.8	1.88 (↓)	93.78 (↑)	73.79 (↑)	309	52.6	6.25
ResNeXt-B	16	**1.26** (↓)	**94.77** (↑)	**75.88** (↑)	**222**	**43.8**	**1.88** (↓)	**95.17** (↑)	**75.87** (↑)	**309**	**52.6**	**6.25**

the inter-group importance $\boldsymbol{\gamma}$. Thus the intuition behind this technique has the same manner as the common regularizations, which adaptively force the model parameters to decay towards zeros. As a result, we expect the proposed Bayesian sharing framework can improve the accuracy of original models.

6.3 Experimental Results

In this section, we apply our Bayesian sharing framework on some popular grouped convolutional neural networks, including ResNeXt [134], ShuffleNet [152] and G-DenseNet [48, 49]. We test them on CIFAR-10, CIFAR-100 [9], and ImageNet [5]. As an ablation study, to clarify the impact of the proposed Bayesian sharing framework, the directly trained sharing grouped convolutional neural networks and the mean sharing method are also implemented for comparison. The direct training method

Table 6.4 ShuffleNet and G-DenseNet on CIFAR Dataset.

Model	g	#P (M)	Acc. (%) C-10	Acc. (%) C-100	FLOPs (M)	Time (ms)	GCR (%)
ShuffleNet-1x (baseline)	4	0.62	91.65	71.48	106	23.0	100.00
ShuffleNet-1x-D	4	0.28 (↓)	90.78 (↓)	70.56 (↓)	106	17.6	25.00
ShuffleNet-1x-M	4	0.28 (↓)	92.47 (↑)	72.29 (↑)	106	17.6	25.00
ShuffleNet-1x-B	4	**0.28** (↓)	**93.56** (↑)	**73.83** (↑)	**106**	**17.6**	**25.00**
ShuffleNet-2x (baseline)	4	1.34	91.48	71.65	123	29.8	100.00
ShuffleNet-2x-D	4	0.48 (↓)	90.32 (↓)	70.49 (↓)	123	22.9	25.00
ShuffleNet-2x-M	4	0.48 (↓)	92.68 (↑)	72.07 (↑)	123	22.9	25.00
ShuffleNet-2x-B	4	**0.48** (↓)	**93.79** (↑)	**73.89** (↑)	**123**	**22.9**	**25.00**
ShuffleNet-1x (baseline)	8	1.35	92.29	72.12	204	33.4	100.00
ShuffleNet-1x-D	8	0.60 (↓)	91.19 (↓)	71.57 (↓)	204	28.1	12.50
ShuffleNet-1x-M	8	0.60 (↓)	93.16 (↑)	72.26 (↑)	204	28.1	12.50
ShuffleNet-1x-B	8	**0.60** (↓)	**94.00** (↑)	**72.98** (↑)	**204**	**28.1**	**12.50**
G-DenseNet-86 (baseline)	4	0.62	93.21	73.89	102	69	100.00
G-DenseNet-86-D	4	0.33 (↓)	92.89 (↓)	73.14 (↓)	102	53	25.00
G-DenseNet-86-M	4	0.33 (↓)	93.78 (↑)	73.76 (↓)	102	53	25.00
G-DenseNet-86-B	4	**0.33** (↓)	**94.91** (↑)	**75.12** (↑)	**102**	**53**	**25.00**

constructs a network with the proposed sharing structure and then trains it. The mean sharing method trains the model from scratch and each group has its own weights. At some certain training epochs, e.g., 80 and 100 epochs, we average the weights and then continue the training process. In the experimental results, "$-D$" represents the results of directly trained sharing grouped convolutional neural networks, "$-M$" represents the results of mean sharing, and "$-B$" represents the Bayesian sharing.

6.3.1 Implementation details and experimental settings

Training settings

On CIFAR-10 and CIFAR-100, we test all of these three methods. The initial learning rate is set as 0.1. For ResNeXt and ShuffleNet, the batch size is 128 and the learning rate is gradually divided by 10 at 81 and 122 epochs, with 164 training epochs in total. For G-DenseNet, the batch size is 64, and the learning rate is divided by 10 at 150 and 225 epochs, with a total of 300 training epochs. CIFAR-10 and CIFAR-100 are shorted as C-10 and C-100 in the result tables.

We test the sharing ResNeXt On ImageNet. The learning rate is initially set to 0.1, divided by 10 at 50 and 70 epochs. There are 90 training epochs, and the batch size is 256.

Our optimizer uses momentum optimizer, with momentum 0.9 and weight decay 2×10^{-4}.

Evaluation Metrics

Parameter volume, model accuracy, and grouped convolution compression ratio (GCR) are considered as the evaluation metrics. Parameter volume, abbreviated as "#P", counts all the parameters in the model, including grouped convolutional layers and other linear or nonlinear layers. GCR is only for grouped convolutional layers, i.e., volume of the sharing layer divided by the original volume before sharing. The compression ratio of the baseline model is also 100%. For a grouped convolutional

Table 6.5 Our proposed sharing ResNeXt-50 and ShuffleNet, in comparison with the state-of-the-art models on CIFAR.

Model	Method	#P (M)	Acc. (%)	FLOPs (M)	Dataset
ResNet-50	CP [47]	6.44	94.15	1620	C-10
	CaP [90]	1.60	92.67	-	
	Genetic [56]	7.71	93.60	954	
	Gaussian [56]	5.94	94.00	735	
ResNeXt-50	-B (g=4)	**1.58**	94.93	**279**	
	-B (g=8)	1.70	95.04	291	
	-B (g=16)	1.88	**95.17**	309	
ResNet-50	FP [69]	7.83	73.60	616	C-100
	CP [47]	9.24	74.10	1740	
	PCAS [138]	4.02	73.84	475	
ResNeXt-50	-B (g=4)	**1.58**	75.46	**279**	
	-B (g=8)	1.70	**76.11**	291	
	-B (g=16)	1.88	75.87	309	
MobileNet-v2	α-1 [66]	2.30	90.20	89	C-10
	α-0.75 [66]	1.73	87.80	**60**	
	FLGC(g=2) [123]	1.18	94.11	158	
	FLGC(g=3) [123]	0.85	**94.20**	122	
	FLGC(g=4) [123]	0.68	94.16	103	
	FLGC(g=8) [123]	0.43	93.09	76	
ShuffleNet	1x-B (g=4)	**0.28**	93.56	106	
	2x-B (g=4)	0.48	93.79	123	
	1x-B (g=8)	0.60	94.00	204	

layer with g groups, after sharing, the compression ratio is $1/g$. Therefore, our compression ratio relies on the number of groups. For ImageNet, we report Top-1 and Top-5 accuracies. The number of floating point operations (FLOPs) and runtime are also attached.

6.3.2 Experiments on CIFAR Dataset

Our sharing method is applied to some baseline models, i.e., ResNeXt, ShuffleNet and G-DenseNet to test CIFAR-10 and CIFAR-100, with some necessary model modifications in Tables 6.3 and 6.4. For ResNeXt-35 and RexNeXt-50, to test the cardinality, some tests are conducted on grouped convolutional layers with 4, 8, 16 groups, while the kernel size is 3×3. The point-wise convolutional layers are not considered here since they are not in the grouped convolutional layers of these two models. For ShuffleNet, grouped convolutional layers with 4 and 8 groups are tested. Different from ResNeXt, the point-wise (1×1) convolutions in ShuffleNet are grouped convolutional layers. Some experiments are conducted on ShuffleNet with 1×1 convolutions to further demonstrate the effectiveness of our sharing method. DenseNet contains both 3×3 and 1×1 convolutional layers, which are both tested to further validate the compatibility of our method.

As ablation studies, to clarify the impacts of our proposed Bayesian sharing framework, we compare the directly trained model with sharing grouped convolution, the mean sharing, and the proposed Bayesian sharing. The results are shown in Tables 6.3 and 6.4. For all of these tests, compared with the corresponding baseline models, the performance degradations occur in all directly trained models. The mean sharing method can achieve slight accuracy improvements in the most cases but

G-DenseNet-86 since it is able to combine parameters among different groups without discrimination, i.e., $\boldsymbol{\gamma} \equiv \mathbf{1}_g$ in Equation (6.27). Compared with the mean sharing method, our Bayesian sharing framework can bring significant accuracy improvements, mostly more than 2%, since it considers the intra-group correlation and the inter-group importance to combine parameters among different groups with discriminations. In other words, it is able to discriminately combine parameters to achieve message passing to different features according to the importances learned from maximum likelihood estimation in Equation (6.9). Some tests achieve higher improvements, e.g., in TABLE 6.3, ResNeXt-50-B with 8 groups on CIFAR-100 improves the accuracy by $76.11\% - 73.16\% = 2.95\%$ with the less parameter volume. As a result, the proposed Bayesian sharing framework can improve the parameter efficiency, reduce the parameter redundancy and alleviate the overfitting issue.

Our Bayesian sharing method can result in impressive compression and runtime performance. Since for grouped convolutional layers with g groups, the GCR is $1/g$, more groups mean better compression ratio. According to Tables 6.3 and 6.4, as the group number increases, our method achieves higher compression ratios. Convolutional layers with 4 groups have the minimal GCR, i.e., compressed to 0.25 times. Dividing to 16 groups can bring the maximal compression ratio, i.e., 0.0625 times. Except for grouped convolutional layers, a typical neural network contains many other linear or non-linear layers. The models with

more grouped convolutional layers have better compression performance for parameter volume by using our Bayesian sharing method. In TABLE 6.3, for ResNeXt models with the limited number of 3×3 convolutional layers, we can achieve up to 21% $((2.01-1.58)/2.01)$ overall volume reduction. In TABLE 6.4, For the sharing G-DenseNet-86, the parameter volume is reduced by 46.77% $((0.62 - 0.33)/0.62)$. The sharing ShuffleNet-1x reduces the parameter volume by 54.8% $((0.62 - 0.28)/0.62)$, and the parameter volume in ShuffleNet-2x reduces more than 64.17% $((1.34 - 0.48)/1.34)$. The proposed sharing method can achieve the more significant parameter reductions for CNN with the more grouped convolutional layers. Generally, the deeper and larger models suffer from higher risks of overfitting. With our Bayesian sharing framework, we can alleviate this problem by reducing parameter volume.

In particular, compared with these baseline methods, our proposed sharing grouped convolution does not reduce FLOPs in the inference stage. However, as shown in Fig. 6.2, the parameters are shared among different groups. The sharing parameter strategy can reduce the actual number of memory accesses so that the inference time can be reduced, as shown in Tables 6.3 and 6.4. The runtime results are tested on one Kaggle Nvida Tesla P100 (16 GB memory, 720 GB/s bandwidth). It is believed that we can achieve better run performances on FPGA with dataflow optimizations [127].

We also compare our method with the state-of-the-art meth-

ods on ResNet, MobileNet and DenseNet on CIFAR dataset, as shown in Tables 6.5 and 6.6. These methods include efficient model architecture methods [56, 66–68, 93] and compression methods at various levels, such as filter pruning [69] and channel pruning [47, 80, 90, 123, 138, 158]. It is worth mentioning that the group pruning [123, 158] is a special channel pruning since the input channels, output channels, and their connections are divided into several groups.

According to Tables 6.5 and 6.6, our method outperforms all of the current efficient model architecture methods [56, 66–68, 93] since they do not consider correlations among parameters in the training stage so that the model performance is significantly degraded. In particular, it is hard to make a better trade-off between accuracy and parameter volume, even through some advanced neural architecture search methods are adopted to determine model configurations [56]. Unlike these traditional compression methods [47, 69, 80, 90, 123, 138, 158], our proposed Bayesian sharing framework does not completely prune channels or filters (kernels). Instead, compared with the grouped convolution, we reserve all filters (kernels) and channels, and discriminately combine parameters to achieve message passing to features with different importances. The reused weights, structures of model parameters and the adaptive importance learning strategy can reduce the parameter redundancy, improve efficiency and alleviate the overfitting issue. In particular, compared with the state-of-the-art group pruning methods which prune several

unimportant groups [123, 158], our method reserves all groups thus striking a better balance between accuracy and parameter volume.

6.3.3 Experiments on ImageNet

To further evaluate the impact of our Bayesian sharing framework for model performance on large data sets, we test the sharing ResNeXt-50 on ImageNet dataset. We follow the configurations in [134]. The number of groups is 32.

We compare our method with the state-of-the-art efficient model architecture methods [54,83,106,133,148,155] to examine model accuracy, parameter volume and FLOPs in TABLE 6.7. It is shown that our proposed Bayesian sharing framework beats these state-of-the-art efficient model architecture methods in terms of the model accuracy and the parameter volume. In particular, some advanced normalization layers are developed to enhance the generalization ability of the model but they make model cumbersome while bringing more FLOPs [54,83,106,133, 148]. Besides, compared with the dynamic grouping convolution [155], the sharing grouped convolution can reduce parameter volume since model parameters are shared among different groups. For the sharing ResNeXt-50-B on ImageNet, we can reduce the parameters in group convolutional layers by 96.875%, i.e., $100\% - 3.125\% = 96.875\%$. The reuse model parameters structure with the adaptive importance learning strategy can

Table 6.6 Our sharing G-DenseNet, in comparison with DenseNet on CIFAR.

Model	#P	Acc. (%)		FLOPs
	(M)	C-10	C-100	(M)
DenseNet-40-pruned [80]	0.66	94.81	74.72	190
DenseDsc(k=36) [68]	0.47	94.05	74.24	123
DVN-77 [93]	0.40	93.09	72.60	-
DenseNet-100-D [67]	0.70	93.12	72.39	269
DenseNet-86 (g=4) [158]	0.59	94.06	74.04	132.7
G-DenseNet-86-B (g=4)	**0.33**	**94.91**	**75.12**	**102**

Table 6.7 Our proposed sharing ResNeXt-50 in comparison with the state-of-the-art models on ImageNet Dataset.

Model	#P	Acc. (%)		FLOPs
	(M)	Top-1	Top-5	(M)
ResNet-50-BN [54]	25.56	77.60	93.70	4151
ResNet-50-GN [133]	25.56	76.00	92.80	4155
ResNet-50-SN [83]	25.56	76.90	93.20	4225
ResNet-50-SSN [106]	25.56	77.20	93.10	4186
ResNet-50-EN [148]	25.91	78.10	93.60	4325
G-ResNeXt-50 [155]	25.00	78.40	94.00	**4090**
ResNeXt-50-B (g=32)	**23.63**	**78.86**	**94.54**	4200

improve model accuracy to 78.86% and 94.54% for top-1 and top-5.

6.3.4 Model Compatibility

The experimental results also verify that our method has strong compatibility while tackling various grouped convolutional mod-

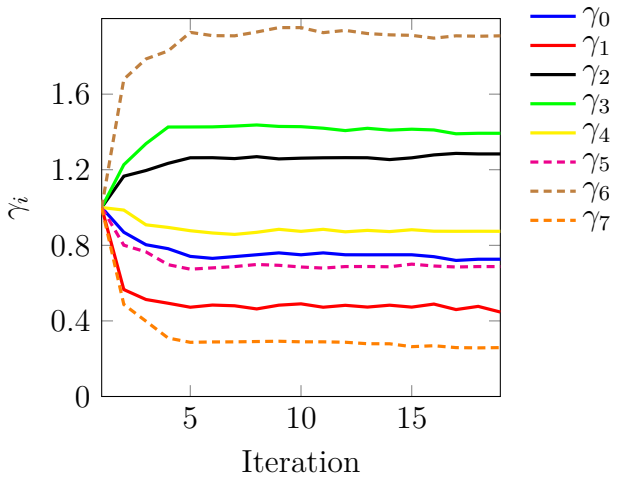


Figure 6.6 Optimization Iteration vs. $\boldsymbol{\gamma}$ values of ResNeXt-50 ($g = 8$).

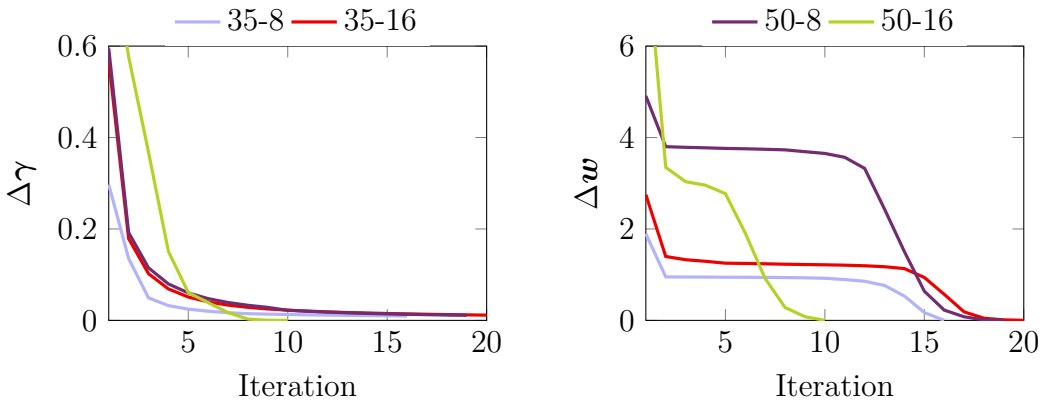


Figure 6.7 Optimization Iteration vs. $\Delta\boldsymbol{\gamma}$ and $\Delta\boldsymbol{w}$. Four models are listed here, i.e. ResNeXt-35 ($g = 8$, $g = 16$) and ResNeXt-50 ($g = 8$, $g = 16$).

els, from 3×3 convolutions to 1×1 point-wise convolutions in ResNeXt, ShuffleNet and G-DenseNet, for different numbers of groups (different cardinalities), on different datasets as shown in Tables 6.3 and 6.4. Overall, our experiments listed above have covered all of these diverse grouped convolution structures.

6.3.5 Inter-Group Importance

Inter-group importance is proposed in our framework. γ_i reflects the inter-group importance of group i. In our model, the stable values of all the γ_i are mutually distinct. Fig. 6.6 is taken as an example to illustrate this. For a layer in ResNeXt-50 with 8 groups, each γ_i is initialized as 1 before training. In other words, in the beginning, these groups are equally important. After about 5 optimization iterations, these γ reach different stable status. The group with higher γ_i is more important. If we use mean sharing, the γ_i can always be regarded as 1, i.e., $\boldsymbol{\gamma} \equiv \mathbf{1}_g$ in Equation (6.27). In comparison, our Bayesian sharing framework can better characterize the differences of inter-group importance, reasonably and effectively.

6.3.6 Convergence

To verify the optimization and convergence of our method more clearly, in ResNeXt-35 ($g = 8$, $g = 16$) and ResNeXt-50 ($g = 8$, $g = 16$), one layer is sampled from each model and its $\Delta\boldsymbol{\gamma}$ and $\Delta\boldsymbol{w}$ are shown in Fig. 6.7(a) and Fig. 6.7(b), respectively. Each layer has several groups, and $\Delta\boldsymbol{\gamma}$ is computed according to Equation (6.28).

$$\Delta\boldsymbol{\gamma} = \sum_{i=1}^{g} \left| \frac{\gamma_i^{(t+1)} - \gamma_i^{(t)}}{\gamma_i^{(t+1)}} \right|. \tag{6.28}$$

$\Delta\boldsymbol{w}$ is for $\boldsymbol{w}$ and is the summation of all the element-wise changes between two continuous training steps. Obviously, by

using Algorithm 8, parameter $\boldsymbol{w}$ converges quickly, and γ for each group also reaches a stable status quickly. Another trend is that the models with more parameters (e.g., ResNeXt-50 with $g = 16$) can have a faster convergence rate.

6.4 Summary

In this chapter, we propose a sharing grouped convolution structure with the Bayesian sharing framework to efficiently eliminate parameter redundancy and boost model performance. Intra-group correlation and inter-group importance are introduced into the prior of the parameters. We handle the Maximum Type II likelihood estimation problem of the intra-group correlation and inter-group importance by a group LASSO type algorithm. Experiments demonstrate the proposed sharing grouped convolution structure with the Bayesian sharing framework can reduce parameters and improve prediction accuracy. The proposed sharing framework can reduce parameters up to 64.17%. For ResNeXt-50 with the sharing grouped convolution on ImageNet dataset, network parameters can be reduced by 96.875% in grouped convolutional layers, and accuracies are improved to 78.86% and 94.54% for top-1 and top-5.

□ **End of chapter.**

Chapter 7

Conclusion

In this thesis, we have proposed, customized and developed a few methodologies to achieve hardware reliability and efficiency in design, verification and application stages in the advanced sensor system. In this chapter, we conclude each methodology and then give a comprehensive achievement for the development of the advanced sensor system.

On one hand, sensor calibration can be used to extend sensor lifetime even if the sensor was designed without considering the reliability. Instead of replacing aged sensors, we have developed a spatial correlation model to calibrate sensor measurements. Thus, maintenance costs can be reduced in the deployed intelligent electronics systems.

Aging reliability simulation is a key step in the design verification. The traditional aging reliability simulation is very time-consuming so that it causes low verification efficiency and increase time and development cost. We have proposed a data-

driven method, deep H-GCN, to achieve a significant speedup in the aging reliability verification. In the industry, this methodology can promote the development of IC design and verification.

Routing is a key step in the physical design stage and plays an important role in the design flow. The traditional verification-then-fix approaches are hard to achieve design closure. We have proposed a thermal-driven PCB routing methodology to integrate thermal reliability verification into the routing stage. Our proposed methodology can be easily transferred to be the other performance-driven routing methodology. In the industry, this methodology can achieve significant speedup in the design closure so that it can reduce time and development cost.

CNN compression is a key step for facilitating practical deployment in the sensor system. We have proposed a Bayesian sharing grouped convolution. It is easy to customize in many advanced CNN architecture. Instead of using expensive and advanced processing unit, CNN compression can facilitate deployment on the cheap processing unit and reduce cost. Besides, CNN compression is also a key step to improve inference efficiency.

On the other hand, our proposed methodologies contain Bayesian modeling, learning-based methodology and gradient-based optimization. As shown in Fig. 7.1, Bayesian modeling is customized for sensor calibration and Bayesian sharing grouped convolution. Learning-based methodology is developed for aging degradation estimation and thermal-driven PCB routing. All formulations

are handled by gradient-based optimization. Thanks to these powerful methodologies, they can be leveraged to achieve hardware reliability and efficiency. We hope this thesis can promote the development of advanced sensor system and achieve good hardware reliability and efficiency.

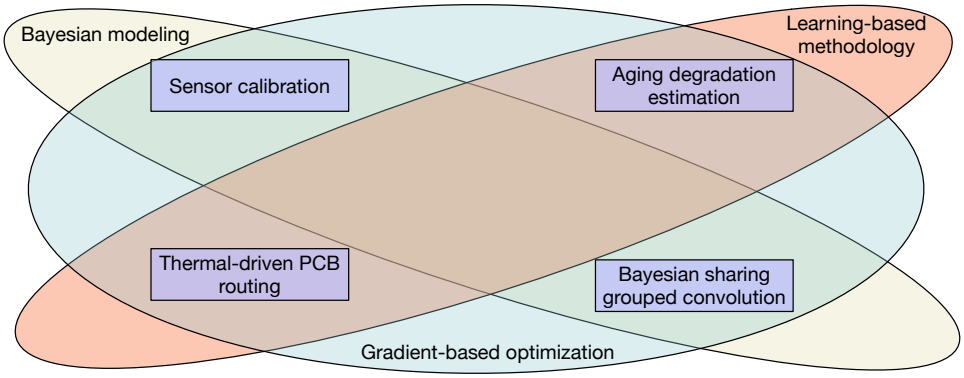


Figure 7.1 The relationship between methodologies and works of this thesis.

□ End of chapter.

Bibliography

[1] Boost. https://www.boost.org/.

[2] Engineer's guide to accurate sensor measurements. http://download.ni.com/evaluation/daq/25188_Sensor_WhitePaper_IA.pdf.

[3] Group LASSO Solver. https://github.com/fabianp/group_lasso.

[4] Hyperlynx. https://eda.sw.siemens.com/en-US/pcb/hyperlynx/.

[5] ImageNet dataset. http://www.image-net.org.

[6] Pcb layout framework. https://github.com/aspdac-submission-pcb-layout/PCB-Layout-Framework.

[7] Pcbbenchmarks. https://github.com/aspdac-submission-pcb-layout/PCBBenchmarks.

[8] Pytorch. https://pytorch.org/.

[9] The CIFAR-10 and CIFAR-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html.

[10] 2-Terminal IC Temperature Transducer. https://www.analog.com/media/en/technical-documentation/data-sheets/AD590.pdf, 2013.

[11] *New Generation Reliability Model*, 2016. http://www.mos-ak.org/berkeley_2016/publications/T11_Xie_MOS-AK_Berkeley_2016.pdf.

[12] *National Renewable Energy Laboratory OpenStudio Standards*, 2018. https://github.com/NREL/openstudio-standards.

[13] *OpenStudio®*, 2018. https://www.openstudio.net.

[14] *Re-Thinking Reliability Analysis*, 2018. https://indico.esa.int/event/222/contributions/2089/attachments/1779/2077/AMICSA_06_18_18_ReThinking_Reliability_Analysis.pdf.

[15] L. Balzano and R. Nowak. Blind calibration of sensor networks. In *International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 79–88. ACM, 2007.

[16] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor. Electronic circuit reliability

modeling. *Microelectronics Reliability*, 46(12):1957–1979, 2006.

[17] C.-E. Bichot and P. Siarry. *Graph partitioning*. John Wiley & Sons, 2013.

[18] E. Bogatin. *Signal and power integrity–simplified*. Pearson Education, 2010.

[19] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[20] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang. Heterogeneous network embedding via deep architectures. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 119–128, 2015.

[21] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker. Learning efficient object detection models with knowledge distillation. In *Conference on Neural Information Processing Systems (NIPS)*, pages 742–751, 2017.

[22] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning (ICML)*, pages 1725–1735, 2020.

[23] P. Chen, S. Si, Y. Li, C. Chelba, and C.-J. Hsieh. GroupReduce: Block-wise low-rank approximation for

neural language model shrinking. In *Conference on Neural Information Processing Systems (NIPS)*, pages 10988–10998, 2018.

[24] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu. Faster region-based hotspot detection. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[25] X. Chen, X. Li, and S. X.-D. Tan. From robust chip to smart building: CAD algorithms and methodologies for uncertainty analysis of building performance. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 457–464, 2015.

[26] X. Chen, X. Li, and S. X.-D. Tan. Overview of cyber-physical temperature estimation in smart buildings: From modeling to measurements. In *INFOCOM Workshops*, pages 251–256, 2016.

[27] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[28] J. Cong and P. H. Madden. Performance-driven routing with multiple sources. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 1997.

[29] R. Diestel. *Graph Theory*. Springer-Verlag New York, 2005.

[30] J. Ding, A. V. Gribok, J. W. Hines, and B. Rasmussen. Redundant sensor calibration monitoring using independent component analysis and principal component analysis. *Real-time systems*, 27(1):27–47, 2004.

[31] J.-W. Fang and Y.-W. Chang. Area-i/o flip-chip routing for chip-package co-design considering signal skews. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 29(5):711–721, 2010.

[32] J. Feng, S. Megerian, and M. Potkonjak. Model-based calibration for sensor networks. In *Sensors*, volume 2, pages 737–742. IEEE, 2003.

[33] M. Feurer, J. T. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 1128–1135, 2015.

[34] K. Ganchev, B. Taskar, and J. Gama. Expectation maximization and posterior constraints. In *Conference on Neural Information Processing Systems (NIPS)*, pages 569–576, 2008.

[35] X. Gao, C. Deng, M. Liu, Z. Zhang, D. Z. Pan, and Y. Lin. Layout symmetry annotation for analog circuits

with graph neural networks. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 152–157, 2021.

[36] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu. Hotspot detection via attention-based deep layout metric learning. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2020.

[37] P. V. Giampouras, A. A. Rontogiannis, and K. D. Koutroumbas. Alternating iteratively reweighted least squares minimization for low-rank matrix factorization. *IEEE Transactions on Signal Processing*, 67(2):490–503, 2019.

[38] R. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[39] G. H. Golub and C. F. Van Loan. *Matrix computations*. JHU Press, 2012.

[40] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Conference on Neural Information Processing Systems (NIPS)*, pages 1025–1035, 2017.

[41] S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing deep neural networks with pruning, trained

quantization and Huffman coding. In *International Conference on Learning Representations (ICLR)*, 2016.

[42] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Conference on Neural Information Processing Systems (NIPS)*, pages 1135–1143, 2015.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[44] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1389–1397, 2017.

[45] T.-Y. Ho, Y.-W. Chang, S.-J. Chen, and D.-T. Lee. Crosstalk-and performance-driven multilevel full-chip routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 24(6):869–878, 2005.

[46] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[47] Y. Hu, S. Sun, J. Li, X. Wang, and Q. Gu. A novel channel pruning method for deep neural network compression. *arXiv preprint arXiv:1805.11394*, 2018.

[48] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger. CondenseNet: An efficient densenet using learned group convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2752–2761, 2018.

[49] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[50] J. Huang, T. Zhang, and D. Metaxas. Learning with structured sparsity. *Journal of Machine Learning Research*, 12(Nov):3371–3412, 2011.

[51] Q. Huang, C. Fang, F. Yang, X. Zeng, and X. Li. Efficient multivariate moment estimation via Bayesian model fusion for analog and mixed-signal circuits. In *ACM/IEEE Design Automation Conference (DAC)*, page 169, 2015.

[52] Q. Huang, C. Fang, F. Yang, X. Zeng, D. Zhou, and X. Li. Efficient performance modeling via dual-prior Bayesian model fusion for analog and mixed-signal circuits. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2016.

[53] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky. Nonparametric belief propagation for sensor self-calibration. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 3, pages 861–864. IEEE, 2004.

[54] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

[55] J. Jarvis and D. R. Shier. Graph-theoretic analysis of finite Markov chains. *Applied mathematical modeling: a multidisciplinary approach*, page 85, 1999.

[56] M. Javaheripi, M. Samragh, T. Javidi, and F. Koushanfar. Adans: Adaptive non-uniform sampling for automated design of compact dnns. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):750–764, 2020.

[57] Y. Kim and A. M. Rush. Sequence-level knowledge distillation. In *Conference Empirical Methods in Natural Language Processing (EMNLP)*, pages 1317–1327, 2016.

[58] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. In *Conference on Neural Information Processing Systems (NIPS)*, pages 2575–2583, 2015.

[59] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2014.

[60] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[61] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2018.

[62] J. Klicpera, S. Weißenberger, and S. Günnemann. Diffusion improves graph learning. In *Conference on Neural Information Processing Systems (NIPS)*, pages 13354–13366, 2019.

[63] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012.

[64] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar. Gana: graph convolutional network based automated netlist annotation for analog circuits. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, pages 55–60, 2020.

[65] K. Kundert. *The Designer' s Guide to Spice and Spectre®*. Springer Science & Business Media, 2006.

[66] S. Kundu, M. Nazemi, M. Pedram, K. M. Chugg, and P. A. Beerel. Pre-defined sparsity for low-complexity convolutional neural networks. *IEEE Transactions on Computers*, 69(7):1045–1058, 2020.

[67] G. Li, X. Shen, J. Li, and J. Wang. Diagonal-kernel convolutional neural networks for image classification. *Digital Signal Processing*, 108:102898, 2021.

[68] G. Li, M. Zhang, J. Li, F. Lv, and G. Tong. Efficient densely connected convolutional neural networks. *Pattern Recognition*, 109:107610, 2021.

[69] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

[70] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 32, 2018.

[71] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, W. Xu, S. S. Sapatnekar, R. Harjani, and J. Hu. A customized graph neural network model for guiding analog IC placement. In

IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pages 1–9, 2020.

[72] Z. Li, Y. Wang, A. Yang, and H. Yang. Drift detection and calibration of sensor networks. In *International Conference on Wireless Communications & Signal Processing (WCSP)*, pages 1–6. IEEE, 2015.

[73] B. Lin and B. Yu. Smart building uncertainty analysis via adaptive Lasso. *IET Cyber-Physical Systems: Theory & Applications*, 2(1):42–48, 2017.

[74] T.-C. Lin, D. Merrill, Y.-Y. Wu, C. Holtz, and C.-K. Cheng. A unified printed circuit board routing algorithm with complicated constraints and differential pairs. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2021.

[75] S. Ling and T. Strohmer. Self-calibration and bilinear inverse problems via linear least squares. *SIAM Journal on Imaging Sciences (SIIMS)*, 11(1):252–292, 2018.

[76] J. Liu, B. Ni, Y. Yan, P. Zhou, S. Cheng, and J. Hu. Pose transferrable person re-identification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4099–4108, 2018.

[77] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan. Towards decrypting the art of analog lay-

out: Placement quality prediction via transfer learning. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2020.

[78] S. Liu, Q. Sun, P. Liao, Y. Lin, and B. Yu. Global placement with deep learning-enabled explicit routability optimization. *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2021.

[79] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, pages 21–37, 2016.

[80] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2736–2744, 2017.

[81] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[82] C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In *Conference on Neural Information Processing Systems (NIPS)*, pages 3288–3298, 2017.

[83] P. Luo, J. Ren, Z. Peng, R. Zhang, and J. Li. Differentiable learning-to-normalize via switchable normalization. In *International Conference on Learning Representations (ICLR)*, 2019.

[84] Y. Ma, R. Chen, W. Li, F. Shang, W. Yu, M. Cho, and B. Yu. A unified approximation framework for compressing and accelerating deep neural networks. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 376–383. IEEE, 2019.

[85] Y. Ma, Z. He, W. Li, L. Zhang, and B. Yu. Understanding graphs in EDA: From shallow to deep learning. In *ACM International Symposium on Physical Design (ISPD)*, pages 119–126, 2020.

[86] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu. High performance graph convolutional networks with applications in testability analysis. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[87] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *International Conference on Machine Learning (ICML)*, pages 689–696, 2009.

[88] M. Meissner and L. Hedrich. Feats: Framework for explorative analog topology synthesis. *IEEE Transactions on*

Computer-Aided Design of Integrated Circuits and Systems (TCAD), 34(2):213–226, 2014.

[89] E. Miluzzo, N. D. Lane, A. T. Campbell, and R. Olfati-Saber. Calibree: A self-calibration system for mobile sensor networks. In *International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 314–331. Springer, 2008.

[90] B. Minnehan and A. Savakis. Cascaded projection: End-to-end network compression and acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10715–10724, 2019.

[91] A. Y. Ng. Preventing overfitting of cross-validation data. In *International Conference on Machine Learning (ICML)*, volume 97, pages 245–253, 1997.

[92] K. Ni, N. Ramanathan, M. N. H. Chehade, L. Balzano, S. Nair, S. Zahedi, E. Kohler, G. Pottie, M. Hansen, and M. Srivastava. Sensor network data fault types. *ACM Transactions on Sensor Networks (TOSN)*, 5(3):25, 2009.

[93] J. Ou and Y. Li. Vector-kernel convolutional neural networks. *Neurocomputing*, 330:253–258, 2019.

[94] M. M. Ozdal and M. D. Wong. A length-matching routing algorithm for high-performance printed circuit boards. *IEEE Transactions on Computer-Aided Design of Inte-*

grated Circuits and Systems (TCAD), 25(12):2784–2794, 2006.

[95] J. Palmer, K. Kreutz-Delgado, B. D. Rao, and D. P. Wipf. Variational EM algorithms for non-Gaussian latent variable models. In *Conference on Neural Information Processing Systems (NIPS)*, pages 1059–1066, 2006.

[96] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[97] D. Randall. Rapidly mixing Markov chains with applications in computer science and physics. *Computing in Science & Engineering (CiSE)*, 8(2):30–41, 2006.

[98] B. Razavi. *Design of analog CMOS integrated circuits*. Tata McGraw-Hill Education, 2002.

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

[100] C. Robert. *Machine learning, a probabilistic perspective*. Taylor & Francis, 2014.

[101] Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node clas-

sification. In *International Conference on Learning Representations (ICLR)*, 2020.

[102] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015.

[103] T. Salimans, D. Kingma, and M. Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning (ICML)*, pages 1218–1226, 2015.

[104] C. Schlünder, K. Waschneck, P. Rotter, S. Lachenmann, H. Reisinger, F. Ungar, and G. Georgakos. From device aging physics to automated circuit reliability sign off. In *IEEE International Reliability Physics Symposium (IRPS)*, pages 1–12. IEEE, 2019.

[105] D. Sengupta and S. S. Sapatnekar. Estimating circuit aging due to BTI and HCI using ring-oscillator-based sensors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 36(10):1688–1701, 2017.

[106] W. Shao, T. Meng, J. Li, R. Zhang, Y. Li, X. Wang, and P. Luo. SSN: Learning sparse switchable normalization

via sparsestmax. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 443–451, 2019.

[107] W. Shao, S. Tang, X. Pan, P. Tan, X. Wang, and P. Luo. Channel equilibrium networks for learning deep representation. In *International Conference on Machine Learning (ICML)*, pages 8645–8654. PMLR, 2020.

[108] N. A. Sherwani. *Algorithms for VLSI physical design automation*. Springer Science & Business Media, 2012.

[109] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[110] M. S. Stanković, S. S. Stanković, and K. H. Johansson. Distributed blind calibration in lossy sensor networks via output synchronization. *IEEE Transactions on Automatic Control (TAC)*, 60(12):3257–3262, 2015.

[111] H.-Y. Su, C.-H. Hsu, and Y.-L. Li. Subhunter: A high-performance and scalable sub-circuit recognition method with prüfer-encoding. In *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, pages 1583–1586, 2015.

[112] Z. Su, L. Fang, W. Kang, D. Hu, M. Pietikäinen, and L. Liu. Dynamic group convolution for accelerating con-

volutional neural networks. In *European Conference on Computer Vision (ECCV)*, pages 138–155. Springer, 2020.

[113] Q. Sun, T. Chen, J. Miao, and B. Yu. Power-driven dnn dataflow optimization on fpga. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–7. IEEE, 2019.

[114] K. B. Sutaria, P. Ren, A. Mohanty, X. Feng, R. Wang, R. Huang, and Y. Cao. Duty cycle shift under static/dynamic aging in 28nm hk-mg technology. In *IEEE International Reliability Physics Symposium (IRPS)*, pages CA–7. IEEE, 2015.

[115] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[116] M. Takruri, S. Rajasegarar, S. Challa, C. Leckie, and M. Palaniswami. Spatio-temporal modelling-based drift-aware wireless sensor networks. *IET wireless sensor systems*, 1(2):110–122, 2011.

[117] R. H. Tu, E. Rosenbaum, W. Y. Chan, C. C. Li, E. Minami, K. Quader, P. K. Ko, and C. Hu. Berkeley reliability tools-BERT. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 12(10):1524–1534, 1993.

[118] F. Wang, P. Cachecho, W. Zhang, S. Sun, X. Li, R. Kanj, and C. Gu. Bayesian model fusion: large-scale performance modeling of analog and mixed-signal circuits by reusing early-stage data. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 35(8):1255–1268, 2016.

[119] H. Wang, X. Hu, Q. Zhang, Y. Wang, L. Yu, and H. Hu. Structured pruning for efficient convolutional neural networks via incremental regularization. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):775–788, 2019.

[120] 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 *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[121] H. Wang, Q. Zhang, Y. Wang, L. Yu, and H. Hu. Structured pruning for efficient convnets via incremental regularization. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[122] J. Wang, H. Bai, J. Wu, and J. Cheng. Bayesian automatic model compression. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):727–736, 2020.

[123] X. Wang, M. Kan, S. Shan, and X. Chen. Fully learnable group convolution for acceleration of deep neural net-

works. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9049–9058, 2019.

[124] Y. Wang, A. Yang, X. Chen, P. Wang, Y. Wang, and H. Yang. A deep learning approach for blind drift calibration of sensor networks. *IEEE Sensors Journal*, 17(13):4158–4171, 2017.

[125] Y. Wang, A. Yang, Z. Li, X. Chen, P. Wang, and H. Yang. Blind drift calibration of sensor networks using sparse Bayesian learning. *IEEE Sensors Journal*, 16(16):6249–6260, 2016.

[126] Y. Wang, A. Yang, Z. Li, P. Wang, and H. Yang. Blind drift calibration of sensor networks using signal space projection and Kalman filter. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 1–6, 2015.

[127] X. Wei, Y. Liang, and J. Cong. Overcoming data transfer bottlenecks in FPGA-based DNN accelerators via layer conscious memory management. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[128] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, pages 2074–2082, 2016.

[129] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. In *ACM international workshop on Wireless sensor networks and applications*, pages 59–67. ACM, 2002.

[130] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinc, D. Neumann, A. Osman, D. Marpe, H. Schwarz, et al. DeepCABAC: Context-adaptive binary arithmetic coding for deep neural network compression. In *ICML Workshop*, 2019.

[131] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning (ICML)*, pages 6861–6871, 2019.

[132] L. Wu, I. E.-H. Yen, Z. Zhang, K. Xu, L. Zhao, X. Peng, Y. Xia, and C. Aggarwal. Scalable global alignment graph kernel using random features: From node embedding to graph embedding. In *ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1418–1428, 2019.

[133] Y. Wu and K. He. Group normalization. In *European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[134] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks.

In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1492–1500, 2017.

[135] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu. Routenet: Routability prediction for mixed-size designs using convolutional neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2018.

[136] Z. Xie, H. Ren, B. Khailany, Y. Sheng, S. Santosh, J. Hu, and Y. Chen. Powernet: Transferable dynamic ir drop estimation via maximum convolutional neural network. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 13–18. IEEE, 2020.

[137] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning (ICML)*, pages 5453–5462, 2018.

[138] K. Yamamoto and K. Maeno. PCAS: Pruning channels with attention statistics for deep network compression. In *British Machine Vision Conference (BMVC)*, 2019.

[139] T. Yan and M. D. Wong. Recent research development in PCB layout. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 398–403. IEEE, 2010.

[140] T. Yan and M. D. Wong. Correctly modeling the diagonal capacity in escape routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 31(2):285–293, 2012.

[141] T. Yan, P.-C. Wu, Q. Ma, and M. D. Wong. On the escape routing of differential pairs. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 614–620. IEEE, 2010.

[142] Y. Yan, B. Ni, and X. Yang. Predicting human interaction via relative attention model. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3245–3251, 2017.

[143] Y. Yang, Q. Huang, B. Wu, T. Zhang, L. Ma, G. Gambardella, M. Blott, L. Lavagno, K. Vissers, J. Wawrzynek, et al. Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded FPGAs. In *ACM Symposium on FPGAs*, pages 23–32, 2019.

[144] M. B. Yelten, P. D. Franzon, and M. B. Steer. Surrogate-model-based analysis of analog circuits—part II: Reliability analysis. *IEEE Transactions on Device and Materials Reliability (TDMR)*, 11(3):466–473, 2011.

[145] Z. Yu, Z. Sun, R. Wang, J. Zhang, and R. Huang. Hot carrier degradation-induced dynamic variability in FinFETs:

Experiments and modeling. *IEEE Transactions on Electron Devices (TED)*, 67(4):1517–1522, 2020.

[146] Q. Zhang, M. Zhang, T. Chen, J. Fan, Z. Yang, and G. Li. Electricity theft detection using generative models. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 270–274. IEEE, 2018.

[147] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu. Recent advances in convolutional neural network acceleration. *Neurocomputing*, 323:37–51, 2019.

[148] R. Zhang, Z. Peng, L. Wu, Z. Li, and P. Luo. Exemplar normalization for learning deep representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12726–12735, 2020.

[149] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang. Interleaved group convolutions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 4373–4382, 2017.

[150] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang. A systematic DNN weight pruning framework using alternating direction method of multipliers. In *European Conference on Computer Vision (ECCV)*, pages 184–199, 2018.

[151] T. Zhang, Y. Zhang, Z. Li, Z. Song, and H. Liu. System-level calibration for data fusion in wireless sensor networks.

ACM Transactions on Sensor Networks (TOSN), 9(3):28, 2013.

[152] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[153] Y. Zhang and P. E. Bagnoli. A modeling methodology for thermal analysis of the pcb structure. *Microelectronics Journal*, 45(8):1033–1052, 2014.

[154] Y. Zhang, A. Pennatini, and P. E. Bagnoli. A thermal model for the pcb structure including thermal contribution of traces and vias. In *IEEE International Workshop on Thermal Investigation of ICs and Systems*, pages 1–6. IEEE, 2012.

[155] Z. Zhang, J. Li, W. Shao, Z. Peng, R. Zhang, X. Wang, and P. Luo. Differentiable learning-to-group channels via groupable convolutional neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3542–3551, 2019.

[156] Z. Zhang and B. D. Rao. Sparse signal recovery with temporally correlated source vectors using sparse Bayesian learning. *IEEE Journal of Selected Topics in Signal Processing*, 5(5):912–926, 2011.

[157] Z. Zhang and B. D. Rao. Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation. *IEEE Transactions on Signal Processing*, 61(8):2009–2015, 2013.

[158] R. Zhao and W. Luk. Efficient structured pruning and architecture searching for group convolution. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1961–1970, 2019.

[159] H. Zhou, W. Jin, and S. X.-D. Tan. Gridnet: Fast data-driven em-induced ir drop prediction and localized fixing for on-chip power grid networks. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[160] Z. Zhou, Z. Zhu, J. Chen, Y. Ma, B. Yu, T.-Y. Ho, G. Lemieux, and A. Ivanov. Congestion-aware global routing using deep convolutional generative adversarial networks. In *ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6. IEEE, 2019.

[161] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan. Geniusroute: A new analog routing paradigm using generative neural network guidance. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[162] M. Zwerger, M. Neuner, and H. Graeb. Power-down circuit synthesis for analog/mixed-signal. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 656–663, 2015.

[163] M. Zwerger, M. Neuner, and H. Graeb. Analog power-down synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 36(12):1954–1967, 2017.

The Chinese University of Hong Kong
Academic Honesty Declaration Statement

Submission Details

Student Name	CHEN Tinghuan (s1155100748)
Thesis Title	Methodologies for Hardware Reliability and Efficiency
Submitted File Name	1155100748_RPG_0_1_1_thesis.pdf

Agreement and Declaration on Student's Work Submitted to VeriGuide

VeriGuide is intended to help the University to assure that works submitted by students as part of course requirement are original, and that students receive the proper recognition and grades for doing so. The student, in submitting his/her work ("this Work") to VeriGuide, warrants that he/she is the lawful owner of the copyright of this Work. The student hereby grants a worldwide irrevocable non-exclusive perpetual licence in respect of the copyright in this Work to the University. The University will use this Work for the following purposes.

(a) Checking that this Work is original
The University needs to establish with reasonable confidence that this Work is original, before this Work can be marked or graded. For this purpose, VeriGuide will produce comparison reports showing any apparent similarities between this Work and other works, in order to provide data for teachers to decide, in the context of the particular subjects, course and assignment. However, any such reports that show the author's identity will only be made available to teachers, administrators and relevant committees in the University with a legitimate responsibility for marking, grading, examining, degree and other awards, quality assurance, and where necessary, for student discipline.

(b) Anonymous archive for reference in checking that future works submitted by other students of the University are original
The University will store this Work anonymously in an archive, to serve as one of the bases for comparison with future works submitted by other students of the University, in order to establish that the latter are original. For this purpose, every effort will be made to ensure this Work will be stored in a manner that would not reveal the author's identity, and that in exhibiting any comparison with other work, only relevant sentences/ parts of this Work with apparent similarities will be cited. In order to help the University to achieve anonymity, this Work submitted should not contain any reference to the student's name or identity except in designated places on the front page of this Work (which will allow this information to be removed before archival).

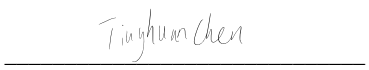
(c) Research and statistical reports
The University will also use the material for research on the methodology of textual comparisons and evaluations, on teaching and learning, and for the compilation of statistical reports. For this purpose, only the anonymously archived material will be used, so that student identity is not revealed.

I confirm that the above submission details are correct.

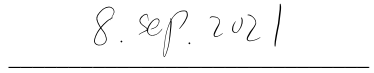
I have read the above and in submitting this Work fully agree to all the terms.

I declare that this Work here submitted is original except for source material explicitly acknowledged, the same or closely related material has not been previously submitted for same or different courses, and that the submitted soft copy with details listed in the <Submission Details> is identical to the hard copy(ies), if any, which has(have) been / is(are) going to be submitted.

I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University website on <Honesty in Academic Work: A Guide for Students and Teachers>.



Signature (CHEN Tinghuan)



Date

Instruction for Submitting Hard Copy / Soft Copy of the Assignment

This signed declaration should be attached to the hard copy of the first draft of your thesis when it is submitted to the Graduate School..