3-28-2022

Thermal Aware Design Automation of the Electronic Control System for Autonomous Vehicles

Ajinkya Bankar
abank013@fiu.edu

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

THERMAL AWARE DESIGN AUTOMATION OF THE ELECTRONIC CONTROL SYSTEM FOR AUTONOMOUS VEHICLES

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

ELECTRICAL AND COMPUTER ENGINEERING

by

Ajinkya Bankar

2022

To: Dean John L. Volakis
College of Engineering and Computing

This dissertation, written by Ajinkya Bankar, and entitled Thermal Aware Design Automation of the Electronic Control System for Autonomous Vehicles, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

Nezih Pala

Ou Bai

Kemal Akkaya

Deng Pan

Gang Quan, Major Professor

Date of Defense: March 28, 2022

The dissertation of Ajinkya Bankar is approved.

Dean John L. Volakis
College of Engineering and Computing

Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2022

DEDICATION

I dedicate this dissertation to my younger brother, the late Abhijit (Pappu). I am grateful that I could complete your wish. May you find peace and happiness in Paradise!

ACKNOWLEDGMENTS

First and foremost, I would like to thank my master H. H. Sri Sri Ravi Shankar, for giving me the strength and patience to accomplish my late younger brother's dream of getting a Ph.D. degree.

Next, I would like to express my deepest gratitude to my advisor, Dr. Gang Quan. I consider myself fortunate and blessed to have Dr. Quan as my advisor for the last four years of my Ph.D. study, who provided constant supervision, personal attention, suggestions, unending encouragement, and strong support. I respect his determination, breadth of knowledge, and unwavering commitment to students and high-quality research. Dr. Quan's ideas and suggestions have aided me in successfully completing this dissertation.

I am very grateful to my Ph.D. committee members, Dr. Nezih Pala, Dr. Ou Bai, Dr. Kemal Akkaya, and Dr. Deng Pan, for their thoughtful insights and suggestions in improving my dissertation. I am incredibly proud to have such knowledgeable people serving on my dissertation committee. I want to give my heartfelt thanks to collaborators, Dr. Vivek Chaturvedi and Dr. Shi Sha for their endless encouragement and support in accomplishing my career objectives. I am thankful to Dr. Gustavo Chaparro-Baquero and Dr. Somar Homsi for their help and insightful suggestions throughout the stay in Miami.

I am also thankful to the ECE department for creating an impactful research environment and providing financial support. I especially thank University Graduate School, Florida International University, for awarding Dissertation Year Fellowship for Fall 2021 and Spring 2022 semesters, which helped me to laser-focus on my doctoral research.

Finally, I want to thank my Bankar family, relatives, friends, the Art of Living family, and extended family of the United States for their unlimited love, faith, encouragement, blessings, and prayers. My life-long gratitude goes to my dearest mother, Mrs. Jayashree Bankar, and my loving father, Mr. Suryakant Bankar, for all the love and affection they have showered. I am grateful to my wife, Mrs. Diksha Bankar, for accompanying me during my Ph.D. journey. And, cute thanks to my loving daughter, Advika, for bringing moments of joy and happiness. I want to thank my uncle, late Mr. Chandrakant Bankar, for being a great wall of support and inspiration in my life. May you find peace and happiness in Paradise!

ABSTRACT OF THE DISSERTATION

THERMAL AWARE DESIGN AUTOMATION OF THE ELECTRONIC CONTROL SYSTEM FOR AUTONOMOUS VEHICLES

by

Ajinkya Bankar

Florida International University, 2022

Miami, Florida

Professor Gang Quan, Major Professor

The autonomous vehicle (AV) technology, due to its tremendous social and economical benefits, is transforming the entire world in the coming decades. However, significant technical challenges still need to be overcome until AVs can be safely, reliably, and massively deployed. Temperature plays a key role in the safety and reliability of an AV, not only because a vehicle is subjected to extreme operating temperatures but also because the increasing computations demand more powerful IC chips, which can lead to higher operating temperature and large thermal gradient. In particular, as the underpinning technology for AV, artificial intelligence (AI) requires substantially increased computation and memory resources, which have been growing exponentially through recent years and further exacerbated the thermal problems. High operating temperature and large thermal gradient can reduce the performance, degrade the reliability, and even cause an IC to fail catastrophically.

We believe that dealing with thermal issues must be coupled closely in the design phase of the AVs' electronic control system (ECS). To this end, first, we study how to map vehicle applications to ECS with heterogeneous architecture to satisfy peak temperature constraints and optimize latency and system-level reliability. We present a mathematical programming model to bound the peak temperature for the ECS. We also develop an approach based on the genetic algorithm to bound the peak temperature under varying execution time scenarios and optimize the system-level reliability of the ECS. We present several computationally efficient techniques for system-level mean-time-to-failure (MTTF) computation, which show several-order-of-magnitude speed-up over the state-of-the-art method. Second, we focus on studying the thermal impacts of AI techniques. Specifically, we study how the thermal impacts for the memory bit flipping can affect the prediction accuracy of a deep neural network (DNN). We develop a neuron-level analytical sensitivity estimation framework to quantify this impact and study its effectiveness with popular DNN architectures. Third, we study the problem of incorporating thermal impacts into mapping the parameters for DNN neurons to memory banks to improve prediction accuracy. Based on our developed sensitivity metric, we develop a bin-packing-based approach to map DNN neuron parameters to memory banks with different temperature profiles. We also study the problem of identifying the optimal temperature profiles for memory systems that can minimize the thermal impacts. We show that the thermal aware mapping of DNN neuron parameters on memory banks can significantly improve the prediction accuracy at a high-temperature range than the thermal ignorant for state-of-the-art DNNs.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Self-driving cars, also known as autonomous vehicles (AVs), have the potential to transform transportation mobility and safety if they are mass-produced. The advancements in the automotive transportation area have been substantial over the years, with the most notable being the one related to autonomous vehicles, which considers numerous factors such as performance, comfort, low emissions, and, most important, safety. With the emergence of AVs, the electronic control system (ECS) design has become very complex, which plays a crucial role in reliable vehicle operation.

As a critical enabling technology, Artificial Intelligence (AI) provides human brain-like functionality to drive the AVs on the road, simultaneously presenting safety and reliability challenges. The accuracy of AI methods, such as artificial deep neural networks (DNNs), mainly depends on the data from various sensors installed in the AV, and the size of DNNs used for processing sensor data is growing exponentially [1], demanding massive compute and storage resources. Besides the significant heat generated from the computing devices, which can lead to high temperature and large thermal gradients across IC chips and thus deteriorate the lifetime and safety conditions of ECS, AVs must be operated reliably under extreme environmental conditions, which make a thermal impact a critical issue in the design of ECS for AVs. Therefore, the substantially increased computation and memory capability demanded by AV technology have presented tremendous challenges in the ECS design optimization to satisfy rapidly elevated latency, throughput, reliability, and other safety-critical requirements [2].

In this chapter, we first discuss the current status and future of AVs and their potentially great social and technological impact. Further, we discuss the current artificial intelligence technique and its role in AV design. Later, we discuss the thermal challenges posed by the evolving data-centric AI-based vehicle technology. At last, we present our research problem with contributions.

1.1 About the Autonomous Vehicle Technology

Autonomous Vehicle Technology is a promising technology that will fundamentally transform transportation and mobility with a range of great social and economic benefits. Specifically, lowering the cost associated with traffic accidents, reducing fuel consumption and carbon emissions, and enabling far greater access to mobility for numerous elderly, disabled adults, and children will bring substantial potential market opportunities and has thus attracted enormous interest from numerous industrial sectors and companies.

Autonomous vehicle technology can significantly minimize road accidents caused due to human mistakes, which can bring substantial social benefits. According to international data, roadway accidents are among the top ten causes of worldwide mortality [3], and human errors are the most common cause of accidents [4]. According to the U.S. Department of Transportation, human error and negligence account for 94% of all road accidents, which cause approximately 35,000 deaths in the United States per year [5]. While at the global level, 1.35 million deaths are reported annually, and it is estimated that every year 20 to 50 million people suffer injuries [6]. As shown in Figure 1.1, over the period of recent industrialization, it is believed that machines can outperform the capabilities of human beings if trained with enough data. Therefore, AVs have the promising ability to eliminate the majority of traffic accidents by perceiving the road conditions better than human beings. Besides minimizing road accidents, AVs can help cut down the expenses caused due to crashes.

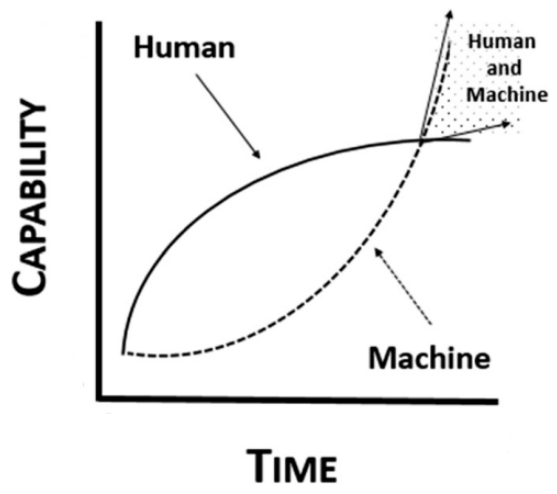


Figure 1.1: Progress rates in the capability of humans and machines [7]. The learning capability of the machines increase exponentially, but the capability of human beings saturate with time.

For example, in 2010, motor vehicle accidents cost the economy \$242 billion, which includes \$57.6 billion in lost workplace efficiency and \$594 billion in impaired life quality because of severe injuries [5]. Moreover, worldwide, costing at least 90% of global GDP (gross domestic product) [6].

AV technology can be a boon for an eco-friendly revolution. AVs can help to improve the stream of traffic and minimize traffic congestion. In 2014, Americans wasted approximately 6.9 billion hours in traffic, limiting work time while increasing fuel costs and pollution [5]. AVs might free up to 50 minutes of drive every day [8], significantly reducing emissions. Moreover, people can work during the commute in the AVs, which can save \$1.3 trillion in the US and \$5.6 trillion in the global economy [9]. AVs would help to solve the parking problem as the shared autonomous taxis can minimize the parking space requirement by 80% in dense cities and reduce the emission by 5-11% in search of the parking space. Similarly, shared autonomous vehicles are anticipated to be more energy efficient with a 35% emission reduction

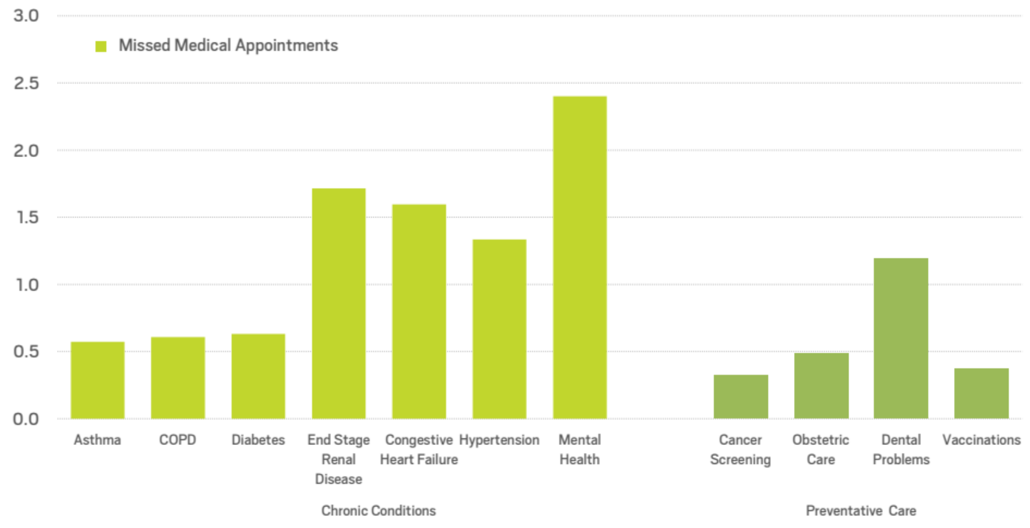


Figure 1.2: Missed health care appointments (in million) due to lack of transportation for disabled persons [12].

[10]. AVs can minimize traffic congestion by choosing to cruise on the streets where they can travel the slowest [11].

Millions of elderly adults, disabled, and children may have new mobility alternatives because of AVs. In the United States, there are 49 million people over the age of 65 and 53 million people with a disability [5]. In many parts of the world, the ability to drive is required for employment or independent life. Hence, a large portion of the demographics could benefit from independence due to AV technology. More important, as shown in Figure 1.2, due to a lack of transportation for disabled citizens, approx. 11 million medical appointments are missed annually. AV technology is beneficial for such elderly and disabled people, and it has the potential to save billions due to medical care at the right time, as shown in Figure 1.3.

Due to its potentially high social and market impacts, the automotive industry is keenly focusing on AV development. Google's Waymo has already begun testing a completely driverless taxicab service in the Phoenix suburbs of Arizona [13]. Tesla has offered an Autopilot driving system, which is mainly suitable for highway

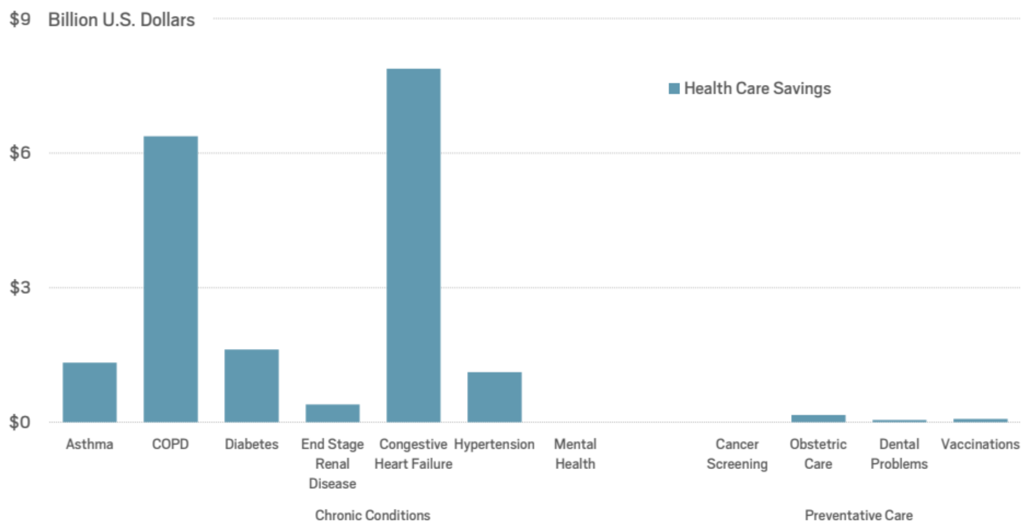


Figure 1.3: Health care savings due to adequate transportation for elderly and disabled persons [12].

driving. Other top auto giants, like, General Motors (GM) and Ford, have a similar capability in their newly launched vehicles with additional fully-automated lane-changing systems only in the GM models. While, Mercedes and Honda are cruising ahead in the technology innovation by introducing semi-autonomy, in which the drivers have to control the vehicle only in aberrant situations [14]. In the meantime, Argo AI is focusing on developing a fully autonomous driving system that can be integrated with Ford and Volkswagen models [15]. Oxbotica is collaborating with ZF, a German vehicle systems company, over the next five years to make the self-driving shuttle a true mainstay for European cities [16]. Recently, a tech giant, Apple, announced that it would launch fully autonomous cars within the next four years, making this race of innovation more competitive.

It is estimated that by 2025, automated mobility could be a $120 billion industry in the US [17]. Similarly, it is predicted that from 2018 through 2030, the market for autonomous vehicles will expand dramatically. By 2030, fully automated cars are estimated to generate $13.7 billion in revenue, and one out of every ten automo-

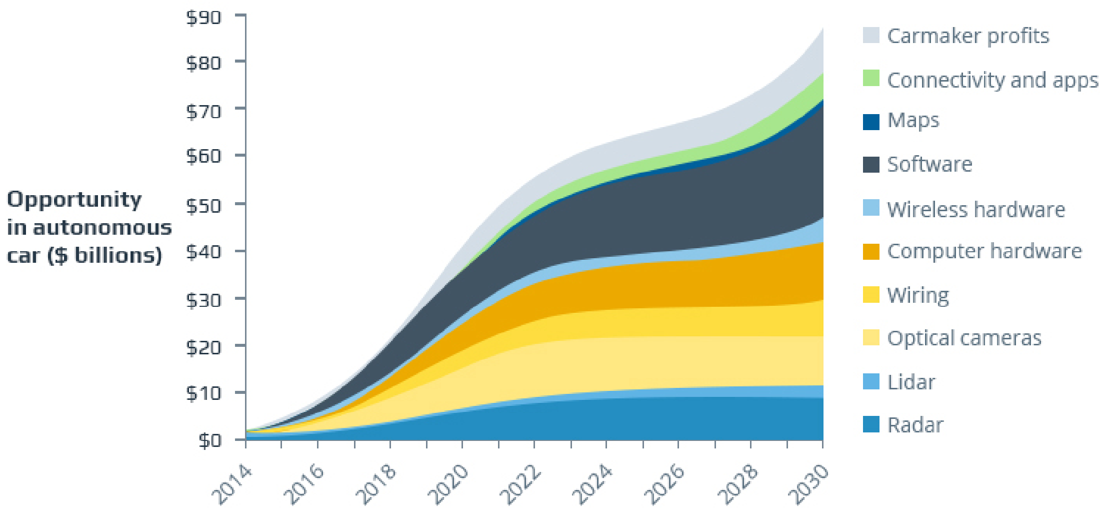


Figure 1.4: Economical opportunity in Autonomous Cars [19]. The software has the largest portion.

biles will be self-driving [18]. It is believed that the software in AVs would yield a significant opportunity among all the design variables, as shown in Figure 1.4. By 2035, it is predicted that the number of autonomous vehicles will increase by 75%, resulting in a $71 billion economic impact on the automotive sector by 2030, with a global market of 44 million vehicles [6].

Along with the development in AV technology, global demand for Advanced Driver Assistance Systems (ADAS) is expected to propel the automotive semiconductors and sensors industry forward. When robots take over control, these systems will be critical in assisting AI-powered driverless vehicles in navigating and detecting impediments on the road. As the AV technology heavily depends on the sensors deployed in the vehicle, eventually, the sensor market is projected to expand from $3 billion to $35 billion by 2030 [20].

Altogether, the AV technology can significantly mitigate human errors while driving, resulting in fewer fatalities and corresponding reduced expenses; assist in navigating the traffic flow smoothly, saving emissions; provide alternative mobility for children, elderly and disabled people. Such transforming capabilities have drawn tremendous investment into AV development by the automotive industry.

1.2 Artificial Intelligence Technology for Autonomous Vehicle Design

While AVs represent an enormous market opportunity, critical challenges must be overcome to fulfill their great social and economic benefits. As an underpinning technology, the autonomous vehicle industry has invested significant resources in researching advanced AI technology. As of today, there is no automotive company with complete autonomy in its matured commercial vehicle products to offer to the customers; instead, they need human vigilance in the loop. The most difficult challenge for those working in the self-driving technology industry is ensuring that vehicles can operate safely, reliably, and efficiently in complex and unexpected human-animal contexts.

The roots of artificial intelligence can be found in the late 1940s. As shown in Figure 1.5, the mathematical and logical theories of AI were evolved up to 1960, and the research was propelled after the Defense Advanced Research Projects Agency (DARPA) funded several institutions [21]. In 1986, the Navlab from Carnegie Mellon University developed the first autonomous van, and a large community of researchers was intrigued after the Deep Blue chess machine of IBM defeated the world champion in 1997. In recent times, AI research gained momentum due to the emergence of high-performance computing platforms and extensive storage, which were major bottlenecks in the previous century [22].

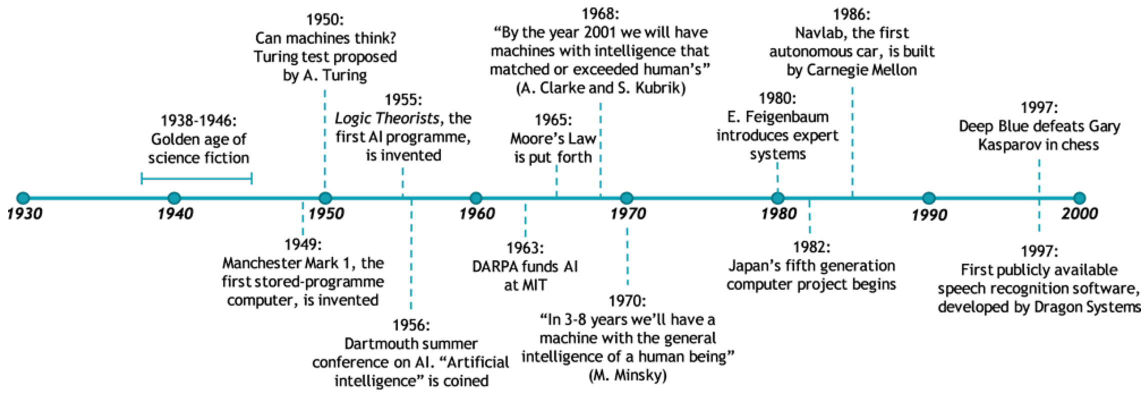

1950:
Can machines think? Turing test proposed by A. Turing
1968:
"By the year 2001 we will have machines with intelligence that matched or exceeded human's" (A. Clarke and S. Kubrik)
1986:
Navlab, the first autonomous car, is built by Carnegie Mellon
1938-1946:
Golden age of science fiction
1955:
Logic Theorists, the first AI programme, is invented
1965:
Moore's Law is put forth
1980:
E. Feigenbaum introduces expert systems
1997:
Deep Blue defeats Gary Kasparov in chess
1930
1940
1950
1960
1970
1980
1990
2000
1949:
Manchester Mark 1, the first stored-programme computer, is invented
1963:
DARPA funds AI at MIT
1982:
Japan's fifth generation computer project begins
1997:
First publicly available speech recognition software, developed by Dragon Systems
1956:
Dartmouth summer conference on AI. "Artificial intelligence" is coined
1970:
"In 3-8 years we'll have a machine with the general intelligence of a human being" (M. Minsky)

Figure 1.5: Artificial Intelligence technology development history [22]. The first AV was built in 1986.

The artificial intelligence paradigm deals with making machines smart. *Artificial intelligence* can be defined as an intelligent behavior in which a computer achieves tasks using a set of principles to tackle a given problem. *Machine learning* is a type of artificial intelligence that enables computers to learn and build intelligence based on their prior experiences. In contrast, *deep learning* is a category of machine learning that is inspired by the information processing mechanism of the human brain. It uses complex deep neural networks to extricate comprehensive attributes as they learn and assess their training data. However, machine learning differs significantly from deep learning because it requires input data attributes to be labeled manually using more rigid rulesets [23]. In contrast, deep learning can automatically identify the features to be utilized for classification in unsupervised exercises [24].

Deep learning has favored automotive companies to accelerate AV development as they rely on deep neural networks (DNNs) for more efficient sensor data processing. The LiDAR (Light Detection And Ranging), Imaging, and RADAR (Radio Detection And Ranging) sensors provide the surrounding environment's snapshot while traveling on the road, as shown in Figure 1.6. DNNs utilize all these sensors' data for object detection, object recognition, image classification, scene segmentation, driving lane detection, etc., vision-related tasks, which are based on the human

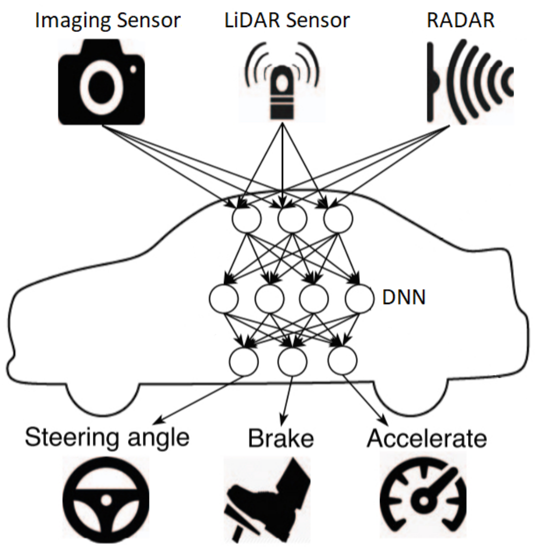


Figure 1.6: Role of DNN to control the AV [26]. As a representative, only one LiDAR, RADAR, imaging sensor, and DNN are shown. Practically, an array of sensors and DNNs are required on the AV to accomplish all vision-related tasks.

brain analogy [25]. It is required to build multiple DNNs for safe autonomous driving, each dedicated to a unique task. The signals generated by each DNN must be processed in real-time to drive the car, which is accomplished by a high-performance computing platform involving an array of CPUs and neural network accelerators. This network of processing units (PUs) commands the actuators, as shown in Figure 1.6, to control the steering angle, brake pedal, and speed-accelerator to drive the car on the road without human assistance.

The AV technology in the current state is not a safe and reliable mobility option as there have been several crashes and fatality incidents reported [27]. For example, in 2016, Google's autonomous Waymo car crashed with another transit vehicle, and several faults were detected in the safety-critical components like steering and brakes [28]. Similarly, Tesla's autonomous cars have been found in several accidents. In 2018, Uber's self-driving car accident killed a pedestrian, and the company has previously been engaged in 37 accidents. In another incident, Google's self-driving

Waymo taxi muddled on the street and went off on its own [29]. There are several malicious security attacks on the AV communication [19, 30, 31] and deliberate attacks on the perception systems [32, 33], raising the safety and reliability concerns for end-users [34, 35], which is beyond the scope of this dissertation.

The human brain-like functionality imitated by one of AI techniques, i.e., deep learning, assist in driving the car autonomously on the road. However, currently, AVs are not entirely reliable, and thermal impact is one of the major factors that threaten its safety and reliability, especially with the significant computing and memory requirement of the DNN execution.

1.3 Thermal Challenges for Autonomous Vehicles

Extreme environmental conditions and growing demand for computing and memory resources for highly accurate control decisions of the DNNs in the AVs bring severe thermal impact to the performance and lifetime reliability of the PUs as well as memory.

Thermal issues are one of the major sources of faults for automotive ECS. The electronics in modern automotives are more complicated than ever due to a combination of safety, luxury, and entertainment features. With the addition of more electronics for AI computation and memory, vehicles will inevitably emit more heat, resulting in unexpected thermal issues. According to one study, the temperature is the leading cause of failures in automotive electronics, as shown in Figure 1.7. For example, high temperatures impact the LiDAR sensors by causing deviation in the measured distance [36], imaging sensors' quality, and smart headlights' performance deteriorate [37]. While, the ambient temperature of the automotive PUs may vary

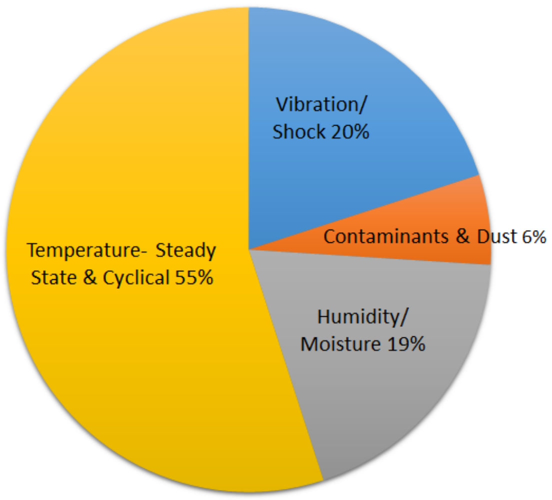


Figure 1.7: Fault distribution in Automotive electronics [40]. 55% of failures in automotive electronics are due to the temperature, significantly higher than counterparts.

significantly, as high as $90 - 155°C$ [38], increasing electronics failure probabilities and deteriorating the lifetime reliability of the ECS rapidly [39].

The complexity in AI computation and limited memory capacity introduces thermal challenges in AV design. The AVs are a highly complex system as they interact with unpredictable dynamic situations on the road, like other motor vehicles, motorbikes, cyclists, pedestrians, and wildlife. As a result, the software for vehicles is much larger, even 15× greater than that necessary for the aircraft, as shown in Figure 1.8. Besides, AVs process roughly 3 GB/s and 40 GB/s data at lower and higher levels of autonomy, respectively, generated by various sensors [41]. Moreover, in an effort to improve the prediction accuracy of the DNNs, there has been exponential growth in the Giga floating-point operations per second (GFLOPs), indicating a dramatic change in the computational complexity, as shown in Figure 1.9. Similarly, the number of parameters in different DNNs has increased exponentially, as shown in Figure 1.10. Contrarily, the memory space of the AI accelerators has scaled linearly, as observed in the figure. Altogether, soaring computational

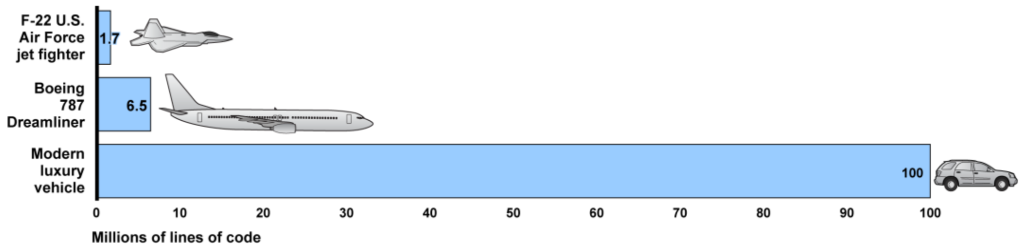


Figure 1.8: Modern vehicle and Aircraft software size comparison [43]. Modern vehicle software is huge in size than aircraft.

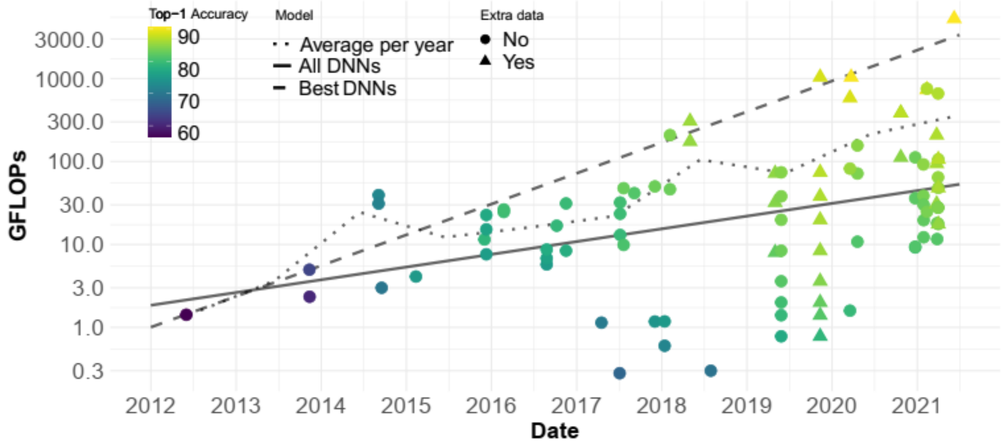


Figure 1.9: Compute complexity (GFLOPs) over the years [44]. Y-axis is a logarithmic scale; hence we can see an exponential rise in the complexity for models with the best accuracy.

complexity, storage requirements, and harsh environmental conditions increase the operating temperature along with thermal gradients across the PUs, which can be severe across 3D ICs [42].

High temperatures impact the performance of the PUs and memory of the ECS platform. When the temperature exceeds a certain threshold, it may cause PUs to reduce the operating speed or shut off automatically for self-protection purposes during the run-time [45], which can affect the performance of mission-critical tasks causing catastrophic events for AVs. Similarly, the performance of the conventional memory, i.e., Dynamic random-access memory (DRAM), degrades with the temperature due to an increase in memory refresh rate [46]. Whereas emerging non-volatile

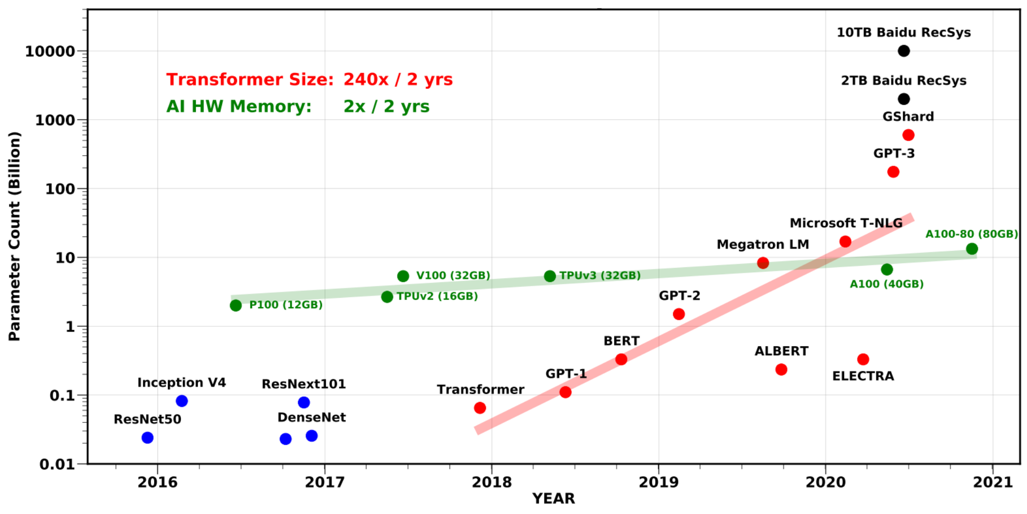


Figure 1.10: Growth pattern of AI accelerator memory size and number of parameters in neural network models [1]. The parameter count increased by 240× / 2 years, whereas the accelerator memory size is linearly increased as 2× / 2 years.

memories, such as Resistive RAM (ReRAM) and Spin-Transfer Torque Magnetoresistive RAM (STT-MRAM), eliminate the need for refreshing along with high density to alleviate the memory bottleneck problem, low power, and improved read latency [47]. In particular, for STT-MRAM, read access latency and read power consumption are reduced by 8.4% and 66.2%, respectively, compared to typical DRAM [48]. However, emerging non-volatile memories suffer from temperature-induced random errors. For example, the classification accuracy of the AI applications executing on ReRAM-based accelerators drops at high temperatures due to deviation in the cell conductances, as shown in Figure 1.11. Similarly, the switching probability of the STT-MRAM cells increases with the temperature, adversely impacting the classification accuracy of the AI applications, as later observed in the motivation example in Chapter 4. Therefore, it is critical that the run-time temperature of the underlying computing infrastructure for ECS should be well controlled.

The reliability challenges in the PUs and memory exacerbate more due to thermal impact. The $10-15°C$ rise in the temperature accelerates aging rapidly and reduces

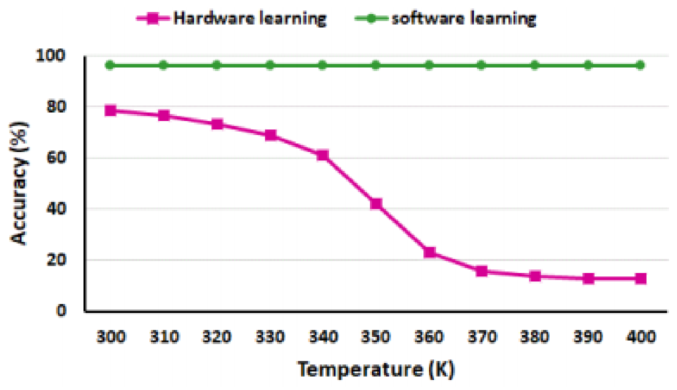


Figure 1.11: The temperature effect on the classification accuracy of ReRAM-based accelerator [49]. The accuracy drops with the temperature.

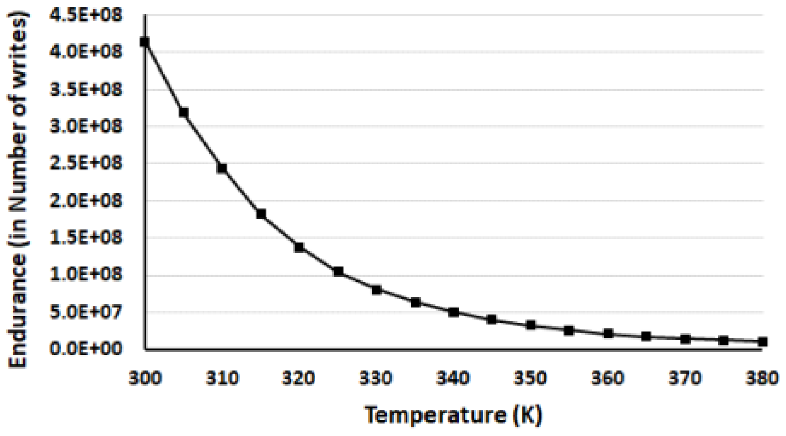


Figure 1.12: Trade-off between endurance and temperature of ReRAM [51]. The endurance of ReRAM cell drops by 0.026× as temperature increases from $300 - 380°K$.

the processing unit's lifetime by half [50]. On the other hand, for emerging memory, i.e., ReRAM, the latency improves with increasing temperatures, but it degrades the endurance of the cells, as shown in Figure 1.12. As the automotive ECS is subjected to high thermal conditions, the operating temperature of the PUs and memory should be optimized to enhance lifetime reliability.

Providing only mechanical cooling for ECS design in AVs can not solve the thermal problems. The emerging 3D ICs are a promising alternative for improving the performance of AI applications [52–54], but they suffer from high thermal gradients [42]. The existing active and passive cooling techniques [55, 56] utilized in modern automotives are ineffective in managing the thermal problems of 3D ICs. Liquid cooling is suggested in the literature to control the thermal problems for 3D ICs [42, 57, 58], but it does not guarantee reliability and adequate functionality [59]. Moreover, the mechanical design of the cooling system requires thermal characteristics and impacts of the ECS for better design trade-offs. Furthermore, a pessimistic design of the cooling system can be extremely costly and infeasible with its large operating temperature range. Hence, thermal-aware resource management and/or protection of delicate neural network parameters across ECS platform can help to minimize the temperature impact along with mechanical cooling solutions.

The elevated thermal profile of the PUs and memory due to the harsh environment of AVs and the complexity of computations degrades their performance and lifetime reliability. Simultaneously deteriorating the AI applications' accuracy can lead to catastrophic events in the AVs.

1.4 Research Problem and Our Contributions

ECS design for AVs is a very complex problem. It requires consideration of heterogeneity of the PUs and vehicle control applications, along with strict real-time latency requirements, throughput, power, safety, and reliability constraints. The resource management and workload distribution play a crucial role in achieving high performance, power efficiency, thermal conditions, and hardware utilization for the applications running on the PUs, which will lead to higher system-wide lifetime reliability. As such, for a given automotive application, we seek to design and develop efficient and effective resource management strategies on the heterogeneous distributed ECS architecture such that design constraints like timing, power consumption, peak temperature can be satisfied, and other criteria such as system-level lifetime reliability, Quality of Service (QoS), etc., can be optimized.

The major contributions of this dissertation are summarized as follows:

1. We study the problem of mapping periodic distributed automotive applications on a heterogeneous architecture to meet the temperature constraint, minimize the latency and temperature-induced system-level reliability degradation. The key challenges are effective and compute efficient temperature bounding as well as lifetime reliability estimation frameworks. In our approach, first, we propose a mathematical programming model to bound the peak temperature for the periodic automotive applications using an existing method [60], which is found to be a very pessimistic bound. Second, we prove a series of lemmas and theorems to find the effective and time-efficient framework to bound the peak temperature by considering the variance of vehicle applications' execution time. Next, we propose two compute-efficient system-level lifetime reliability analytical methods. In the first method, all the PUs of ECS are assumed to have the same degradation rate, while, in another method, we consider different degradation rates. We use a genetic algorithm to minimize the application latency, peak temperature estimated by the proposed framework, and maximize the system-level reliability using proposed models. The developed peak temperature algorithm is more accurate by $17-40°C$ than the existing method [60, 61] for practical benchmarks. Based on both synthesized test cases and practical benchmarks, the experimental results show that the proposed system-level reliability estimation approach can achieve $54\times$ to $110\times$ speed-up over the state-of-the-art approach [62] for design space exploration.
2. The temperature impacts to ECS can be substantial due to environmental conditions and the growing complexity of the PUs along with memory re-

quirements, which threatens to degrade the DNN classification accuracy significantly and even cause an AV to fail catastrophically. The challenge is how to quantify the thermal impact in a quantitative way for DNNs, which may demonstrate drastically different architectures and meta parameters. First, using the emerging memory, i.e., STT-MRAM, as a case study, we find that when DNNs are stored in it, their baseline accuracy can drop to as low as below 10% when the temperature rises above $110°C$, as a result of data bit flipping error in memory due to temperature variation. Then, we develop an analytical method to quantify the sensitivity of neurons under thermal impact. We validate the sensitivity analysis with standard datasets and DNNs. Our experiment results for LeNet, ResNet20, DenseNet40, and ResNet56 improve the accuracy by 0.5-4.2% than the state-of-the-art sensitivity method [63]. Further, we present the limitation of the proposed sensitivity framework when the error is large for other DNNs, such as the MobileNet.

3. We study the problem of mapping neuron parameters to memory systems when incorporating thermal impacts in designing DNN applications for ECS. We first study the thermal aware mapping problem of DNN parameters to memory banks with different temperature profiles. Based on the quantitative sensitivity metric for DNN neurons we developed before, we develop a bin-packing-based approach to map neuron parameters to memory banks with different temperatures with the goal to maximize the DNN prediction accuracy. The experiment results demonstrate that the thermal aware mapping improves the accuracy by 0.18-47.91% than the thermal ignorant approach for a range of common DNNs, such as LeNet, Conv6, ResNet20, DenseNet40, and ResNet56 networks. We also study the problem of how to identify the optimal temperature profiles in terms of maximizing the prediction accuracy when im-

plementing DNN applications with the same energy consumption. Here, a key challenge is the computational complexity of the mapping problem due to its NP-hard nature. In our approach, we develop the method by combining the traditional bin packing algorithm with convex optimization. From experiment results, we find that the developed method performs well at low temperatures, while it shows inconsistencies at high temperatures due to large discrepancies resulting from the sensitivity analysis. Furthermore, we also find that the traditional approach to having temperature distributed uniformly across the memory system does not seem to be an optimal solution when mapping neuron parameters to memory.

1.5 Structure of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we introduce several pertinent preliminary models used in this dissertation. We present a mathematical programming method and another framework to bound the peak temperature for periodic automotive applications in Chapter 3. This chapter also presents two system-level lifetime reliability methods for ECS. In Chapter 4, we present the neuron-level sensitivity analysis method along with its limitation. We present the importance of thermal awareness while mapping DNN on memory banks and present a convex optimization-based heuristic to find the optimal temperature profile of the memory banks to maximize the AI application classification accuracies in Chapter 5. Finally, in Chapter 6, we discuss the conclusions of this dissertation and present possible future works.

CHAPTER 2

PRELIMINARIES

This chapter covers the background and pertinent preliminaries of our research. We first introduce the heterogeneous architecture of the ECS and behavioral model of automotive applications. Then, we introduce power and thermal models in detail. Afterward, we present the lifetime reliability model used in this research. At last, we introduce the deep neural network architecture.

2.1 Heterogeneous Architecture and Behavioral Model

In this dissertation, we assume integrating multiple processing units (PUs) as an ECS, which can be Central Processing Units (CPUs), Graphical Processing Units (GPUs), Field Programmable Gate Arrays, and Processing-In-Memory (PIM) accelerators. The safety-critical automotive applications will be executed on the heterogeneous distributed architecture of the ECS.

We assume an ECS consists n_c heterogeneous PUs interconnected by a network of C buses through a central gateway [64,65] as $\mathbf{PU} = \{PU_1, PU_2, ..., PU_{n_c}\}$, as shown in Figure 2.1. Each PU can be interfaced with various sensors and actuators required for vehicle control operations. For example, the imaging sensor interfaced with PU_1 scans the vehicle surrounding periodically. The captured images are sent to PU_2 through a communication bus to classify the objects from captured images, and the brake control signal can be sent to PU_3 if required in real-time [66]. The ambient temperature of the PU is a cumulative effect of the environmental temperature and the surrounding electronics/mechanical devices heat transfer. We assume different ambient temperatures for each PU, $\mathbf{T_{amb}} = \{T_{amb_1}, T_{amb_2}, ..., T_{amb_{n_c}}\}$, as the PUs

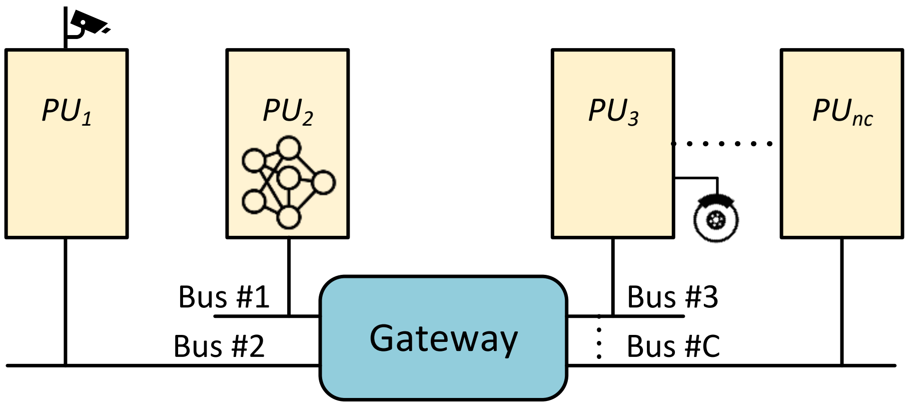


Figure 2.1: Heterogeneous ECS Architecture. Different processing units communicate through a network of buses interconnected by the central Gateway.

have different ambient temperatures according to their mounting location in the vehicle [38].

The automotive tasks follow sequential behavior, where the current task execution depends on the data from the completion of the predecessor task. Hence, we use Directed Acyclic Graph (DAG), $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ to model the automotive applications, which will be mapped on the heterogeneous ECS architecture. We assume that the node-set $\mathcal{V}$ consists of total n_v nodes for sensing/control tasks in an ECS as $\mathcal{V} = \{\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_{n_v}\}$. Each task may require different computation times due to the heterogeneity of the PUs. Therefore, we construct a 2-D matrix for worst-case execution time (WCET) of the task set on different PUs as,

$$\mathbf{W}_{n_v \times n_c} = \{w_{ik}, i = 1, ..., n_v;\ k = 1, ..., n_c\}, \tag{2.1}$$

where w_{ik} represents the worst-case execution time of the i-th task ($\mathcal{V}_i$) executed on the k-th PU (PU_k). We assume that the overhead for administering task executions can be reckoned by calibrating each task's execution times, and the WCET of the tasks is not affected by the thermal profile. The edge set $\mathcal{E} = \{e_{ij}$, with $\mathcal{V}_i, \mathcal{V}_j \in \mathcal{V}\}$ represents the worst-case communication cost between different task nodes in a system.

2.2 Power and Thermal Models

The power consumption of a CMOS IC chip is an aggregate of dynamic power (P_d) and leakage power (P_s)

$$P_{total} = P_d + P_s. \tag{2.2}$$

The dynamic power depends on switching activities of the underlying transistors in a CMOS IC, which depends on the supply voltage (V_{dd}) and operating frequency (f_r) as

$$P_d = C_s \cdot V_{dd}^2 \cdot f_r, \tag{2.3}$$

where C_s is the switching capacitance. The leakage power is observed in the absence of switching activity and represented as [67]

$$P_s = N_g \cdot I_l \cdot V_{dd}, \tag{2.4}$$

where N_g is the number of gates in an IC, and I_l is the leakage current responsible for leakage power given as [67]

$$I_l = I_r(A_l \cdot T^2 \cdot e^{((\alpha_l V_{dd} + \beta_l)/T)} + B_l \cdot e^{(\gamma_l V_{dd} + \delta_l)}), \tag{2.5}$$

where I_r is the reference leakage current, T is the temperature, and $A_l,\ B_l,\ \alpha_l,\ \beta_l,\ \gamma_l,\ \delta_l$ are technology-dependent constants. As the leakage current depends on T and V_{dd}, the correlation between leakage power and temperature can be imprecisely considered as a linear behavior [68]

$$P_s = \phi_1 \cdot T + \phi_0, \tag{2.6}$$

where ϕ_0 and ϕ_1 are technology-dependent constants. Therefore, the total power is

$$P_{total} = C_s \cdot V_{dd}^2 \cdot f_r + \phi_1 \cdot T + \phi_0. \tag{2.7}$$

Equation (2.7) models the power consumption of the processing unit.

We consider a thermal model similar to [61,68–73], in which thermal phenomenon is represented as an R-C lumped circuit. The thermal model for a PU with 4 processing cores is shown in Figure 2.2. Let P_m, R_{mm}, and C_m represent the power ($Watt$), thermal-resistance ($Joule/^{\circ}C$), and thermal-capacitance ($W/^{\circ}C$) for the processing core m. Whereas R_{mn} is the thermal-resistance between m^{th} and n^{th} processing cores, respectively, to consider the heat transfer effect among neighboring processing cores. Let T_{a_m} denote the ambient temperature for processing core m, then the thermal behavior of the core at time instant t be

$$C_m \cdot \frac{dT_m(t)}{dt} + \frac{T_m(t) - T_n(t)}{R_{mm}} + \sum_{n \neq m} \frac{T_m(t) - T_n(t)}{R_{mn}} = P_m(t). \tag{2.8}$$

Let $\delta_m = \frac{T_{a_m}}{Rmm}$ and

$$g_{mn} = \begin{cases} \sum_{n=1}^{M} \frac{1}{R_{mn}}, & \text{if n=m} \\ \frac{-1}{R_{mn}}, & \text{otherwise.} \end{cases} \tag{2.9}$$

Then we have

$$C_m \cdot \frac{dT_m(t)}{dt} + \sum_{n=1}^{M} g_{mn} \cdot T_n(t) = P_m(t) + \delta_m. \tag{2.10}$$

Similarly, the thermal model for the entire system is

$$\mathbf{C}_p \frac{d\mathbf{T}(t)}{dt} + \boldsymbol{g}_p \mathbf{T}(t) = \mathbf{P}_{total}(t) + \boldsymbol{\delta}_p, \tag{2.11}$$

where

$$\mathbf{C}_p = \begin{bmatrix} C_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & C_M \end{bmatrix} \boldsymbol{g}_p = \begin{bmatrix} g_{11} & \cdots & g_{1M} \\ \vdots & \ddots & \vdots \\ g_{M1} & \cdots & g_{MM} \end{bmatrix} \boldsymbol{\delta}_p = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_M \end{bmatrix}.$$

The parameters $\mathbf{C}_p$, $\boldsymbol{g}_p$, and $\boldsymbol{\delta}_p$ depend on the architecture of the processing units. We can find transient and stable state temperatures using equation (2.7) in equation (2.11), which are required for the thermal-aware reliability design of the ECS.

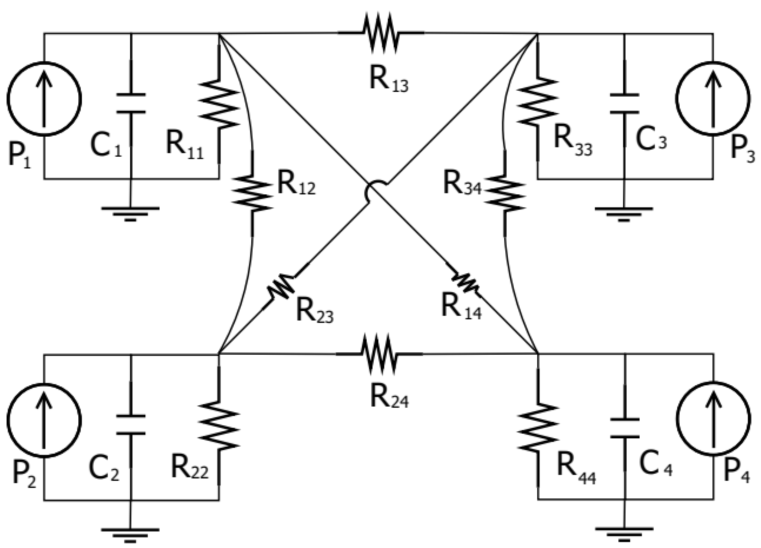


Figure 2.2: Thermal model for 4-core processing unit [74]. Each processing core is represented by the equivalent power source, thermal resistance, and thermal capacitance.

2.3 Lifetime Reliability Model

The reliability of a component is the probability of the component performing its intended operation for a specific time duration. Usually, reliability is characterized as a mean-time-to-failure (MTTF) [75]. If X be the lifetime of the component and $f_p(t)$ be the probability density function (PDF), then its reliability is

$$\mathbb{R}(t) = P(X > t) = \int_t^\infty f_p(t)dt. \tag{2.12}$$

Hence, $\mathbb{R}'(t) = -f_p(t)$. Then, MTTF of the component is specified as

$$MTTF = \int_0^\infty t \cdot f_p(t)dt = -\int_0^\infty t \cdot \mathbb{R}'(t)dt. \tag{2.13}$$

After solving the integration, we have

$$MTTF = -t \cdot \mathbb{R}(t)|_0^\infty + \int_0^\infty \mathbb{R}(t)dt. \tag{2.14}$$

Finally, we get

$$MTTF = \int_0^\infty \mathbb{R}(t)dt. \tag{2.15}$$

According to equation (2.15), we can estimate the MTTF if the reliability of the component is available.

Failures in the processing units are classified as hard and soft faults. The permanent and unrepairable damage to the IC chips is caused by hard faults, whereas latter faults can be recovered from temporary failures. The temperature-dependent hard failures in ICs are time-dependent dielectric breakdown, thermal cycling, hot carrier injection, negative bias temperature instability, and electromigration (EM). In this dissertation, we focus on the electromigration phenomenon to characterize the thermal-aware system-level MTTF of the automotive ECS. However, our framework can be readily used for other hard failures.

At the higher temperature and current density, the electromigration causes displacement of the mass in the conductors of inadequate cross-section, leading to the vacancies in the chip geometry, which can not be recovered. This can be correlated with the mean-time-to-failure as [76],

$$MTTF(T) \propto A_c \cdot J^{-n} \cdot e^{\frac{E_a}{K \cdot T}}, \tag{2.16}$$

where A_c is a cross-section area of the conductor-related constant; J is the current density (Amp/cm^2); E_a is the activation energy (*electron-volts*); K is the Boltzmann's constant; T is the temperature (*Kelvin*), and $n = 2$ unless otherwise specified. From equation (2.16), the lifetime reliability of the device is inversely dependent on the peak temperature, and it would be improved if the peak temperature is reduced.

2.4 Deep Neural Network Architecture

The deep neural networks are used in AVs as they mimic the human brain's functionality of taking complex decisions on the road. A neuron is the fundamental

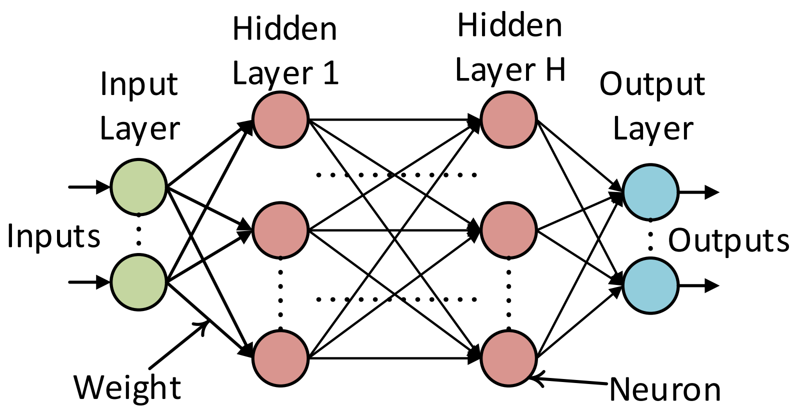


Figure 2.3: Feed Forward Neural Network containing input layer, multiple hidden layers, and an output layer.

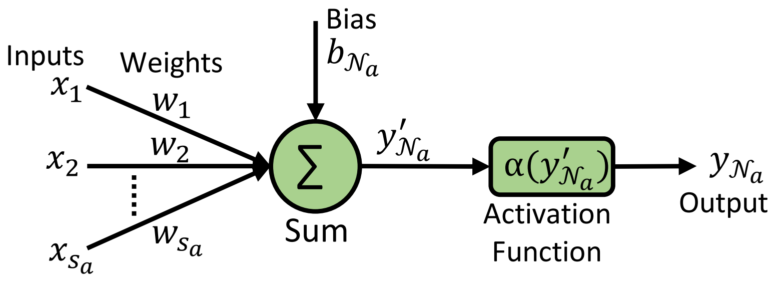


Figure 2.4: Structure of the Neuron. All inputs of the neuron are associated with trainable network parameters.

block of the deep neural network, arranged in different layers as shown in Figure 2.3. Each layer in the figure has multiple neurons, and the layers are categorized as the input layer, hidden layers $(1, ..., H)$, and an output layer [77]. External input is pre-processed and applied to the input layer. The outputs of the input layer are fed to the number of hidden layers to extract features from the input data. The number of neurons in a layer and the number of hidden layers in a network are application-dependent, required to extricate attributes from the input. The neurons are interconnected through the links known as weights or parameters, and the structure of the neuron is shown in Figure 2.4.

Let the DNN parameters be $\boldsymbol{W} \in \mathbb{R}^N$, where N is the size of the parameter space. Let the neuron set be $\boldsymbol{\mathcal{N}} = \{\mathcal{N}_1, \mathcal{N}_2, ...\mathcal{N}_{n_e}\}$, where the DNN has n_e neurons with sizes $s_1, s_2, ..., s_{n_e}$ respectively, such that, $s_1 + s_2 + ... + s_{n_e} = N$. The neuron shown in Figure 2.4 has connection weights $\boldsymbol{W}_{\mathcal{N}_a} = \{w_1, w_2, ...w_{s_a}\}$ from the previous layer neuron outputs $(x_1, x_2, ..., x_{s_a})$ and the *bias* $(b_{\mathcal{N}_a})$. It performs weighted summation of the input feature maps:

$$y'_{\mathcal{N}_a} = \sum_{i=1}^{s_a} x_i \; w_i + b_{\mathcal{N}_a}. \tag{2.17}$$

In this dissertation, we refer to weights and bias as parameters if not explicitly specified. The weighted sum (y'_{z_a}), also known as the activation, is passed through a nonlinear activation function. Finally, we get the output of the neuron as

$$y_{\mathcal{N}_a} = \alpha(y'_{\mathcal{N}_a}). \tag{2.18}$$

The nonlinear activation functions such as Sigmoid, Tanh, and ReLU are commonly employed in neural networks [78]. The activations pass layer-by-layer through the network sequentially and reach the output layer.

2.5 Summary

In this chapter, we present several pertinent preliminary models and architectures for our research. We first introduce the distributed heterogeneous architecture of the processing units in the ECS of the AVs. Later, discuss the automotive application behavior as a directed acyclic graph. Then, we present the thermal and power models required for considering the thermal-aware behavior of the PUs. Since temperature affects the reliability, we present electromigration-based lifetime reliability of the PU in terms of the MTTF metric. Afterward, we discuss the basic deep neural network architecture and associated neuron terminology.

In this dissertation, our research aims to optimize the automotive mission-critical latency and system-level lifetime reliability of the ECS design by satisfying thermal constraints, simultaneously maximizing the accuracy of the AI applications under thermal variations. In the following chapter, i.e., Chapter 3, we present our contributions on latency, system-level lifetime reliability, and thermal constraint design.

CHAPTER 3

THERMAL AWARE LIFETIME RELIABILITY OPTIMIZATION FOR AUTOMOTIVE DISTRIBUTED COMPUTING APPLICATIONS

As the automotive industry is shifting the paradigm towards autonomous driving, safety guarantee has become a paramount consideration in design. Temperature plays a crucial role in the system-wide reliability of the ECS used in the automotive. A vehicle is usually subjected to harsh temperature conditions from its operating environment. The increasing power density of ICs in the ECS further exacerbates the operating temperature and thermal gradient condition on the chip, thereby significantly impacting the vehicle's reliability. In this chapter, we study how to map a periodic distributed automotive application on a heterogeneous multiple-core processing architecture with temperature and system-level reliability issues in check. We first present a mathematical programming model to bound the peak operating temperature for the ECS. Then we propose a sophisticated method using genetic algorithm to effectively bound the peak temperature and optimize the system-wide reliability of the ECS by maximizing its MTTF. We present an algorithm to guarantee the peak temperature for periodic applications with variable execution times to ensure our approach's effectiveness. We present several computationally efficient techniques for system-wide MTTF computation, which show several-order-of-magnitude speed-up over the state-of-the-art method when tested using synthetic cases and practical benchmarks.

3.1 Introduction

The mainstream innovation in the automotive industry is driven by electronic systems that have transformed automobiles from a mechanical-only system to a sophisticated network of embedded systems. Today, roughly 1/3 of a car's cost is for automobile electronics, and the Electronic Control Units' costs can reach 50% to 60% in self-driving cars. To this end, the global Electronic Control Unit (ECU) market is continuing to grow by approx. 24% from 2020 to 2024 [79]. Utilizing innovative electronics technologies provides safer integration systems, which reduce human failure in driving, e.g., from traditional GPS positioning to inter-vehicle communication, from the traditional CAN bus to Automotive Ethernet and Media Oriented Serial Transport [80]. Meanwhile, consolidating auto-electronics and enhancing ECS reliability becomes more important as endorsed by ISO-26262 standard [81] because more and more life-critical control decisions are made electronically on the roads.

The automotive callbacks had tripled in the previous decade due to critical safety failures and faulty components [82]. Notably, in 2018, electronic defects have led to the highest percentage of vehicle recalls [83], and temperature plays a key role in terms of reliability for automotive systems. Among various failure mechanisms in an automotive ECS, the temperature-induced fault can reach as high as 55% [40]. It is because the aggressive scaling of transistor sizes soars the chip power density and runtime temperature, and the ever-increasing computing complexity dramatically exaggerates the chip thermal gradient, which could harm the system reliability [84]. Moreover, the vehicles undergo stringent environmental conditions with the temperature of ECUs ranging from $90°C$ to $155°C$ [38]. As a result, it is subjected to rapid aging due to temperature-induced failure phenomena such as electromigration (EM), time-dependent dielectric breakdown, thermal cycling, negative bias temperature instability, and hot carrier injection [85]. Also, exceeding the temperature threshold can automatically shut off the ECU [45] during runtime and may cause catastrophic consequences due to latency violations of the control decisions. Therefore, the runtime temperature should be carefully managed for reliability concerns to meet the safety requirements for the automotive [81].

In addition, to runtime thermal solutions, automotives use various active and passive cooling methods to restrict the ECU temperature, such as convection heat sinks and spreaders in low-end ECUs [55] or passive cooling for high-end ECUs, e.g., Tesla Model 3 [56]. However, for such systems, it requires non-trivial efforts to tightly bound the peak temperature for effective cooling system design. As shown later in the chapter, our proposed algorithm (in Section 3.5.2) can tighten the peak temperature bound by $15°C - 20°C$ compared with the *thermal ignorant approach*, resulting in cost, area, and energy savings of the cooling system design.

In this chapter, we study the problem of mapping periodic distributed automotive applications on a heterogeneous ECU architecture to mitigate thermal-induced system-level reliability degradations. In particular, we aim at electromigration-aware MTTF maximization and latency minimization without violating the temperature threshold on distributed heterogeneous ECUs. Our proposed optimization framework can also be readily utilized to improve other failure types. In our approach, we first present a mathematical programming model to bound the peak operating temperature by assuming the automotive ECUs reach their thermal stable-status immediately [60,61]. Then, we propose a method using the genetic algorithm to effectively bound the peak temperature and optimize the system-level reliability of the ECS by maximizing its MTTF. We present a computationally efficient approach to guarantee the peak temperature for periodic applications with variable execution times to ensure our approach's effectiveness. We also present novel computationally efficient system-level MTTF formulas, which show several-order-of-magnitude speed-ups over the state-of-the-art method. Overall, we have made the following contributions:

1. A simple thermal-aware mathematical programming-based method is proposed to bound the peak temperature when mapping periodic vehicle applications on a heterogeneous ECS architecture.
2. We find that existing approaches cannot safely bound the peak temperature when tasks' actual execution times vary from their Worst-Case Execution Times (WCETs). To this end, we develop a computation-efficient temperature bounding approach that can be safely adopted in practical cases with execution time variance. Further, we formally prove a series of supportive lemmas and theorems in this work to ensure its effectiveness.
3. We improve the system-level MTTF computing efficiency. Our proposed formula can be used for problems with higher design complexity and more optimization dimensions than [62, 86]. Further, we extend our MTTF calculation approach to the more general case (when different ECUs may have different aging rates).
4. We incorporate our thermal bounding method and system-level MTTF estimation into a genetic algorithm-based approach to map the periodic vehicle applications on ECS. Based on both synthesized test cases and real-life benchmarks, the experimental results show that the proposed approach can achieve $54\times$ to $110\times$ speed-up over the state-of-the-art approach [62].

3.2 Related Work

An automotive ECS carries safety-critical applications, which demands both stringent real-time responsiveness and high-reliability requirements. To this end, first of all, an ECS runtime management policy must guarantee its thermal safeness and tolerate harsh thermal environments. Meanwhile, thermal safeness reduces the risk of real-time violation and reliability degradation. In this chapter, we improve the optimization framework that can simultaneously deal with ECS temperature limitations, real-time constraints, and reliability requirements.

The thermal safeness ensured in this work is essential for achieving a sustainable running ECS, which prevents automatically triggered power gating, clock throttling, or shutting down PUs for thermal protection. Some reliability optimization approaches have been proposed, but they did not consider the thermal threshold of the hardware, e.g. [64, 65, 87–90]. So, their policies are not always feasible in thermal-constrained platforms. For example, Huang et al. [91] optimized the energy, reliability, and makespan jointly on directed acyclic graphs (DAGs); however, they did not consider the thermal effects on frequency-dependent transient faults in the reliability framework. Other existing approaches employ simplified power/thermal models, which can either cause thermal threshold violation during runtime or pessimistically predict the system performance. For example, Lee et al. [71] utilized constant maximum powers for peak temperature identification on consecutive execution phases, so its predicted peak temperature can overly constrain its actual resource utilization. Besides, the solution in [71] does not optimize the reliability.

To facilitate more rigorous analytical thermal analyses, some more sophisticated algorithms have been developed to identify the peak temperature [92, 93]. However, their computational cost can be prohibitively high when scaling the application complexity and processing platforms [92–94]. To reduce the peak temperature identification complexity, thermal bounding approaches [61, 95] were developed to estimate the highest possible peak temperature for given task sets with different mappings. However, as shown in a later section of this chapter, these approaches [61, 95] cannot effectively bound the peak temperature when the task execution time varies from their worst-case execution time. Other works, e.g. [96, 97], use the regression-based model and practical set-up, respectively, to estimate the peak chip temperature. These solutions cannot guarantee that the actual peak temperature stays below the estimated results and are also difficult to be incorporated into an optimization framework. There is a need to develop more effective and efficient temperature-constrained methodologies that can be safely applied on ECS with additional optimization factors, such as reliability enhancement and makespan minimization.

The aforementioned temperature estimation methodologies are of utmost importance to avoid thermal-induced failure and system reliability degradation. Also, the maximally allowed temperature limits transient throughput, which increases the design complexity in the reliability-makespan trade-off. To this end, many works consider temperature when establishing reliability optimization frameworks [98]. When incorporating the temperature dynamics into reliability optimization work, Zhou et al. [98] presented mixed-integer linear programming (MILP) method that can exponentially increase the computational cost. Ergun et al. [99] maximized the system reliability on IoT systems without considering the timing constraints; thus, their solutions can be infeasible on the latency-critical automotive applications. These methodologies fall short of effectively dealing with the thermal challenges and their impacts on system reliability (such as the system lifetime reliability, e.g., Mean-Time-To-Failure) and fulfill critical PU requirements in automobile system design.

To incorporate system reliability optimization into our work, we adopt the widely used Mean-Time-To-Failure (MTTF) to determine a system's lifetime reliability and develop a fault-tolerant mechanism based on processor/system state [100]. A recent study extended the validation scope of MTTF from Electromigration to Thermomigration and stress-migration induced failure [101]. However, one primary concern is that the computational cost of MTTF calculation can be prohibitively high [62, 86, 102, 103] to obtain high accuracy. We presented a fast MTTF estimation technique in one of our earlier works [39] based on uniform wear-out rates to reduce the computational cost. As indicated in [38], due to the heterogeneity of ECS and different operating conditions, there is a need to consider that different ECS components have different failure rates caused by different PU temperatures, different processing elements, and different mounting locations.

The rest of the chapter is organized as follows. We introduce the system models and formally define the problem in Section 3.3. In Section 3.4, a linear mathematical programming model for a simple thermal approach is presented. In Section 3.5, a more accurate peak temperature bounding method is proposed. Section 3.6 presents computationally efficient system-level MTTF schemes for the PUs with (1) the same wear-out rate and (2) different wear-out rates. In Section 3.7, our genetic algorithm details, experiment set-up, and results are described. At last, we conclude in Section 3.8.

3.3 Preliminary

In this section, we first discuss the architecture and system models used in this chapter. Later, we define the thermal and system-level reliability models.

3.3.1 Architecture and Application Models

In a cost-sensitive industry, the face value of the PU hardware is reduced by optimizing the memory and computational power, and automobile manufacturers import such PUs for various functionalities from different vendors integrating them into a system [104]. The PUs can be broadly categorized as (1) Commodity PUs for simple functions; (2) Integrated PUs realizing safety and latency-critical applications; (3) Computing platform PUs act as a little computing cluster in a car [105]. In this study, we consider integrating multiple PUs as an ECS targeting safety-critical applications. The heterogeneous architecture and application model is described in Section 2.1.

3.3.2 Power and Thermal Models

The total power consumption of a PU is a combination of dynamic power (P_d) and leakage power (P_s). The dynamic power does not depend on the transient temperature [68], but each task in an automotive system has different dynamic power based on the switching activity [71]. Therefore, to exploit this phenomenon, we consider the dynamic power of task $\mathcal{V}_i$ as $P_d(i) = \alpha_i.V_{dd}^3$. Where α_i is the activity factor with $\alpha_i = 0$ for the idle task, and V_{dd} is the supply voltage of the PU.

We assume that the leakage power of a PU is linearly dependent on the thermal state [68], i.e., $P_s = (\phi_1.T + \phi_0)$, where T is the temperature of the PU, ϕ_0, and ϕ_1 are PU-dependent constants. Therefore, the all-inclusive total power consumption (P_{total}) of a PU while running the task $\mathcal{V}_i$ can be expressed as

$$P_{total} = P_d(i) + P_s = V_{dd}^3 \cdot \alpha_i + (\phi_1 \cdot T + \phi_0). \tag{3.1}$$

We adopt the widely used RC-thermal model [61, 68, 71–73] to capture the PU temperature dynamics. $T(t)$ and $P(t)$ are the transient temperature ($^\circ C$) and its corresponding power consumption ($Watt$) at time instant t, respectively. Then, the transient temperature $T(t)$ follows

$$R_{th} \cdot C_{th} \cdot \frac{dT(t)}{dt} + T(t) - R_{th} \cdot P_{total}(t) = T_{amb_k}, \tag{3.2}$$

where R_{th}, C_{th} represent thermal-resistance ($°C/W$) and thermal-capacitance ($J/°C$).

Given the equation (3.2), for task $\mathcal{V}_i$, we can easily identify its ending temperature $T(t_2)$ of the interval $[t_1, t_2]$ with $T(t_1)$ being the initial temperature as

$$T(t_2) = \frac{A(i)}{B} + \left(T(t_1) - T_{amb_k} - \frac{A(i)}{B}\right) e^{-B(t_2 - t_1)} + T_{amb_k}, \tag{3.3}$$

where $A(i) = (\phi_0 + V_{dd}^3 \cdot \alpha_i)/C_{th}$ and $B = 1/(R_{th} \cdot C_{th}) - \phi_1/C_{th}$. If the PU executes the periodic task profile, then the stable status temperature of the PU can be given by the following theorem (Similar proof is described in Quan et al. [68] and hence omitted.)

Theorem 3.3.1. *Let a processing unit, i.e., PU_k runs a periodic schedule with the period of t_p, starting from the ambient temperature (T_{amb_k}). Let T_L be the ending temperature of the first period and let T_{ss} be the temperature when it reaches a stable status. Then,*

$$T_{ss} = T_{amb_k} + \frac{T_L - T_{amb_k}}{1 - \mathbb{K}}, \tag{3.4}$$

where $\mathbb{K} = exp(-B \cdot t_p)$.

3.3.3 Lifetime Reliability Model

In this chapter, we consider electromigration-based permanent faults in the ICs. At higher temperatures and current densities, electromigration produces mass displacement in conductors with insufficient cross-sections, resulting in voids in the chip design that cannot be filled. The detailed model is described in Section 2.3.

3.3.4 Problem Formulation

For the given automotive DAG application set $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, we seek mapping strategies on the heterogeneous architecture **PU** such that the latency is reduced (by satisfying the deadline) and system-level lifetime reliability is maximized. Simultaneously, we intend to optimize the peak temperature as (1) the temperature higher than the threshold limit could automatically trigger the thermal protection mechanism of a PU [61, 106], which may cause applications' deadline violations or even catastrophic consequences in an automotive control. (2) $10-15°C$ rise in the temperature accelerates wear-out rapidly and reduces the processor's lifetime by half [50].

Problem 3.3.2. *Given* **PU** *and DAG* $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$*, allocate all task nodes in* $\mathcal{G}$ *to* **PU** *such that (1) the peak temperature of the design (*T_{peak}*) is lower than the given temperature threshold (*T_{max}*); (2) the makespan (*C_{max}*), peak temperature, and system-level lifetime reliability (*$MTTF$*) of the design are optimized.*

3.4 Mathematical Programming Approach

Clearly, Problem 3.3.2 in Section 3.3.4 is NP-hard, and different approaches can be applied to solve it, such as mathematical programming [107], simulated annealing [62], genetic algorithm [108], convex optimization [85, 109], and analytical approach [68, 71]. In what follows, we incorporate the thermal impacts into the mathematical program as it guarantees the optimal solution.

To satisfy the peak temperature constraint of the **PU** after partitioning of the task set $\mathcal{V}$, we define the decision variables x_{ik} as,

$$x_{ik} = \begin{cases} 1, & \text{if } \mathcal{V}_i \text{ is assigned to } PU_k; \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

Each task $\mathcal{V}_i$ must be allocated to only one processor as constrained by,

$$\sum_{k=1}^{n_c} x_{ik} = 1, \quad \forall \mathcal{V}_i \in \mathcal{V}, \forall PU_k \in \mathbf{PU}. \tag{3.6}$$

Let us define another decision variable σ_i as the starting time of task $\mathcal{V}_i$, then the makespan of the system is defined to be,

$$\sigma_i + \sum_{k=1}^{n_c} w_{ik} \cdot x_{ik} \leq C_{max}, \quad \forall \mathcal{V}_i \in \mathcal{V}, \forall PU_k \in \mathbf{PU}. \tag{3.7}$$

In the meantime, if the predecessor-successor task pairs are allocated to different PUs, then the communication costs among them are managed by the following equation,

$$\sigma_i + w_{ik} \cdot x_{ik} + e_{ij} \cdot (x_{jl} + x_{ik} - 1) \leq \sigma_j, \tag{3.8}$$

where PU_k and $PU_l \in \mathbf{PU}, \mathcal{V}_i$ and $\mathcal{V}_j \in \mathcal{V}, \forall e_{ij} \in \mathcal{E}$. On the other hand, the following constraints ensure the executions of two tasks allocated on the same PU will never overlap,

$$\begin{gathered} \sigma_i + w_{ik} - \sigma_j \leq \mathcal{M} \cdot (2 - x_{ik} - x_{jk}) \\ \text{OR} \\ \sigma_j + w_{jk} - \sigma_i \leq \mathcal{M} \cdot (2 - x_{ik} - x_{jk}), \end{gathered} \tag{3.9}$$

where $\forall \mathcal{V}_i \neq \mathcal{V}_j \in \mathcal{V}$, $\forall PU_k \in \mathbf{PU}$, and $\mathcal{M}$ is a large positive constant.

Note that the constraints imposed by the equations (3.6 - 3.9) ensure the minimal makespan and task precedence, but they do not contemplate the thermal behavior in the mathematical model. From equations (3.2) and (3.3), we can observe that the temperature is a non-linear function of time (t), which is unsolvable by linear solvers. Therefore, we use a simple thermal approach in which each task has a constant stable status temperature irrespective of the starting temperature and thermal capacitance of the PU [60,61]. In particular, for any task $\mathcal{V}_j \in \mathcal{V}$, we can set $\frac{dT(t)}{dt} = 0$ in equation (3.2) and obtain the constant stable status temperature as $T = T_{amb_k} + R_{th} \cdot P_{total}$.

With this knowledge, we can find the stable status temperature for any $\mathcal{V}_i \in \mathcal{V}$ on PU_k as

$$T_k^*(i) = \frac{A(i)}{B} + T_{amb_k}, \quad \forall T_{amb_k} \in \mathbf{T_{amb}}. \tag{3.10}$$

Therefore, the highest system-wide temperature (denoted as T^*) can be formulated as

$$T^* = \max_{i,k}(T_k^*(i)), \quad \forall \mathcal{V}_i \text{ allocated on } PU_k. \tag{3.11}$$

Further, we have the following theorem to ensure that T^* bounds the peak temperature for any resultant task allocation.

Theorem 3.4.1. *For any mapping of $\mathcal{V}$ to $\mathbf{PU}$, with periodic execution, the resultant highest system temperature never exceeds T^*.*

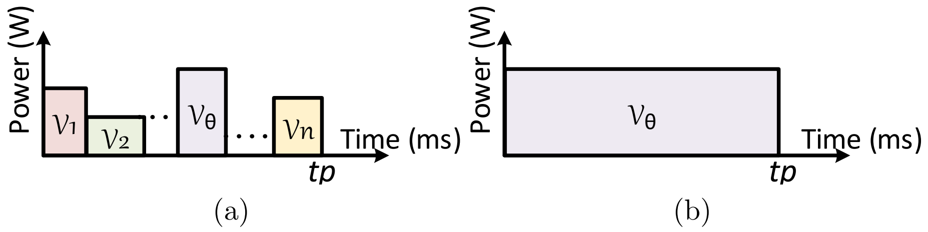


Figure 3.1: (a) The WCET schedule with period t_p. (b) A hypothetical schedule with only task $\mathcal{V}_\theta$ for the entire period t_p.

Proof: Suppose PU_k periodically executes n tasks $\{\mathcal{V}_1, ..., \mathcal{V}_n\}$ at a period t_p, whose dynamic powers are $\{P_d(1), ..., P_d(n)\}$ as shown in Figure 3.1a. Let task $\mathcal{V}_\theta$ consume the highest dynamic power among all tasks allocated on PU_k, i.e., $P_d(\theta) = max\{P_d(1), ..., P_d(n)\}$. Then, for all $\mathcal{V}_i$ allocated on PU_k, since $P_d(\theta) \geq P_d(i)$, we have

$$T_k^*(\theta) \geq T_k^*(i). \tag{3.12}$$

Using a hypothetical schedule that keeps at a constant power consumption $P_d(\theta)$ in one period as Figure 3.1b can bound the peak temperature of the schedule shown in Figure 3.1a because both the execution time and power consumption for all $\mathcal{V}_i$ are not larger than $\mathcal{V}_\theta$ on PU_k in Figure 3.1b. When applying equation (3.12) on all PUs, $T^* = \max_{i,k}(T_k^*(i))$ can effectively bound the system-wide peak temperature.

$\square$

With Theorem 3.4.1, we can readily add a peak temperature-related constraint as follows,

$$T^* \leq T_{max}. \tag{3.13}$$

Also, we need to incorporate the minimization of peak temperature into the design objective. Note that to reduce the makespan and to reduce the peak temperature are two conflicting design objectives. How to deal with multi-criteria optimization in a more sophisticated manner is not the focus of this chapter. Instead, we use a weighted sum as our optimization objective as follows,

$$Max : \mathbb{W}_1 \cdot \frac{C_s - C_{max}}{C_s} + \mathbb{W}_2 \cdot \frac{T_{max} - T^*}{T_{max}}, \tag{3.14}$$

where $\mathbb{W}_1, \mathbb{W}_2$ are the weights and $\mathbb{W}_1 + \mathbb{W}_2 = 1$. C_s is the minimal makespan that a task set can complete on a given ECS without considering its temperature constraint, reliability optimization, etc.

The mathematical programming approach presented in this section has three challenges. First, the proposed method is well known to be NP-hard in nature, and its computational complexity increases exponentially with the size of the task sets, number of PUs, and the complexity of the real-time schedules, etc. For a larger problem size, it could be infeasible to obtain a solution. Second, to identify the peak temperature, this approach adopts a strategy that assumes a PU can immediately reach its stable status [60]. This can lead to overly pessimistic results, especially when the task execution time is very short (from tens of microseconds to several milliseconds [61]). Third, we can't incorporate another optimization parameter, i.e., system-level lifetime reliability, in the mathematical programming framework due to the non-linearity between the task execution latency and its resulting temperature dynamics. Therefore, we resort to the meta-heuristic genetic algorithm [108] to solve Problem 3.3.2. In the following sections, we first study how to capture the peak temperature under a task mapping configuration, and then we develop efficient MTTF calculations accordingly.

3.5 Bound the Worst-case Peak Temperature

A key to solve Problem 3.3.2 in Section 3.3.4 is to estimate an accurate peak temperature for a given task/PU mapping configuration. Several methods for estimating a processor's peak temperature have been studied. For example, the numerical method breaks each execution time in tiny fragments and attempts to find the peak temperature by exploring each small stretch (e.g. [110–112]). Also, an epoch-based peak temperature detection methodology has been proposed using a mix of mathematical and numerical methods in [72] or by solving the first-order derivative on each processing core via the Newton-Raphson method in [73]. Another approach greedily searches the worst-case task arrival cases to bound the runtime peak temperature has been proposed in [93]. However, these approaches have very high computation complexity(e.g. [93, 110–112]), so they are challenging to be utilized during the exploration of design space.

For more computationally efficient approaches, besides the *simple thermal approach* stated in Section 3.4, Chaturvedi et al. [95] proposed a so-called "hypothetical step-up schedule" to bound the peak temperature for a periodic schedule. This approach takes advantage of the periodicity of tasks and can bound the peak temperature at a linear time complexity. However, as shown in what follows, this approach works under the assumptions of each task instance always takes its worst-case execution times. It may fail when the task's execution time varies in runtime, which is quite normal in practical automotive scenarios [113].

3.5.1 Motivation Examples

In real-time computing, it is a standard exercise to use the worst-case execution times to bound the longest completion time of a task set under a scheduling policy on a hardware platform. However, we find that using the worst-case execution time-based real-time schedules (e.g. [110–112]) can not always identify the highest possible peak temperature during runtime.

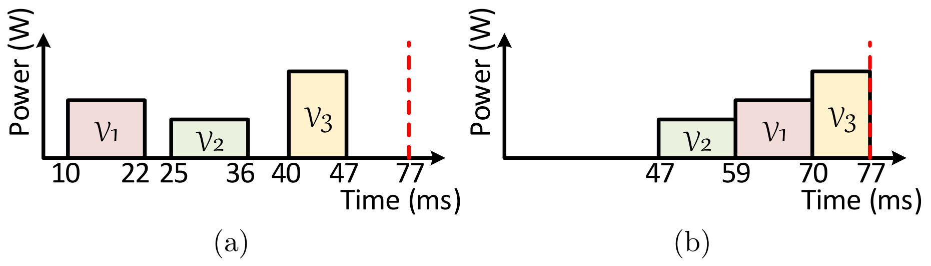


Figure 3.2: An example of the step-up schedule. (a) Original schedule. (b) All tasks in the given schedule are arranged in the increasing order of their power.

To put our discussions into perspective, assume a task set contains three periodic tasks with an identical period. One period of its schedule (based on the worst-case execution times) is depicted in Figure 3.2a. Figure 3.2b shows its corresponding *step-up* schedule, with the same interval lengths but organized so that the dynamic power consumptions are monotonically increasing from the first interval to the last. It is formally demonstrated that the peak temperature of a schedule is bounded by its corresponding "step-up" schedule [61, 95]. For example, if $w_{1k} = 12\ ms$, $w_{2k} = 11\ ms$, $w_{3k} = 7\ ms$ and $P_d(1) = 2.204\ W$, $P_d(2) = 0.962\ W$, $P_d(3) = 2.924\ W$ with a period 77 ms, using the experimental set-up in Section 3.7.1, the step-up schedule's peak temperature (Figure 3.2b) is $51.97°C$, which bounds the actual peak temperature of $51.69°C$ for the schedule in Figure 3.2a. It is much smaller than the peak temperature of $125.45°C$ using the simple peak temperature prediction approach of Section 3.4.

The example shown in Figure 3.2 demonstrates that the step-up schedule can guarantee peak temperature only when all the tasks run with their worst-case execution times (WCETs). However, the temperature bound given by a step-up schedule may be violated when the tasks run with execution times lower than their WCETs, as shown in the following motivational example.

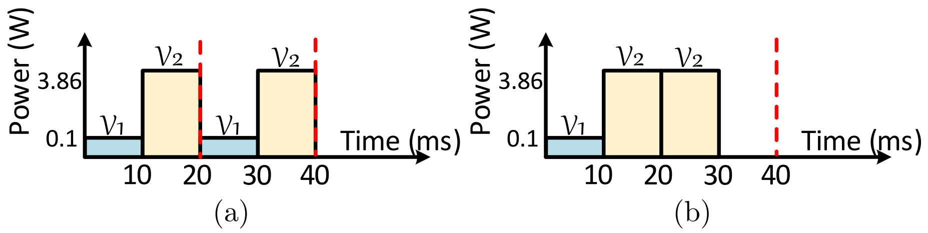


Figure 3.3: A motivational example. (a) Two periods of the step-up schedule of two tasks. (b) The modified schedule after the execution time of the task $\mathcal{V}_1$ is zero in the second period.

Figure 3.3a shows a step-up schedule with two tasks, $\mathcal{V}_1$ and $\mathcal{V}_2$. Assume the dynamic power consumptions and the WCETs for $\mathcal{V}_1$ ($\mathcal{V}_2$ resp.) are 100 mW (3.86 W resp.) and 10 ms (10 ms resp.), with a period of 20 ms. When tasks $\mathcal{V}_1$ and $\mathcal{V}_2$ take their WCETs as Figure 3.3a to reach their thermal steady state, the peak temperature is $93.61°C$. However, the actual task execution time may vary with the concurrent workload on one chip [114]. Without losing the generality, assume in a period of the stable status, task $\mathcal{V}_1$ changes its execution time between 0 and 10 ms, as shown in Figure 3.3b. Then, the highest peak temperature can reach $94.02°C$ in Figure 3.3b and violate the peak temperature predicted by Figure 3.3a. This motivation example demonstrates that the step-up schedule is unable to bound the peak temperature for periodic schedules under execution time variance. Although it is possible to greedily search all possibilities of execution time combinations (e.g., in [93]) for the highest possible temperature, the computational cost could be prohibitively high when applying to multiple-PU cases with tens to hundreds of applications. Therefore how to *efficiently* bound the worst-case peak temperature remains a problem.

Since the temperature dynamic in equation (3.2) is a function of the transient power consumption, execution time length, and other architecture-dependent variables, we further conduct a more thorough empirical study to see how different ranges of the dynamic power, execution time variance, and different numbers of tasks may lead to the inaccurate peak temperature prediction by its corresponding WCET-based step-up schedule. We consider a system of 3-PUs and increase the number of tasks from 10 to 60. The PU and task parameters are explained in Section 3.7.1, and we set the $T_{max} = 80°C$ [39]. We use the genetic algorithm described in Section 3.7.1 for task partitioning on the PU system with equation (3.39) as a fitness function, where the peak temperature bound for each PU configuration is set using the step-up schedule.

As shown by the number of infeasible cases in Table 3.1, the variable execution time can indeed cause the violation of the temperature threshold (T_{max}). For example, we observe that the larger the execution time variations are, the more likely the infeasible cases may occur. Moreover, we observe that the infeasible cases occur more frequently at the high-power range (e.g., $P_d(i) < 1.5W$) than at the low-power range (e.g., $P_d(i) < 0.5W$). The reason is that peak temperature is proportional to the highest transient power density. So, larger execution time variations and higher power ranges could change the temporal power density more significantly compared

Table 3.1: A case study for temperature bound's effectiveness using the step-up theory

Num. of Tasks	**Num. of infeasible cases**							**Total infeasible cases**
	$P_d(i) < 0.5$	$P_d(i) < 1$			$P_d(i) < 1.5$			
	♣	♣	♠	¶	♣	♠	¶	
10	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0
40	0	3	3	0	3	2	0	11
50	1	6	4	1	6	4	2	24
60	0	4	2	0	4	2	0	12

Note: The case study is conducted on a 3-PU system.
♣ a varied execution time ≈ 0 ms, ¶ a varied execution time $= 2$ ms,
♠ a varied execution time $= 1$ ms, $P_d(i)$ is measured in W for each task.

to the baseline WCET-based step-up schedule, which, in some cases, causes a higher peak temperature that exceeds the temperature limitations.

Note that the execution time variation is a statistical runtime phenomenon, so the execution time variance-induced thermal violation may not always happen in all cases. Its occurrence also depends on the concurrent tasks, resource contention, and power ranges, among other factors. Our work considers the life-critical real-time applications on automotive PUs, which require all applications to be completed by their deadlines while maintaining high reliability. To this end, our work ensures thermal safety and does not automatically trigger thermal protection in any cases during runtime, which hurts the throughput performance and causes real-time timing violations. In what follows, we propose a time-efficient algorithm to bound the peak temperature for variable execution time scenarios.

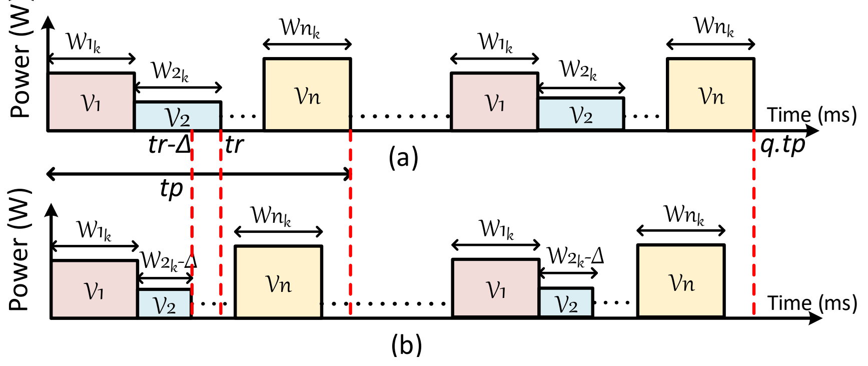


Figure 3.4: (a) The WCET schedule with an ending temperature of $T_m(q \cdot t_p)$ in the stable status. (b) The schedule with a varied execution time with an ending temperature of $T(q \cdot t_p)$ in the stable status.

3.5.2 Accurate peak temperature identification algorithm

We begin with several lemmas to support the algorithm that tightly bounds the peak temperature for variable execution time scenarios and later prove a theorem to validate the proposed algorithm's effectiveness.

Lemma 3.5.1. *Let PU_k run n tasks $\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_n$ consecutively with a t_p period. Let $T_m(q \cdot t_p)$ be the temperature at $t = q \cdot t_p$ when all tasks are executed with their WCETs. Then, we can infer $T_m(q \cdot t_p) \geq T(q \cdot t_p)$, if $T(q \cdot t_p)$ is the temperature at $t = q \cdot t_p$ when at least one task instance from $t \in [0, q \cdot t_p]$ runs with execution time less than its WCET.*

Proof: Let's consider an arbitrary schedule with n tasks and period t_p as shown in Fig. 3.4. Let $T_m(q \cdot t_p)$ and $T(q \cdot t_p)$ be the temperature at $t = q \cdot t_p$ in Fig. 3.4a and Fig. 3.4b, respectively. In Fig. 3.4a, all the tasks take their respective WCETs. In Fig. 3.4b, the execution time of task $\mathcal{V}_2$ is reduced by Δ compared to its WCET. Therefore, the starting time of the remaining tasks in the scheduling sequence remains the same or is shifted to the left by Δ amount of time. From these two figures, we indeed find that both schedules are identical up-to-time instant $(t_r - \Delta)$, so as their temperature, $T(t_r - \Delta) = T'(t_r - \Delta)$. Let $T(t_r)$ and $T'(t_r)$ be the starting temperatures for the remaining part of the schedules, respectively. The schedule in Fig. 3.4a has a higher accumulated computing time at $(q \cdot t_p)$ than the schedule in Fig. 3.4b, and the workload in Fig. 3.4b is moved away from $t = q \cdot t_p$. Based on *Lemma 3* of Schor et al. [115], we conclude that $T_m(q \cdot t_p) \geq T(q \cdot t_p)$. $\square$

Lemma 3.5.2. *Let PU_k run $\mathcal{V}_i$ during an interval $t \in [t_1, t_2]$ and let PU_k's temperature at t_1 be T_1. Then PU_k's temperature monotonically increases (decreases* resp.*) within interval t, if $T^*_k(i) \geq T_1$ ($T^*_k(i) \leq T_1$* resp.*).*

Proof: From equation (3.2), we can infer that when EC_k reaches its stable state temperature, equation $T^*_k(i) = R_{th} \cdot P_{total}(i) + T_{amb_k}$ holds, since $\frac{dT(t)}{dt} = 0$. From equation (3.2), we can also infer that

$$\begin{aligned} \frac{dT_1}{dt} &= \frac{-T_1 + R_{th} \cdot P_{total}(i) + T_{amb_k}}{R_{th} \cdot C_{th}} \\ &= \frac{-T_1 + T^*_k(i)}{R_{th} \cdot C_{th}}. \end{aligned} \tag{3.15}$$

Since $T^*_k(i) \geq T_1$, we have $\frac{dT_1}{dt} \geq 0$. Then, EC_k's temperature monotonically increases and vice-versa. $\square$

As per Lemma 3.5.1, if at least one of the tasks on PU executes shorter than its WCET, then the temperature at the epilogue of q-th period is no larger than its WCET counterpart. From Lemma 3.5.2, we deduce that if the individual task stable status temperature is higher (lower resp.) than the starting point temperature in an interval, the PU temperature monotonically increases (decreases resp.) when running in a constant execution mode during the interval.

With Lemma 3.5.1 and 3.5.2, an algorithm is developed to identify the peak temperature when a PU runs a set of tasks that considers task completion time variation during the runtime.

Algorithm 1 first calculates the stable status temperature at the end of the application period, assuming that each task takes its WCET (line 1). Then, lines 2-8

Algorithm 1 Bounding the peak temperature with execution time variance.

```
    Inputs: Task mapping based on WCET schedule of $PU_k$; Power/thermal parameters; Timing parameter matrix $\mathbf{W}_{n_v \times n_c}$;
    Output: Temperature bound $T_m$ of $PU_k$.
  Let $T_m = T_{ss}$ in Theorem 3.3.1 for the WCET schedule;
  for each task $\mathcal{V}_i$ allocated on $PU_k$ do
      Calculate $T_k^*(i)$ based on equation (3.10);
      if $T_k^*(i) > T_m$ then
          $T'_m$ = the ending temperature of running $\mathcal{V}_i$ with initial temperature $T_m$;
          $T_m = T'_m$;
      end if
  end for
  return $T_m$
```

iteratively identify the highest temperature of each interval in the schedule. Specifically, if a task's stable state temperature is more than the latest temperature bound (T_m), the task is taken into account to bound the possible higher peak temperature. Otherwise, the current interval is not considered in determining the peak temperature bound (T_m). Finally, the ending temperature for the last interval is output as the peak temperature (line 9). The complexity of Algorithm 1 is $O(n)$, where n is the number of tasks allocated to a PU. Also, we formulate Theorem 3.5.3 as follows to ensure the guarantee of the peak temperature identified through Algorithm 1. Note that Algorithm 1 is only used for bounding the runtime temperature. All tasks need to be executed according to their real-time schedules to satisfy their timing requirements.

Theorem 3.5.3. *Let tasks $\{\mathcal{V}_1, ..., \mathcal{V}_n\}$ be executed on PU PU_k periodically with variable execution times. Then, the highest possible temperature during the execution is no more than T_m output from Algorithm 1.*

Proof: Let a baseline schedule contain n tasks $\{\mathcal{V}_1, ..., \mathcal{V}_n\}$ with period t_p and each task takes its WCET, as shown in Fig. 3.5a. Without losing generality, we

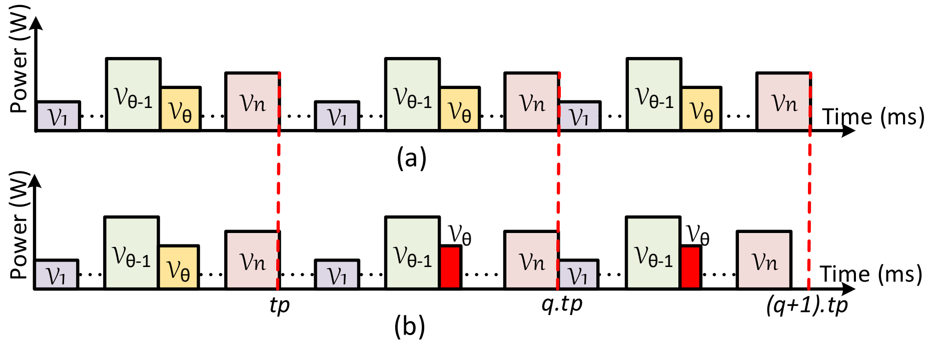


Figure 3.5: (a) The WCET schedule. (b) The schedule with a varied execution time of tasks $\mathcal{V}_\theta$ in a stable state.

assume Fig. 3.5b contains all the same intervals as Fig. 3.5a, except task $\mathcal{V}_\theta$ whose execution time is reduced by Δ in the q-th period.

Let $T_W(t)$ and $T_A(t)$ be the temperatures at the instance of t for Fig. 3.5a and Fig. 3.5b, respectively. Based on Lemma 3.5.1, at $t = (q \cdot t_p)$, we can infer that the temperature of Fig. 3.5b is no larger than Fig. 3.5a, so we have

$$T_W(q \cdot t_p) \geq T_A(q \cdot t_p). \tag{3.16}$$

Now, consider a period from $t = (q \cdot t_p)$ to $t = ((q+1) \cdot t_p)$. Starting from $t = (q \cdot t_p)$ let the temperature output based on Algorithm 1 be $\widetilde{T}_W(i)$ after going through task $\mathcal{V}_i$ in the loop (line 2-7) in Algorithm 1. Then, based on Lemma 3.5.2, $\widetilde{T}_W(i)$ must be equal or higher than the temperature when completing $\mathcal{V}_i$ with the schedule in Fig. 3.5b. Meanwhile, the starting temperature at each interval considered in Algorithm 1 is no lower than that of $T_A(q \cdot t_p)$. Therefore, $\widetilde{T}_W(i)$ monotonically increases to T_m in each valid interval as per Algorithm 1.

On the other hand, for the schedule in Fig. 3.5b, if task $\mathcal{V}_i$ has a stable state temperature lower than $T_A(q \cdot t_p)$, the temperature will be lowered. Therefore, we certainly find that at any time instant, the temperature predicted by Algorithm 1 (i.e., T_m) is higher than its periodic counterpart with varied execution times of the tasks. □

This theorem substantiates the effectiveness of the peak temperature output from the proposed Algorithm 1. With Algorithm 1 and Theorem 3.5.3 described, we next discuss our new methods to compute the system-level MTTF.

3.6 Computationally efficient system-level MTTF formulation

In this section, we first briefly introduce the relevant background for the state-of-the-art method that accurately calculates the system-level MTTF [62]. We then discuss the limitations of the method in [62] and propose our computationally efficient approaches to estimate the system-level MTTF.

Without losing the generality, we adopt the Weibull distribution to model the wear-out effects at the system level [62]. The reliability of a single PU at time instant t with temperature T is

$$\mathcal{R}(t,T) = e^{-\left(\frac{t}{\eta(T)}\right)^{\beta}}, \tag{3.17}$$

where $\eta(T)$ and β represent scale and slope parameters of the Weibull distribution, respectively. Then, we have

$$\eta(T) = \frac{MTTF(T)}{\Gamma\left(1+\frac{1}{\beta}\right)}. \tag{3.18}$$

When executing periodic tasks, the reliability of a PU in one period (t_p) is a function of the temperature and duration for each sub-interval ($a = 0, \cdots, s-1$) [62], which can be formulated as

$$\mathcal{R}(t_p) = e^{-\left(\sum\limits_{a=0}^{s-1}\frac{\Delta t_i}{\eta(T_i)}\right)^{\beta}}. \tag{3.19}$$

Let the aging factor of the PU [62] be

$$\mathcal{A} = \sum_{a=0}^{s-1}\frac{\Delta t_i}{\eta(T_i)}. \tag{3.20}$$

Then, the reliability of the PU for q consecutive periods is

$$\mathcal{R}(q \cdot t_p) = e^{-\left(\sum_{a=0}^{q\cdot(s-1)} \frac{\Delta t_a}{\eta(T_a)}\right)^{\beta}} = e^{-\left(q\cdot\sum_{a=0}^{s-1} \frac{\Delta t_a}{\eta(T_a)}\right)^{\beta}} = e^{-(q\cdot\mathcal{A})^{\beta}}.$$

Therefore, the reliability of the system with n_c PUs can be expressed as

$$\mathcal{R}_{system}(q \cdot t_p) = e^{-\sum_{b=1}^{n_c}(q\cdot\mathcal{A}_b)^{\beta_b}}. \tag{3.21}$$

As MTTF $\gg t_p$, the system-level $MTTF$ approximation can be calculated at the end of each period as

$$MTTF_{system} = \sum_{a=0}^{\infty} e^{-\sum_{b=1}^{n_c}(a\cdot\mathcal{A}_b)^{\beta_b}} \cdot t_p. \tag{3.22}$$

To reduce the computational cost of (3.22), a more efficient method, Speedup Technique-I is proposed in [62] by assuming the reliability of a PU keeps the same for every v periods. The estimated system-level MTTF is

$$MTTF_{system}^{speed\_up-I} = \sum_{a=0}^{\infty} e^{-\sum_{b=1}^{n_c}(a.\mathcal{A}_b.v)^{\beta_b}}.t_p.v. \tag{3.23}$$

The accuracy of the state-of-the-art system-level MTTF in (3.23) depends on three parameters, i.e., the highest value of a, period t_p, and the number of periods v used to expedite the approximation. It is advisable to reach a high value of a until the MTTF is saturated, which drains a significant amount of computation time, as later observed in Section 3.7.2. To achieve higher computational efficiency, it is necessary to find an alternative strategy to estimate the system-level MTTF efficiently, reducing the computation time without compromising the MTTF estimation accuracy significantly.

3.6.1 A Computing Efficient Approach for MTTF Calculation

When optimizing the system design goal such as MTTF using meta-heuristics like a genetic algorithm or simulated annealing, the computation efficiency could be one of the primary concerns since a computation efficient algorithm enables a more effective design space exploration. As mentioned before, the approach based on (3.23) is very time-consuming, which severely limits its searching scope and efficiency. Therefore, to acquire better design results, it is highly desirable that a more computing-efficient approach for the MTTF calculation can be utilized in the search engine during the design space exploration process.

As the automotive applications have task interval lengths in ms to μs [65, 116], a PU's temperature dynamics can be safely treated as a constant value in a period. This fact helps to simplify the system-level formulation of MTTF. Let η_1, η_2,...,η_{n_c} be the scale parameters of the **PU** given by (3.18). We assume that they depend on a constant temperature over the period, which can be calculated using the average temperature of the period in the stable status

$$T_{avg,b} = \frac{\sum_{a=0}^{s-1} \int_{l_a}^{l_{a+1}} \frac{A(a)}{B} + \left(T(t_a) - T_{amb_k} - \frac{A(a)}{B}\right) e^{-B \cdot \Delta_a} + T_{amb_k}}{t_p}, \tag{3.24}$$

where s represents the number of intervals in a period, $\Delta_a = (l_{a+1} - l_a)$ is the length of each interval, t_p is the period, and b is the PU index from 1 to n_c.

To further improve the computational efficiency and safely bound the MTTF in an ECS, we take advantage of using the Weibull distribution's slope parameter as a uniform value for all PUs to expedite MTTF computations by adopting the worst-case wear-out rate on each unit. The ECS system reliability can then be formulated as

$$\mathcal{R}_{system} = e^{-\sum\limits_{b=1}^{n_c}\left(\frac{t}{\eta_b}\right)^{\beta}}. \tag{3.25}$$

Based on equation (3.25), we can formulate the system-level MTTF as

$$MTTF_{system}^{Fast-same} = \int\limits_0^{\infty} e^{-\sum\limits_{b=1}^{n_c}\left(\frac{t}{\eta_b}\right)^{\beta}} dt. \tag{3.26}$$

By substituting $x = \sum\limits_{b=1}^{n_c}\left(\frac{t}{\eta_b}\right)^{\beta}$ in (3.26), we have

$$x = t^{\beta} \cdot \sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right). \tag{3.27}$$

$$x^{\frac{1}{\beta}} = t \cdot \left(\sum_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}. \tag{3.28}$$

Further, we have

$$t = \frac{x^{\frac{1}{\beta}}}{\left(\sum\limits_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}}, \tag{3.29}$$

$$dt = \frac{x^{\frac{1}{\beta}-1} \cdot dx}{\beta \cdot \left(\sum\limits_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}}. \tag{3.30}$$

Note that $t \to 0$ leads to $x \to 0$ and $t \to \infty$ leads to $x \to \infty$. Now, we can factor out (3.26) and obtain our fast MTTF formula as

$$MTTF_{system}^{Fast-same} = \frac{1}{\beta \cdot \left(\sum\limits_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}} \int\limits_0^{\infty} e^{-x} \cdot x^{\frac{1}{\beta}-1} \cdot dx, \tag{3.31}$$

which converges to the following simple form [117],

$$MTTF_{system}^{Fast-same} = \frac{\Gamma(\frac{1}{\beta})}{\beta \cdot \left(\sum\limits_{b=1}^{n_c}\left(\frac{1}{\eta_b^{\beta}}\right)\right)^{\frac{1}{\beta}}}. \tag{3.32}$$

Equation (3.32) is independent of compute-expensive integration operations, and $\Gamma(\frac{1}{\beta})$ can be readily calculated. So, our proposed MTTF estimation method in (3.32) is highly computation efficient and can tightly bound the MTTF for a heterogeneous ECS. As shown in our experimental Section 3.7.1(B), the proposed MTTF formula in (3.32) can speed up the computing efficiency of (3.23) by 12× for synthetic test cases and 164× for the *Real-Life* benchmark. Moreover, the proposed approximation approach can achieve an average error below 0.1% for the estimation of system-level MTTF.

3.6.2 Fast System-Level MTTF Calculation for PUs with different degradation rates

The MTTF formula proposed in Section 3.6.1 can accurately estimate a system-level MTTF, when (i) the temperature fluctuation in one period is small enough to be considered as a constant value, e.g., when applications' periods on an ECS are as short as several μs or ms; (ii) the PU's wear-out rate has a negligible variance compared to the worst-case wear-out rate in an ECS, e.g., when all PUs in an ECS have similar operational status and ambient temperatures. However, according to [38], the ambient temperature of different PUs in one vehicle could vary as much as $90°C - 150°C$ due to different mounting locations, which may potentially cause a large wear-out rate variance within an ECS. Therefore, equation (3.32) could result in an inaccurate system-level MTTF estimation without an adequately justified wear-out rate.

When adopting the uniform wear-out rate MTTF formula in equation (3.32), results in the same slope parameter (β) of the Weibull distribution, could save several magnitudes of the computational cost, but it deviates from the practical heterogeneous PUs settings. Although the state-of-the-art method for system-wide MTTF calculation in equation (3.23) can be used to address the heterogeneity of the PU wear-out rates, its computational cost could be prohibitively high, since its MTTF formula needs an iterative accumulation of each *scheduling interval* on all PUs. In the meantime, the system-level MTTF in (3.32) is no longer a closed-form if taking different wear-out rates directly, which harms the computational efficiency. To this end, we use heterogeneous thermal wear-out rates of all the PUs to construct system-wide statistical wear-out rate value, which can be safely implanted into equation (3.32) for maintaining high accuracy and computational efficiency at the same time.

To address these challenges, we propose an approximation approach, which assumes the wear-out rates of the entire system to be an identical statistical value. Specifically, let $\mathfrak{B} = \{\beta_1, ..., \beta_{n_c}\}$ be the slope parameters for Weibull distributions for PUs. Then, we implant four different statistical parameters in (3.32), including maximal, minimal, average, and geometric mean values, as follows:

1. Let $\beta = \beta_{max}$, where
$$\beta_{max} = max\{\beta_1, ..., \beta_{n_c}\}. \tag{3.33}$$

2. Let $\beta = \beta_{min}$, where
$$\beta_{min} = min\{\beta_1, ..., \beta_{n_c}\}. \tag{3.34}$$

3. Let $\beta = \beta_{avg}$, where
$$\beta_{avg} = average\{\beta_1, ..., \beta_{n_c}\} = \frac{\beta_1 + \beta_2 + ... + \beta_{n_c}}{n_c}. \tag{3.35}$$

4. Let $\beta = \beta_{geo}$, where
$$\beta_{geo} = geo\ mean\{\beta_1, ..., \beta_{n_c}\} = \sqrt[n_c]{\beta_1 \cdot \beta_2 \cdot ... \cdot \beta_{n_c}}. \tag{3.36}$$

With different statistical wear-out rate approximations given by (3.33)-(3.36), we can obtain an effective slope parameter for each PU as,

$$\eta_b = \frac{MTTF(T_{avg,b})}{\Gamma\left(1 + \frac{1}{B}\right)}, \tag{3.37}$$

where $b = 1, ..., n_c$ and $B \in \{\beta_{max}, \beta_{min}, \beta_{avg}, \beta_{geo}\}$. With the help of (3.32) we estimate the computationally efficient system-level MTTF for different wear-out rates of the **PU** as,

$$MTTF_{system}^{Fast-diff} = \frac{\Gamma(\frac{1}{B})}{B \cdot \left(\sum_{b=1}^{n_c} \left(\frac{1}{\eta_b^B} \right) \right)^{\frac{1}{B}}}. \tag{3.38}$$

Using the same parameter settings in Section 3.7.1(A), we compare the MTTF accuracy of (3.38) with state-of-the-art equation (3.23). We observe that the geometric mean ($\beta = \beta_{geo}$) generates the lowest error in average ($\approx 2\%$) among all the candidates for both synthetic test cases and practical benchmarks. Therefore, the rest of the chapter uses $\beta = \beta_{geo}$ for fast system-level MTTF calculation for PUs with different degradation rates with minimal accuracy degradation.

With a safer peak temperature bound and a more accurate and efficient MTTF formulation, we are ready to employ a genetic algorithm for Problem 3.3.2. Our genetic algorithm implementation is similar to that presented in [108], and its set-up is briefly described in the following section.

3.7 Experimental Results

In this section, we present our experiments and results to validate the performance of our proposed approaches with different parameter settings.

3.7.1 Experimental Set-up

Our genetic algorithm is set up as follows:

- *Chromosome design*: Each chromosome represents a feasible solution to the task scheduling problem. Chromosome comprises the first part as task mapping and later as the task schedules. We randomly initialized the task mappings on the given PUs and initial task schedules were generated with respect to the precedence constraints of the DAG $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The pool of the initial population was randomly generated with the chromosomes for the given size.

- *Fitness function*: The formulation of the fitness function is crucial to refining the PU design alternatives, and it also impacts the solution's superiority using a metaheuristic approach. To solve Problem 3.3.2, we need to co-optimize the makespan, peak temperature, and MTTF. First, we calculated the makespan similar to that in [108]. Then, we adopted our proposed peak temperature bound in Algorithm 1 and fast system-level MTTF with the same wear-out rate together. In another experiment, we endorsed the proposed fast system-level MTTF formulation's effectiveness with different wear-out rates. Accordingly, our fitness functions are:

  1. The peak temperature for both Genetic Algorithms (GAs) is calculated using our proposed Algorithm 1. The system-level MTTF values are estimated by (3.23) and (3.32) for both GAs to compare their performance. We normalized the makespan, temperature, system-level MTTF and defined the weighted fitness function as,

$$f_1 = \mathbb{P}_1 \frac{C_l - C_{max}}{C_l} + \mathbb{P}_2 \frac{T_{max} - T_m}{T_{max}} + \mathbb{P}_3 \frac{MTTF_{system}}{MTTF_{max}}, \tag{3.39}$$

where $\mathbb{P}_1, \mathbb{P}_2, \mathbb{P}_3$ are the weights having $\mathbb{P}_1 + \mathbb{P}_2 + \mathbb{P}_3 = 1$, and C_l is the smallest possible makespan derived from the solutions of the first generation. $MTTF_{max}$ is a system-level maximum MTTF estimated by assuming the ambient temperature for each PU.

2. Our experimental study also examines the efficacy of the proposed system-wide MTTF calculation for PUs with different wear-out rates by implementing two separate GAs with the system-level MTTF calculation by (3.23) and (3.38), respectively. We used a normalized weighted sum of the makespan and system-level MTTF in the objective function as

$$f_2 = \mathbb{R}_1 \frac{C_l - C_{max}}{C_l} + \mathbb{R}_2 \frac{MTTF_{system}}{MTTF_{max}}, \tag{3.40}$$

   where $\mathbb{R}_1, \mathbb{R}_2$ are the weights with $\mathbb{R}_1 + \mathbb{R}_2 = 1$, and other parameters have the same significance as in the above case-(1).

- *Parent/survivor selection*: Due to multi-objective fitness functions, we maintained the Pareto optimal front during the evolution process from generations-to-generations and used the tournament selection policy to choose the survivors for the next generation [108]. In the tournament selection, we randomly chose 10 chromosomes, sorted them based on their fitness, and picked the top 5 for crossover and mutation. The remaining 5 elements act as the non-evolved chromosomes for the next generation.
- *Crossover/mutation operators*: We used an adaptive crossover operator on the task mapping part of the survivor chromosomes. For randomly chosen two chromosomes, a crossover point is randomly selected between 1 to n_v, and the parts of the two chromosomes after it are exchanged to produce two descendants. Afterward, the crossover descendants are mutated, where any of the pre-mapped tasks in a chromosome is randomly selected, and its mapping is randomly altered to another PU. (see [108] for more details)

We utilize both synthetic test cases and practical benchmarks for verification of the performance of the proposed approaches regarding temperature bound and

Table 3.2: PU parameters for the experiment [71, 118]

ϕ_0(A)	ϕ_1(A/°C)	C_{th}(J/°C)	R_{th}(°C/W)	max. Power(W)
0.035	0.016	0.0454	22	3.86

Table 3.3: Benchmark parameters

Benchmark	n_c	n_v	$\mathbf{P_d}$**(task)**	$\mathbf{W}_{n_v \times n_c}, \mathcal{E}$	$\mathbf{T_{amb}}$
Real-Life[‡]	16	31	[0.1W, 3.86W][¶]	[$100\mu s$, $400\mu s$][‡]	[35°C, 45°C]
ACC[§]	4	20	[0.1W, 3.86W][¶]	Specified for each task/edge[§]	[35°C, 45°C]

[‡]described in [65], [§]described in [116], [¶]described in [71]

system-level MTTF formulations. The parameters for the PUs used in our experiments are described in Table 3.2. For synthetic test cases, the task graphs were randomly generated using a Task Graph Generator [65] with *average computation cost* = 15 ms, *communication to computation ratio* = 1, and the *shape* parameter varied in the range of $[0.3, 0.9]$ to cover a wide gamut of the test cases. The dynamic power for all task nodes was chosen to be uniformly distributed in the range of $[0.1\ W, 3.86\ W]$ adhering to practical automotive systems [71]. Apart from synthetic tests, we verified our proposed formulations on two practical benchmarks, *Real-Life* [65] and *Automotive Cruise Controller (ACC)* [116], with their parameters listed in Table 3.3.

To achieve about 10 years of a lifetime by assuming a drive time of 2.5 hours per day for a PU at the temperature of $80°C$ [39], we used $A_c = 10^{-7}\ cm^2$, $J = 5.7 * 10^5\ Amp/cm^2$, and $E_a = 0.48\ eV$ to estimate the electromigration-induced MTTF [76]. For the existing model of system-level reliability, we used $v = 100$ in equation (3.23) [62]. The ambient temperatures were randomly chosen in the interval [35°C, 45°C] to account for the variation of the PU mounting locations if not otherwise specified.

Our experiments were lined-up in the following two major categories:

A) *Verification of different approaches to our target research problem:*

For these experiments, we compare the effectiveness and efficiency of different approaches to our research problem, i.e., Problem 3.3.2 in Section 3.3.4. Our experiments were conducted based both on synthetic test cases as well as practical real-life benchmarks. Specifically, for synthetic test cases, we adopted an ECS containing 3 PUs with 10, 15, 20, and 25 tasks to simplify tracing parameter trade-offs. A sufficiently high-temperature threshold T_{max} of 150°C was set because the *simple thermal approach* can lead to a very high peak temperature and the mathematical solver is unable to produce a feasible solution under a tightly constrained T_{max} otherwise.

In our genetic algorithm implementation, the initial solution pool was chosen to be 500 random samples, and the genetic algorithm was progressed for 400 generations, thereby terminating with Pareto optimal front solution in the end. The uniform worst-case slope parameter of the Weibull distribution was chosen to be $\beta = 2$ [62]. For both synthetic test cases and benchmarks, without losing generality, we set the weight combinations as $\mathbb{W}_1 = \mathbb{W}_2 = 1/2$ and $\mathbb{P}_1 = \mathbb{P}_2 = \mathbb{P}_3 = 1/3$, which commits an equal influence to each parameter. The four approaches tested in this category are stated below:

- **Temperature Oblivious Approach (M-TO):** The mathematical programming model that minimizes the application makespan but does not consider thermal behavior as the state-of-the-art integer linear programming method in [107].

- **Simple Thermal Aware Approach (M-STA):** The mathematical programming approach explained in Section-3.4 that assumes a PU reaches its stable status temperature immediately [60, 61].
- **Accurate Peak Temperature Identification with Fast MTTF Calculation (G-APTI-FMC):** The genetic algorithm-based approach bounds the peak temperature accurately with the proposed Algorithm 1 and proposed fast system-level MTTF formulation using a uniform system-wide wear-out rate as described in Section-3.6.1.
- **Accurate Peak Temperature Identification with Traditional MTTF Calculation (G-APTI-TMC):** The genetic algorithm-based approach that identifies the peak temperature accurately using the proposed Algorithm 1 and the state-of-the-art system-level MTTF formulation in equation (3.23) [62].

B) *Verification of the proposed approaches for fast system-level MTTF calculation*:

For these experiments, we focus our study on the efficacy of the proposed approaches for fast MTTF calculation. To study the performance of our fast MTTF calculation based on the same degradation rate, i.e., based on equation (3.32), we assumed that all tasks were assigned to a single PU in our generated synthesized test cases and real-life benchmarks. We compared this approach with the traditional approach (i.e., [62]), and the results for the average CPU times and error percentages by these two approaches were collected and listed in Table 3.4 and Table 3.5, respectively.

We then extended our experiments to more sophisticated scenarios when different PUs may have different degradation rates. To this end, we assumed

Table 3.4: Average CPU time

Num. of tasks	10	20	30	40	50	Real-Life	ACC
Avg. CPU time (ms) for MTTF method in [62]	44.21	48.59	50.46	52.96	56.40	685.156	49.84
Avg. CPU time (ms) for proposed MTTF method	3.609	3.812	3.906	4.063	4.188	4.17	3.859

Table 3.5: Average MTTF error percentage

Num. of tasks	10	20	30	40	50	Real-Life	ACC
Avg. error (%)	0.023	0.032	0.050	0.067	0.084	0.00033	0.077

the system contains 100 tasks and the number of PUs to be 5, 10, 15, and 20. We randomly chose the slope parameter β for each PU in the range [2, 3], which signifies the early lifetime wear-out [119]. We used the equal weights as, $\mathbb{R}_1 = \mathbb{R}_2 = 0.5$ in this set-up. The following approaches were implemented and tested:

- **Traditional MTTF Calculation with different degradation rates (G-TMC-DDR)**: It is a genetic programming-based approach with the objective of makespan minimization and system-level MTTF maximization, which is estimated by the existing method in equation (3.23) [62]. Here we used the GA with the initial solution space of 100 samples and evolution over 200 generations.
- **Fast MTTF Calculation with different degradation rates for small size of GA (G-FMC-DDR-S)**: It is based on a genetic algorithm to minimize the makespan and maximize the system-level MTTF

estimated by the proposed formulation in Section-3.6.2 with an initial population pool of 100 samples, and we evolve it for 200 generations.

- **Fast MTTF Calculation with different degradation rates for large size of GA (G-FMC-DDR-L)**: It is based on a genetic algorithm to minimize the makespan and maximize the system-level MTTF estimated by the proposed formulation in Section-3.6.2 with an initial population pool of 500 samples and evolved for 1000 generations.

The industrial-grade linear solver CPLEX 12.10.0 was used to solve the formulated mathematical programming problems of M-TO and M-STA. Whereas G-APTI-FMC, G-APTI-TMC, G-FMC-DDR, G-TMC-DDR-S, and G-TMC-DDR-L were implemented using Matlab R2020a, running on an HP Z800 server with 24 cores and 32GB memory.

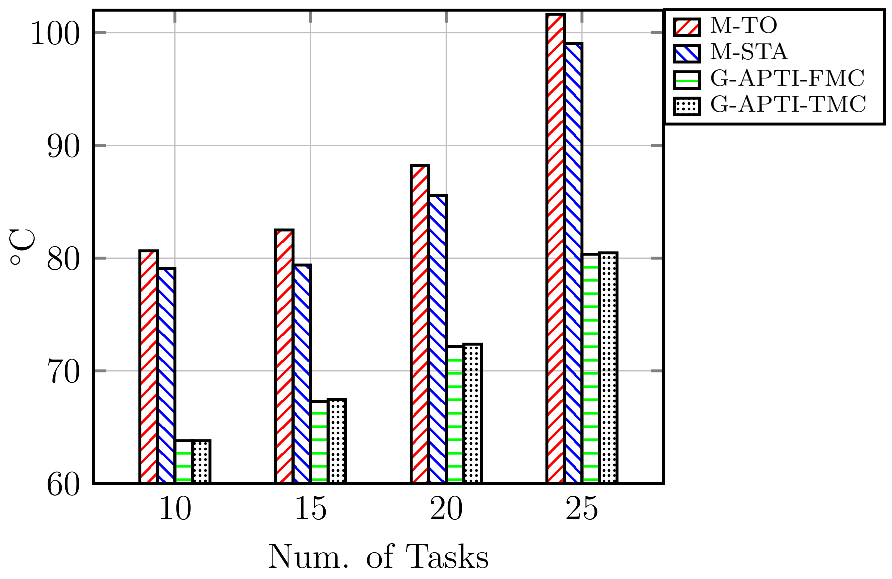
3.7.2 Experimental Results and Discussions

This section presents the experimental results for the set-up described in the previous section and discusses our findings in detail.

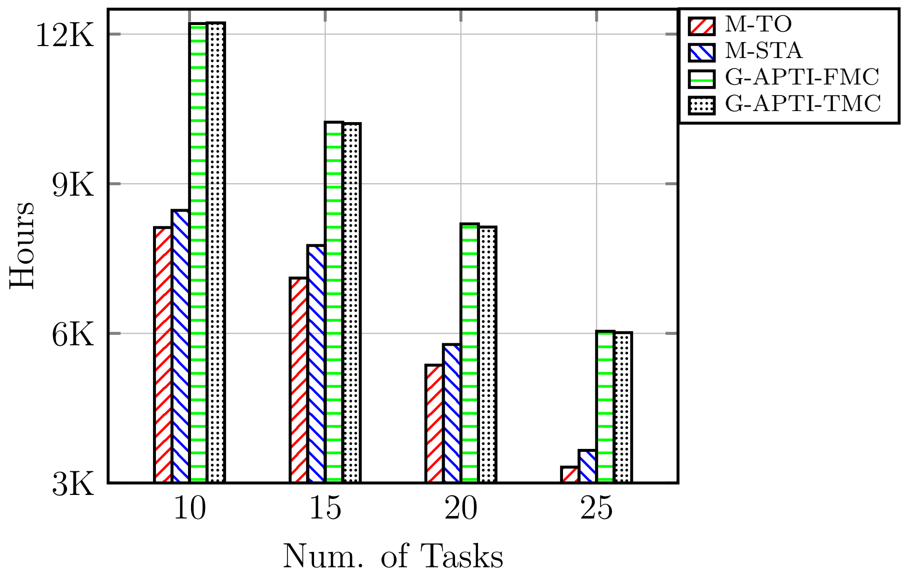
A) *Verification of different approaches to our target research problem:*

Figure 3.6 shows the results (average peak temperature, average system-level MTTF, average makespan, and average CPU time) by four different approaches, i.e., *M-TO*, *M-STA*, *G-APTI-FMC*, and *G-APTI-TMC*, as introduced above.

From Figure 3.6c, we can see that *M-TO* [107] has the smallest makespan among all the approaches, but Figure 3.6a and Figure 3.6b show that *M-TO* has the highest peak temperature and the lowest system-level MTTF. This result clearly indicates that constraining the peak temperature in an automotive

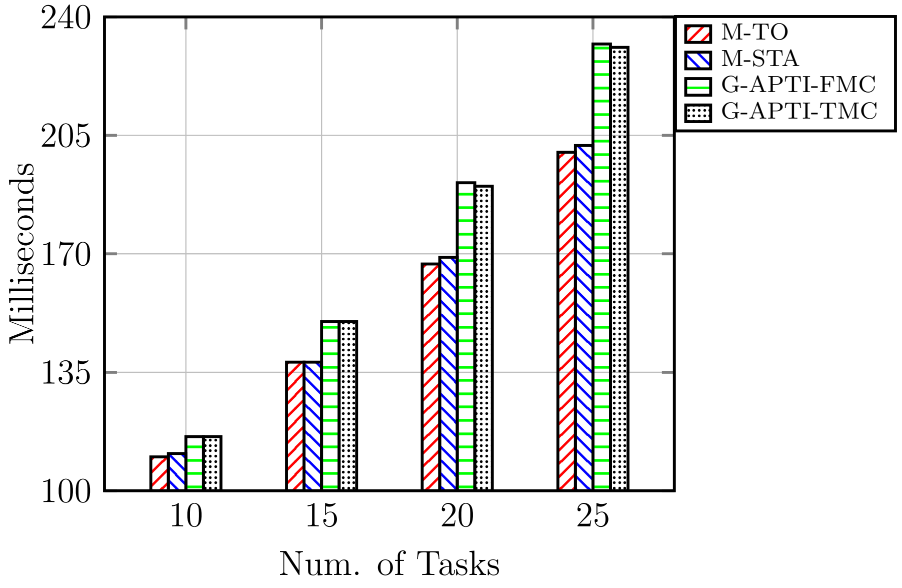


(a) Avg. system peak-temperature. The temperature bound estimated by the *M-STA* [60, 61] is 15-18°C pessimistic than the developed *G-APTI-FMC* method.

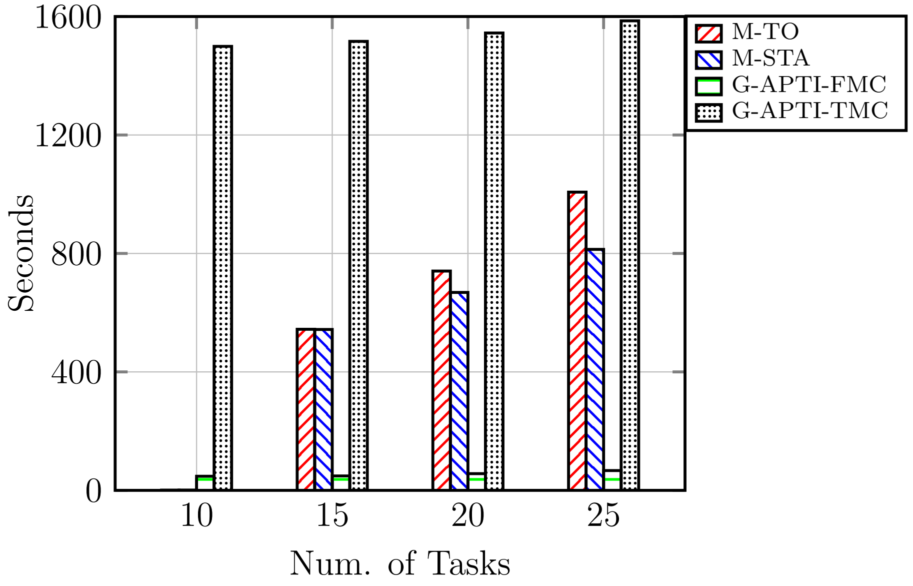


(b) Estimated avg. system-level MTTF. The MTTF values estimated by the developed *G-APTI-FMC*, and the existing *G-APTI-TMC* [62] method are similar.

Figure 3.6: Experimental results for 3-Processing Unit system



(c) Avg. makespan. The makespan of the thermal ignorant method, i.e., *M-TO* is always lower than the others. Makespans by the developed *G-APTI-FMC*, and the existing *G-APTI-TMC* [62] method are similar.



(d) Avg. CPU time. The CPU time of the developed *G-APTI-FMC* method is 23 to 31 times lower than the existing *G-APTI-TMC* [62] method.

Figure 3.6: Experimental results for 3-Processing Unit system (cntd.)

ECS is necessary for hardware protection and system-wide lifetime reliability enhancement. When thermal behavior is considered, the makespans for other approaches become longer since the temperature constraint limits the computational throughput on a PU. Also, with increased thermal prediction accuracy, *G-APTI-FMC* and *G-APTI-TMC* [62] allow to have a longer makespan but a much lower peak temperature and higher system-level MTTF, as clearly shown in Figures 3.6a and 3.6b.

Meanwhile, from Figure 3.6a, we observe that the average peak temperatures for *G-APTI-FMC* and *G-APTI-TMC* [62] differ approximately in the range of $0.1°C$ to $0.3°C$. Also, in Figure 3.6c, their makespans differ by 0 ms to 1 ms. Similarly, the estimated average system-level MTTF differ by 0.1%, 0.26%, 0.77%, and 0.44% respectively from 10 to 25 tasks as depicted in Figure 3.6b. But, as observed in Figure 3.6d, there is a substantial difference in CPU times between the *G-APTI-TMC* and *G-APTI-FMC*. Thanks to our fast MTTF calculation methods, our approach (*G-APTI-FMC*) can achieve similar performance results (average peak temperature, average makespan, and overall average system-level MTTF) with the CPU times about 23 to 31 times lower than *G-APTI-TMC* [62].

In terms of peak temperature identification accuracy, if we compare the results by *G-APTI-TMC* with *M-STA* in Figure 3.6a, we can see that *M-STA* [60,61] is quite pessimistic, i.e., by $15.3°C$, $11.9°C$, $13.2°C$, and $18.5°C$ as task numbers increase from 10 to 25. These results demonstrate that our proposed temperature identification methods, as shown in Algorithm 1, can greatly help *G-APTI-TMC* to obtain the solution with significantly reduced peak temperature and much improved system-level MTTF over *M-STA* [60, 61], as much as 44.38%, 31.49%, 40.78%, and 64.61%, as observed in Figure 3.6b.

Table 3.6: Results of Real-Life Benchmark

	M-TO	M-STA	G-APTI-FMC	G-APTI-TMC
Avg. Makespan (μs)	518	532	1167	1169
Avg. Sys. T_{peak} ($^{\circ}C$)	116.1	114.6	74.5	74.4
Avg. Sys. MTTF (Hr)	1336	1389	5005	5006
Avg. CPU time (s)	237.8	683.3	135.5	26001.2

Table 3.7: Results of ACC Benchmark

	M-TO	M-STA	G-APTI-FMC	G-APTI-TMC
Avg. Makespan (ms)	147	150	184	188
Avg. Sys. T_{peak} ($^{\circ}C$)	107.4	102.7	85.3	84.9
Avg. Sys. MTTF (Hr)	2219	2209	4020	4095
CPU time (s)	259.51	123.93	79.34	1370.5

From Figure 3.6d, we can also see that the computational cost for *M-TO* [107] and *M-STA* [60, 61] increases rapidly due to the NP-hardness of the mathematical programming technique. Even so, their CPU times are still lower than that by *G-APTI-TMC* [62]. Besides their ineffectiveness in dealing with temperature issues, the results clearly show that *M-TO* and *M-STA* cannot be used for large-scale systems when the task and PU numbers continue to grow. In the meantime, the results also highlight the need to speed up the system-wide MTTF calculation in design space exploration.

Similar findings are observed for the two automotive benchmarks. For a practical *Real-Life* benchmark, as shown in Table 3.6, there is an average makespan difference of 2 μs, an average peak temperature difference of $0.1^{\circ}C$, and an MTTF difference of 1 Hr between *G-APTI-FMC* and *G-APTI-TMC*, but *G-APTI-FMC* can speed up the system-level MTTF calculations by 191 times than *G-APTI-TMC* [62]. On the other hand, *M-STA* [60, 61] is $40.1^{\circ}C$ higher in peak temperature and 3616 Hr lower in system-level MTTF than

G-APTI-FMC, which underlines the important contribution of the proposed *G-APTI-FMC* for the practical systems.

In the meantime, for the *ACC* benchmark, in Table 3.7, we can see that our proposed approach, *G-APTI-FMC*, can improve the computation efficiency of *G-APTI-TMC* [62] by over 17 times with degradation in the peak temperature and system-level MTTF no more than $0.4°C$ and 75 Hr, respectively. Compared with *M-TO* [107] (*M-STA* [60, 61], resp.), *G-APTI-FMC* can improve the peak temperature and system-level MTTF with $22.1°C$ ($17.4°C$, resp.) and 1801 Hr (1811 Hr, resp.), respectively.

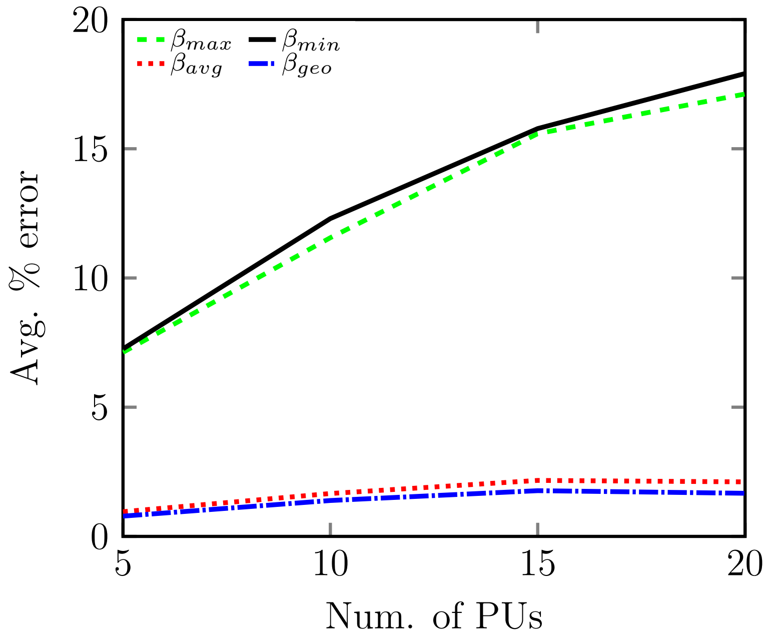
B) *Verification of the proposed approaches for fast system-level MTTF calculation:*

For single PU or PUs with similar degradation rates, as shown in Tables 3.4 and 3.5, our proposed fast MTTF calculation can be extremely efficient and effective. As shown in Table 3.4, the proposed MTTF method can speed up the computations about 12× for synthetic test cases and 164× for the Real-Life benchmark compared with the state-of-the-art method (3.23) [62]. In the meantime, from Table 3.5, we can see that the average error is less than 0.1% for both synthetic task cases and practical automotive benchmarks.

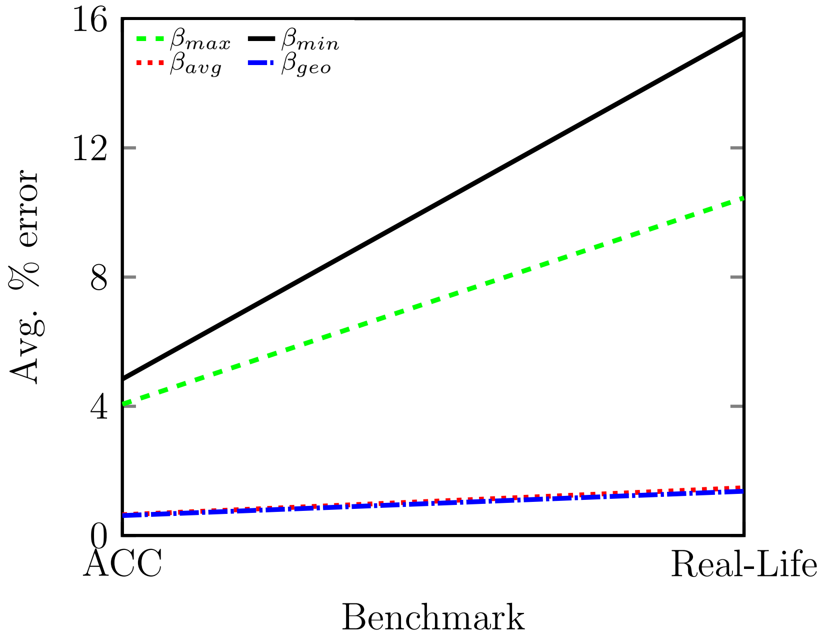
To consider the case when different PUs may have different degradation rates, we propose four candidate approaches, i.e., (3.33)-(3.36), to be used in (3.38). We tested these four approaches by mapping 100 tasks on 5, 10, 15, 20 PUs in the synthesized test cases and our two practical benchmarks. We compare their performance in MTTF calculations by collecting the estimation errors normalized by the traditional one [62] (equation (3.23)). These results are denoted as β_{max}, β_{min}, β_{avg}, and β_{geo}, respectively, and shown in Figure 3.7.

As shown in Figures 3.7a and 3.7b, we can see that the approach with β_{geo} has less than a 2% average error compared to the existing system-level MTTF equation (3.23) for both synthetic test cases and practical benchmarks. Further, we analyze the proposed methodology to compute the system-level MTTF with different degradation rates. We randomly generated the synthetic test cases for the configurations of 100-Tasks and 5, 10, 15, 20 PUs and collected the 100 feasible test results, together with their peak temperatures, system-level MTTFs, makespans, and CPU times of the solutions, shown in Figure 3.8.

In design space exploration, e.g., in a genetic algorithm or simulated annealing, evaluating the design objective function accurately and quickly is vital for the algorithm's effectiveness. To accurately evaluate the fitness of a design alternative helps to direct the search in the right direction, while the computation efficiency in fitness evaluation enables an extensive exploration of the design space. With the similar size of design space by *G-TMC-DDR* [62] and *G-FMC-DDR-S*, as shown in Figure 3.8a, we observe that the average peak temperatures differ by $0.1°C$, $0.2°C$, $0.9°C$, $0.3°C$ when PU numbers increase from 5 to 20. Accordingly, their average system-level MTTFs differ by 0.8%, 0.6%, 1.9%, and 2.1%, as shown in Figure 3.8b and their average makespans differ by 0 ms, 5 ms, 1 ms, 2 ms in Figure 3.8c. However, as observed in Figure 3.8d (left Y-axis for the *G-TMC-DDR*, right Y-axis for the *G-FMC-DDR-S* and *G-FMC-DDR-L*), the CPU time of the *G-TMC-DDR* [62] is 54, 95, 110, 108 times higher than that of *G-FMC-DDR-S*, respectively, as the PU number increase from 5 to 20. Hence, with our fast MTTF calculation as

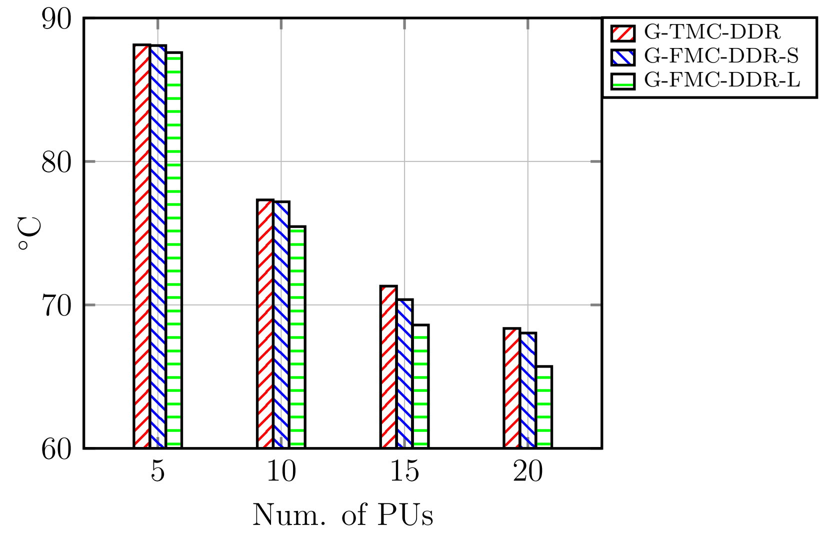


(a) Synthetic Test Cases. The β_{geo} method has the lowest average error percentage.

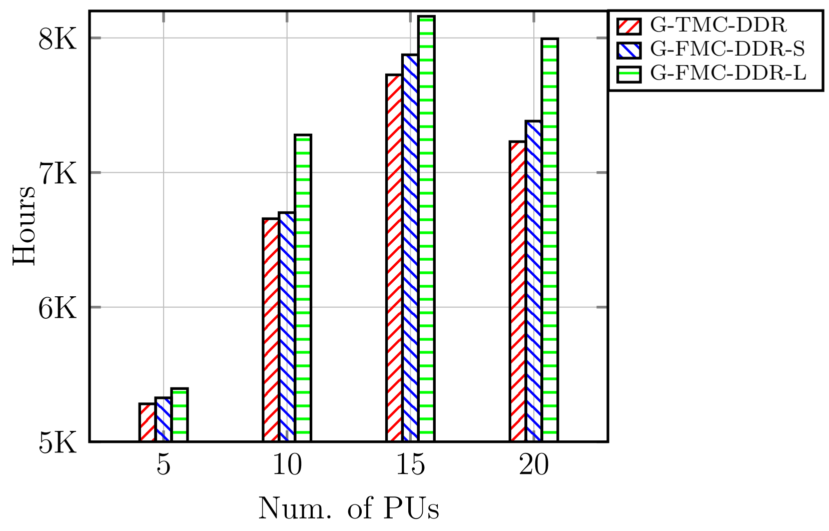


(b) Practical Benchmarks. The β_{geo} method has the lowest average error percentage.

Figure 3.7: Average error percentage

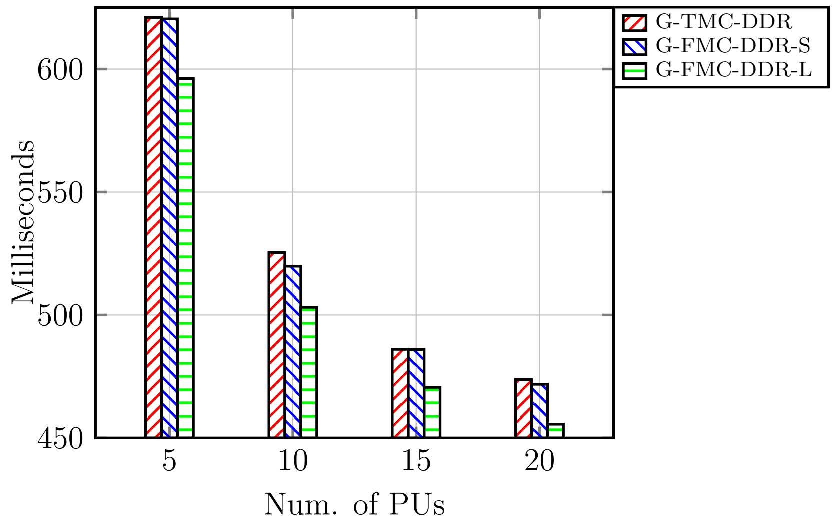


(a) Avg. system peak-temperature. The existing *G-TMC-DDR* [62] and the developed *G-FMC-DDR-S* methods have comparable peak temperature estimation values. The developed *G-FMC-DDR-L* method with a large size of the genetic algorithm improves the peak temperature estimation than the others.

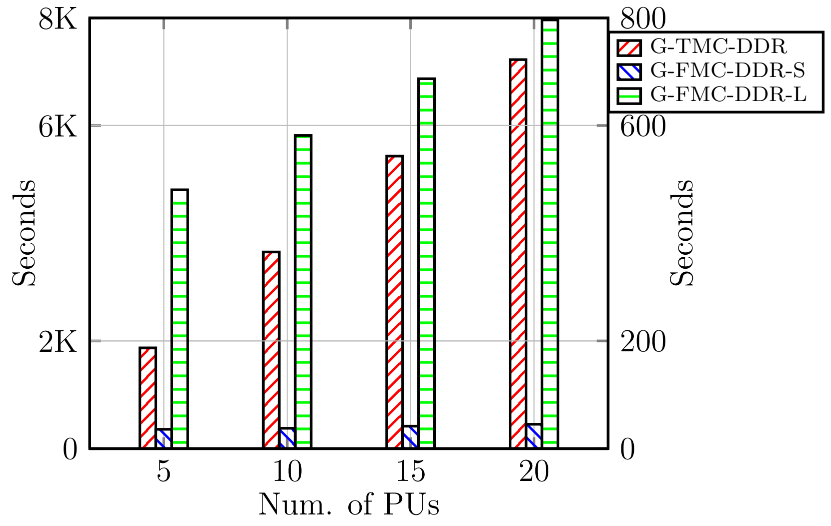


(b) Estimated avg. system-level MTTF. The existing *G-TMC-DDR* [62] and the developed *G-FMC-DDR-S* methods have comparable system-level MTTF estimation. The developed *G-FMC-DDR-L* method with a large size of the genetic algorithm significantly improves the MTTF estimation than the others.

Figure 3.8: Experimental results for 100-Tasks system



(c) Avg. makespan. The existing *G-TMC-DDR* [62] and the developed *G-FMC-DDR-S* methods have comparable makespan values. The developed *G-FMC-DDR-L* method with a large size of the genetic algorithm significantly improves the makespan than the others.



(d) Avg. CPU time. The CPU time of the existing *G-TMC-DDR* [62] method is 54 to 110 times higher than the developed *G-FMC-DDR-S*. (Note: Left hand side Y-axis is for *G-TMC-DDR*, and right hand side Y-axis is for *G-FMC-DDR-S* and *G-FMC-DDR-L*)

Figure 3.8: Experimental results for 100-Tasks system (cntd.)

Table 3.8: Results of ACC Benchmark for diff degradation rate

	G-TMC-DDR	**G-FMC-DDR-S**	**G-FMC-DDR-L**
Avg. Makespan (ms)	188	188	180
Avg. Sys. T_{peak} ($°C$)	89.69	90.02	88.34
Avg. Sys. MTTF (Hr)	4149	4164	4264
Avg. CPU time (s)	1567.9	15.58	178.92

formulated in (3.38), we can achieve a similar (a little inferior) performance in terms of the peak temperature, system-level MTTF, and makespan, but significant CPU time improvement.

The highly efficient MTTF calculation enables us to search a much larger solution space. In *G-FMC-DDR-L*, we increase both the population size and the reproduction generations in the genetic algorithm, with the total size of design space increased by 25 times. As observed in Figure 3.8d, with increased search space, the *G-FMC-DDR-L* approach requires more CPU time than *G-FMC-DDR-S*, but it helps to yield better results. For *G-FMC-DDR-L*, in Figure 3.8a, we see that the average peak temperature is reduced by $0.5°C$, $1.8°C$, $2.7°C$, $2.6°C$ when PU numbers increase from 5 to 20. Accordingly, their average system-level MTTFs improve by 113 Hr, 622 Hr, 434 Hr, 764 Hr as shown in Figure 3.8b, and Figure 3.8c, the average makespans reduced by 25 ms, 23 ms, 16 ms, 18 ms compared with *G-TMC-DDR* [62]. In the meantime, it is worth pointing out that, by increasing the size of the initial population and number of generations for evolution in *G-FMC-DDR-L*, we can obtain solutions with better performance metrics (i.e., average peak temperature, average system-level MTTF, and average makespans). Simultaneously consuming much fewer CPU times, i.e., 3, 6, 7, 8 times less than the *G-TMC-DDR* [62] according to Figure 3.8d.

Table 3.9: Results of Real-Life Benchmark for diff degradation rate

	G-TMC-DDR	**G-FMC-DDR-S**	**G-FMC-DDR-L**
Avg. Makespan (μs)	1489	1490	1463
Avg. Sys. T_{peak} ($°C$)	69.70	68.94	66.06
Avg. Sys. MTTF (Hr)	8173	8195	8867
Avg. CPU time (s)	32245.2	12.42	260.1

We can see similar results from the two practical automotive benchmarks. As shown in Table 3.8, comparing *G-TMC-DDR* and *G-FMC-DDR-S*, there is not much change in the average makespan. Their peak temperatures differ slightly by 0.33$°C$, and average MTTF also differ slightly by 15 Hr. However, *G-FMC-DDR-S* is faster than *G-TMC-DDR* [62] by 100 times. In the meantime, as shown in Table 3.9, the makespan, peak temperature, and MTTF estimation for *G-FMC-DDR-L* outperform *G-TMC-DDR* [62] significantly, i.e., by 26 μs, 3.64$°C$, and 694 Hr, respectively, with CPU time 124 times lower. The prominence of our fast system-level MTTF calculation helps to achieve better design space exploration due to low timing complexity in the practical systems and offers better results.

3.8 Conclusion

In this chapter, we study how to map a periodic automotive application on PU with temperature issues taken into consideration. We first propose a mathematical programming approach to meet the peak temperature constraint while reducing the application latency. We further propose a more sophisticated genetic algorithm-based approach to deal with the peak temperature constraint and system-wide reliability optimization. To this end, we develop two key algorithms, i.e., guaranteeing the peak temperature for periodic tasks with variable execution times and the analytical approach for system-wide MTTF calculation, which can speed up the process by several orders of magnitudes. Experimental results for two practical automotive benchmarks show that the proposed peak temperature algorithm is accurate by $17°C$ and $40°C$, which results in 81% and 260% more accurate MTTF estimation. At the same time, the proposed MTTF formulation is faster by $17\times$ and $191\times$ with an accuracy loss of less than 0.1%, which validates the efficiency of our proposed approach.

CHAPTER 4

THERMAL AWARE NEURON SENSITIVITY ANALYSIS FOR DEEP NEURAL NETWORKS

In the age of miniaturization and Artificial Intelligence/Machine Learning, various DNN accelerators are proposed for mission-critical systems, such as AVs. With the rapidly increasing complexity of the deep learning applications, there is the substantial growth of the size of the DNNs, which demands large memory space. Such large DNNs are continuously retrieved from the memory during real-time inference operations. The shrinking of the memory IC chip's size increases the thermal gradients. On the other hand, as the AVs undergo harsh thermal conditions, the accelerators' memory system is exposed to a high-temperature state. Overall, the temperature affects the performance of DNNs in terms of reduced classification accuracy. Hence, thermal impacts should be considered in the ECS design to better address the accuracy degradation.

As a case study, this chapter considers emerging non-volatile memory technology, e.g., STT-MRAM for DNN storage, and quantifies the thermal impact on accuracy as a sensitivity metric. Despite being a viable solution, data cells of the STT-MRAM suffer from temperature-induced random bit flipping, leading to classification inaccuracy. Not all the DNN parameters' random bit flipping affect the accuracy; therefore, it is required to search and protect the delicate parameters to prevent accuracy loss. As all the parameters associated with the neuron are structured together in memory, we develop a neuron-level analytical sensitivity estimation framework to detect their saliency. We validate the sensitivity framework using popular DNNs and datasets.

4.1 Introduction

Artificial intelligence (AI) technology allows computer applications to learn from their experiences through iterative processing and algorithmic training. With each successful cycle of data processing, AI systems become smarter since each interaction allows the system to test and measure solutions while also developing experience in the task at hand. AI systems can become experts significantly faster than humans because this can be done quickly, far faster than a human can do a similar task, making them very effective solutions for any process requiring intelligent decision making [120].

Machine learning is one of the ways to inculcate artificial intelligence. Traditional Machine Learning techniques require most of the pertinent features to be defined by a domain expert in order to reduce data complexity and make patterns more suitable for learning algorithms to work [23]. On the other hand, Deep Learning algorithms have the advantage that they seek to learn high-level features from data. It does away with domain knowledge and laborious extrication of hardcore features [24]. Therefore, deep neural networks are considered a core component in the upcoming autonomous vehicles.

The modern automotives employ distributed processing architecture to meet the latency requirements of mission-critical functionalities, which involves CPUs, GPUs, and FPGAs [39, 121]. The temperature plays a crucial role in reliable computing system performance. The high thermal gradients across a semiconductor chip degrade the lifetime; for example, every $10°C$ rise in the temperature reduces the lifetime by half [50]. Similarly, exceeding the temperature threshold of the processor throttles the power and may be catastrophic for mission-critical systems, such as automobiles [45]. Therefore, managing thermal issues of the computing platform for safety-critical applications by guaranteeing performance is a prime design objective.

Although deep neural networks are assumed to be fault-tolerant [122], prior studies have shown that the random errors in the underlying computing hardware can misclassify the input objects [123]. For example, Hanif et al. presented a case study in which the memory faults can affect the reliability of the neural networks [124]. For the case study, faults in the main memory and on-chip accelerator memory misclassified the *Hammerhead Shark* as *Great White Shark*. Moreover, a single bit flip in the entire VGG-f network dramatically drops the average accuracy from 80% to about 5%. Likewise, for the VGG11 network on the Cifar10 dataset consisting of 132 million parameters, only 3-bit flips are enough to bring down the classification accuracy from 89.40% to 10.27% [125]. While, authors in [126] have demonstrated that high energy particle strike induced soft errors in the accelerator hardware can wrongly classify the *truck* as a *bird*, which may lead to catastrophic events for autonomous vehicles.

Similarly, the temperature impact on Resistive RAM (ReRAM) based accelerator is studied in [49]. ReRAM stores the parameters as a unique conductance value, and as the temperature increases from $27°C$ to $127°C$, the difference between ON state-OFF state conductance reduces by almost 50% [49]. Due to such a drastic change in the conductance, the encoded parameter values in the ReRAM array get perturbed at high temperatures. Finally, it results in an accuracy drop below 10% at $127°C$ for the state-of-the-art networks such as LeNet, VGG16, ResNet50, and Inceptionv3 [127]. Such a variety of faults in the hardware can manifold as the size of the DNNs increase.

In recent times, the convolution neural networks have scaled from 57.7 million [128] to 829 million parameter space [129]. Similarly, as shown in Figure 1.10 in

Chapter 1, the number of parameters in Transformer models, widely used in commercial natural language processing and computer vision applications, has increased exponentially [1]. Also, the parameter count in the Recommendation System (RecSys) models has reached up to 10000 billion. In particular, the parameter space for such networks has increased by 240× for every 2 years. But, the accelerator memory has not scaled up in the same proportion, which is just 2× improvement per 2 years. The DNN parameters are stored and later retrieved from the memory during inference. The overwhelming growth of neural networks can present power and latency problems for safety-critical applications, such as autonomous cars [130]. The emerging memory technology, i.e., STT-MRAM, can alleviate these challenges, as found in Graphical Processing Unit (GPU) and Processing-In-Memory (PIM) accelerators [131–133]. It has a non-volatility feature along with low access latency, low power, high density, and DRAM-like endurance [47]. In particular, the read access latency and read power requirements are improved by 8.4% and 66.2%, respectively, than the traditional DRAM [48] and integration density (6-20 F^2) as high as DRAM (6-10 F^2) [134].

As a case study, in this chapter, we consider STT-MRAM for DNN parameter storage. Despite several significant advantages over the existing memory system, the STT-MRAM's stability degrades with the temperature. As later found in the motivational example in Section-4.4.1, the STT-MRAM suffers from thermal perturbations, and the DNN accuracy drops abruptly at high temperatures. It indicates a need to protect the accelerators from temperature-induced detrimental effects. The sensitivity analysis of the DNN parameters plays a key role for such scenarios to identify their saliency.

Dash et al. have demonstrated that protecting the most sensitive DNN parameters can maintain the classification accuracy without protecting the entire parameter set [63]. Such parameters selected for protection to maintain the classification accuracy mainly depend on how accurately their sensitivity is estimated. The most straightforward sensitivity estimation is the magnitude of the parameter itself [135]. However, it suffers from the classification accuracy problem at high temperatures, as observed later in the experiment results Section-4.5.2, indicating inaccurate parameter saliency. Khalid et al. have evaluated the sensitivity by calculating the error indices, which involves perturbing individual DNN parameters and tracking the change in the cost function due to each parameter [136]. This method is computationally intense, and moreover, it ignores the cumulative effect of other parameter perturbations.

The first-order derivative, i.e., gradients of the DNN parameters over the cost function is another most commonly used approach for sensitivity estimation [137, 138]. While, authors in [63] have evaluated the sensitivity of DNN parameters based on the second-order derivative, i.e., hessian matrix, which is approximated using dominant eigenvalue and eigenvector pairs [139, 140]. The gradient-based method in [137] assumes the higher-order terms are zero, whereas the hessian-based method in [63] assumes that the gradient and higher-order terms are zero. Therefore, the gradient and hessian methods ignore the correspondence of each other, resulting in classification inaccuracies. Hence, in this chapter, we propose to estimate the sensitivity based on gradient and hessian information.

The chosen parameters for protection based on their sensitivity are not necessary to be in a contiguous memory location while they are accessed by the DNN accelerators, which may lead to more cache misses, thereby affecting the latency for mission-critical applications. On the other hand, parameter index-tracking can be cumbersome for large networks [141]. Therefore, due to the structural ease of parameter access, we opt to find the sensitivity at neuron-level whose associated parameters guarantee to be in the adjacent memory locations. Our contributions are as follows:

- First, we analyze the temperature impact on DNN prediction accuracy when the parameters are stored in the STT-MRAM. It highlights the need to protect the delicate parameters from random thermal perturbations as the accuracy drops sharply at high temperatures.
- Then, we propose the analytical method for neuron-level sensitivity estimation, which employs the first-order and second-order derivative information of the cost function.
- Later, we validate the proposed sensitivity analysis framework for state-of-the-art neural networks and datasets. Our experimental results demonstrate that the proposed neuron-level sensitivity estimation methodology achieves better classification accuracy than the existing methods by 0.5-4.2% for LeNet, ResNet20, ResNet56, and DenseNet40 architectures at a high-temperature range.
- At last, we analyze the limitation of the proposed sensitivity estimation method when the network parameters have a high variance in MobileNet architecture. We train the MobileNet by applying regularization to minimize the parameter variance and evaluate its impact on the accuracy of the developed sensitivity metric.

The rest of the chapter is organized as follows. We present the existing work on sensitivity analysis in Section-4.2. Then we define deep neural network, STT-MARM, and error models in Section-4.3. We describe the motivation for accurate sensitivity estimation in Section-4.4 and later mathematically define the proposed methodology. We present our experiment set-up details and the results in Section-4.5 followed by conclusion in Section-4.6.

4.2 Related Work

The sensitivity analysis of the neural network provides saliency information of the parameters, which can be readily used to detect the vulnerable parameters. A small unnoticeable perturbation in those vulnerable parameters can lead to a significant accuracy loss, which is undesirable for mission-critical systems like AVs. Sensitivity analysis is primarily important in network pruning to reduce its size and maintain security under malicious attacks. In this section, we discuss different sensitivity estimation frameworks available in the literature.

The existing work for sensitivity analysis can be broadly classified as the input sensitivity and the network element sensitivity. The input sensitivity methods identify the saliency of the input data perturbations on the prediction accuracy, whereas later define the saliency of the network elements when they are perturbed. The network elements can be termed as bits, parameters, neurons, and layers.

Input Sensitivity: Several approaches have been proposed in the literature to estimate the input sensitivity. For example, authors in [142] identify the sensitivity of both input and parameter perturbations on DNN's output. It is challenging to identify which element is more sensitive to the output change, as both the input and parameter sensitivities are amalgamated. Sensitivity estimation plays a crucial role in the security aspect of the DNNs. Li et al. have analyzed the robustness of the network under the input variations using a threat model [143].

Meanwhile, Kowalski et al. defined the sensitivity of the inputs for Probabilistic Neural Networks [144]. The authors have utilized Sobol's approach to derive the sensitivity metric [145], which requires first-order, second-order, higher-order, and total sensitivity indices calculation leading computationally intractable for large neural networks. In another method, the sensitivity of the neuron is estimated by calculating the derivative of the output w.r.t. the inputs of the DNN [138].

All these methods quantify the sensitivity over the input set. However, in our approach, we study the temperature impact on the system itself; therefore, we assume that the inputs are undisturbed and focus only on the sensitivity due to thermal parameter/neuron perturbations.

Network Element Sensitivity: This category can be again categorized as a bit-level, parameter-level, neuron-level, and layer-level sensitivity based on the network elements.

Bit-level Sensitivity: It is the lowest abstraction of the network, where DNN parameters are represented as binary bits stored in the memory. There are few recent works exploiting bit-level sensitivity information. Authors in [124] have simulated the sensitivity of bit flips for parameters stored in a single-precision format. According to their analysis, the bit flips from 1 to 0 do not affect the accuracy much, but, contrarily, 0 to 1 bit flips significantly drop the accuracy. The MSB bits of the exponent part are highly sensitive to the perturbations, and sensitivity subsides gradually as the flipping location changes from MSB to LSB.

Long et al. heuristically defined the layer-wise sensitivity by quantizing the parameter representation bits from floating-point to fixed-point notation [146]. They simulate the number of bits required for a layer's parameter representation by assuming other layers still have the floating-point parameters and decide the *number of bits* as a sensitivity metric for the layer under given accuracy loss. However, the methods in [124, 146] fail to provide the analytical model for sensitivity estimation.

In another work, Yao et al. analyzed the bit-level sensitivity using a gradient-based approach combined with a flip-aware search heuristic [125]. In this method, first, the bit-wise gradients are computed to determine the vulnerable bits, followed by the flip-aware search to detect a group of bits responsible for accuracy drop. As the gradients are computed at the bit level, and memory page configurations are required to be maintained for the search algorithm, this method demands high computation cost.

Parameter-level Sensitivity: Parameters of the neural network consists of learnable weights and bias terms. They are initialized to a random guess and later optimized to reduce the network's cost function. Many of the existing applications using sensitivity estimation focus on the parameter level. Han et al. prune the network connections using the parameter sensitivity information [147]. In another approach, the parameter-level sensitivity metric is used to preserve a certain number of the parameters to approximate the entire network's gradients, which helps to speed up the back-propagation phase [148].

For DNN security applications, the *sensitive-sample* signatures are generated for each parameter, which can be later used to check the outsourced model's integrity in the cloud-edge architecture. A similar application can be found in [149]. While Choi et al. have used the parameter sensitivity due to random bit flipping [137]. Similarly, Dash et al. developed the network parameter sensitivity under the noisy stochastic environment for the processing-in-memory accelerator [63]. They apply the sensitivity information to protect the most sensitive network parameters to minimize the accuracy loss.

Neuron-level Sensitivity: A neuron represents a group of trainable network parameters and the bias term. It acts as an interface between layers of the DNN. Neuron level sensitivity information is vital as there is an ease of access for associated parameters. Li et al. defined the neuron sensitivity for pruning the network [135]. Similarly, authors in [150] use the neuron-level sensitivity to prune the insensitive neurons by maintaining similar accuracy. Instead of the incoming parameter values only, authors in [151] defined the neuron's sensitivity by considering the incoming and outgoing parameter values. In another work, Luo et al. estimated the sensitivity of the neurons based on their ability to approximate the output of the following layer [152]. This approach does not consider the impact of the neuron's output on the final output of the network unless it is the neuron in the last layer. Therefore, it does not satisfy the ideal explication of sensitivity estimation. Based on the parameter-level sensitivity model, the neuron-level sensitivity is estimated in [153]. Later, neurons within the higher sensitivity range are assigned maximum precision bits, and neurons in the lower sensitivity range are assigned minimum precision bits.

Layer-level Sensitivity: The layer of DNN is a collection of neurons, which operate simultaneously over the input features. There are few works utilizing the highest abstraction level sensitivity information. Wu et al. used layer-wise unique threshold value as a sensitivity metric, calculated by solving the optimization problem as a differential evolutionary algorithm [154]. The calculated threshold is applied to facilitate the pruning to reduce the memory space. The computational cost of the genetic algorithm-based evolution process used in the paper increases quadratically with the size of the network. In another work, layer-wise sensitivity is applied to detect the adversarial noise saliency [155].

Further, the existing work on sensitivity estimation can be classified based on the metrics used in the computation.

Metric-based Sensitivity: The sensitivity metrics can be divided into three types, such as magnitude-based or L1/L2 norm, gradient-based, and hessian-based.

Magnitude-based Sensitivity: The simplest metric among all the techniques is the absolute parameter value or L1-norm as a sensitivity metric given by [147]:

$$S^i_{L1} = |w_i|, \quad i \in N, \tag{4.1}$$

where $\boldsymbol{W} \in \mathbb{R}^N$ is the parameter space of the DNN. Similarly, Li et al. defined the L1-norm at neuron-level as a sum of the associated parameter magnitudes [135]. If s_a is the size of the neuron ($\mathcal{N}_a$), then L1-norm is defined as

$$\mathbb{S}^{\mathcal{N}_a}_{L1} = \sum_{i=1}^{s_a} |w_i|. \tag{4.2}$$

Also, the authors define the neuron-level L2-norm as

$$\mathbb{S}^{\mathcal{N}_a}_{L2} = \sum_{i=1}^{s_a} ||w_i||_2, \tag{4.3}$$

where w_i represents the parameters associated with the corresponding neurons. In the previous method, only the incoming parameters of the neuron are considered. But, Jiang et al. quantified the neuron's sensitivity by combining the quadratic sum of the incoming and outgoing parameter values. For example, the sensitivity of the a^{th} neuron in l^{th} layer can be

$$\mathbb{S}^{\mathcal{N}_a,l}_{Q} = \sum_h (w^l_{ah})^2 \cdot \sum_j (w^{l+1}_{ja})^2. \tag{4.4}$$

While L2-norm of the layer activations is used to quantify the layer-level sensitivity in [154]. Let a^l_{adv} and a^l are l^{th} layer's activations when the input is compromised by the adversary and original input, respectively. The sensitivity of the l^{th} layer be

$$\mathsf{S}^l_{L2} = \frac{||a^l_{adv} - a^l||_2}{||a^l||_2}. \tag{4.5}$$

Gradient-based: Another most common approach is back propagation-based, i.e., first-order derivatives or gradients. The absolute value of the gradients is used as a sensitivity metric of the parameters in [148]. If C is the network cost function, the i^{th} parameter sensitivity be

$$S^i_{g,L_1} = |\frac{\partial C}{\partial w_i}|. \tag{4.6}$$

In another variant, the square of the L2-norm of the parameter gradients is used as the sensitivity metric in [149, 156], which can be defined as

$$S^i_{g,L_2} = ||\frac{\partial C}{\partial w_i}||^2_2. \tag{4.7}$$

Authors in [137] quantify the sensitivity of the random bit flips using a gradient-based approach. If ϵ_i is the perturbation, and $\mathbb{E}(w_i)$ is the average value due to random bit flips of the i^{th} parameter, then the sensitivity is estimated as

$$S^i_{g,\epsilon_i} = |\frac{\partial C}{\partial w_i}\epsilon_i|, \quad \epsilon_i = w_i - \mathbb{E}(w_i). \tag{4.8}$$

While Hassan et al. used the parameter sensitivity information in equation (4.8) to model the neuron-level sensitivity as [153]

$$\mathbb{S}^{\mathcal{N}_a}_{g,\epsilon} = \frac{1}{s_a}\sum_{i=1}^{s_a}|\frac{\partial C}{\partial w_i}\epsilon_i|. \tag{4.9}$$

In another method, the sensitivity of the neuron is estimated by calculating the derivative of the output w.r.t. the inputs of the DNN [138]. In this approach, the sensitivity of the a^{th} neuron in the current layer w.r.t. the output of the i^{th} neuron in the previous layer for n^{th} sample of the input dataset (x_n) is

$$\mathbb{S}^{\mathcal{N}_a}_i|_{x_n} = \frac{\partial y_a}{\partial x_i}(x_n). \tag{4.10}$$

The average sensitivity of the neuron over the entire dataset of length N be

$$\mathbb{S}^{\mathcal{N}_a,avg}_i = \frac{\sum_{j=1}^{N}\mathbb{S}^{\mathcal{N}_a}_i|_{x_j}}{N}. \tag{4.11}$$

Hessian-based: Recently, the second-order sensitivity analysis has been gaining popularity due to the development of a time-efficient hessian approximation frameworks [139, 140]. Dash et al. developed the network parameter sensitivity using approximated hessian information [63], and the described method achieves better accuracy by protecting a small fraction of the network parameters. If λ and q are the n dominant eigenvalues and eigenvectors, then the i^{th} parameter sensitivity be

$$S_h^i = (\sum_{j=1}^{n} |\lambda_j| q_j^2) \bigodot w_i^2, \tag{4.12}$$

where $\bigodot$ is the Hadamard product operator. Yu et al. used the second-order information to quantify the sensitivity of the group of neurons [150]. In particular, if $H_{p,p}$ is a hessian block of the parameters w_p associated with p neurons, their sensitivity is

$$\mathbb{S}_h^p = \frac{Trace(H_{p,p})}{2p} ||w_p||_2^2, \tag{4.13}$$

where $Trace(H_{p,p})$ is a trace of the hessian block, which can be approximated using the utility in [140].

Overall, the gradient-based sensitivity methods described in [137, 138, 148, 149, 156] ignore the higher-order derivatives. Whereas the sensitivity frameworks in [63, 150] assume gradients to be zero; therefore, they use only the second-order derivative for sensitivity estimation, ignoring higher order derivative terms. The second-order method achieves better accuracy than the gradient, highlighting its importance. These two methods ignore the preeminence of each other. Also, there is structural ease to access the neuron's parameters. Therefore, we propose to utilize the first-order and second-order derivative information of the cost function landscape to better address the sensitivity estimation at the neuron-level. In the following section, we present our system models in detail.

4.3 Preliminary

In this section, first, we discuss the architecture of the STT-MRAM cell, which is used for DNN parameter storage, followed by thermal impact on its switching probability. Then, we present the floating-point and fixed-point representations used while storing the data in memory. At last, we describe the DNN parameter error model due to thermal effects and random error model to correlate the thermal impact with switching probability.

4.3.1 DNN Architecture

We consider the DNN architecture described in Section 2.4 in Chapter 2.

4.3.2 Temperature Impact on STT-MRAM Cell

In this chapter, we assume that neural network parameters are stored in STT-MRAM and retrieved during the inference phase. In the future, STT-MRAM is seen as one of the favorable non-volatile memory technology for both standalone and embedded applications to serve as a universal memory option. It is a leading candidate to mitigate the problems between DRAM and secondary memory system [47] and last-level cache alternative to SRAM [157]. Compared with SRAM and DRAM, STT-MRAM is non-volatile, has negligible leakage power consumption, high endurance [158], and CMOS compatibility.

The structure of the STT-MRAM cell is as shown in Figure 4.1a. It is a three-terminal cell consisting of one N-channel MOSFET (NMOS) transistor and a magnetic tunnel junction (MTJ) device. The word line (WL) is used to access the cell. One of the remaining terminals is interfaced to the bit line (BL) and the remaining terminal to the source line (SL). The MTJ device is the fundamental element in the STT-MRAM cell to function as resistive storage. It has a free layer (FL) and pinned layer (PL) on two ends, and a tunnel barrier is sandwiched between them.

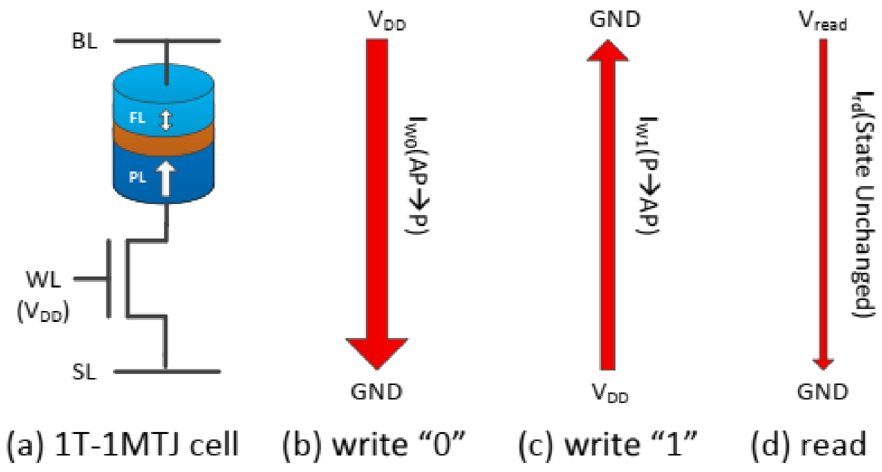


Figure 4.1: STT-MRAM cell with read-write current directions [134].

The magnetic alignment of the FL can be reversed; in contrast, that of PL can not be altered.

The STT-MRAM cell has two modes of operation: parallel (P) and anti-parallel (AP). The magnetic alignments of FL and PL are in the same direction in P-state, whereas they are in the reverse direction in the AP-state [47]. The word line and bit line are set to the supply voltage, and the source line is grounded to make the cell in P-state, facilitating the write '0' operation. Contrarily, the word line and source line are set to the supply voltage, and the bit line is at the ground to force the cell in AP-state, writing '1'. To read the state of the cell, a voltage V_{read} is applied across it. The current directions through the cell during write '0', write '1', and read operations are shown in Figure 4.1b-d, respectively, and the width of the arrows indicates proportionate current magnitudes during the operations. We have

$$I_{w0} > I_{w1} > I_{rd}, \tag{4.14}$$

where I_{w0}, I_{w1}, and I_{rd} are the currents during write '0', write '1', and read operations, respectively. For vehicle control operations, once the application design is finalized, the PUs are required during inference state, hence requiring only read operation of the memory, which has a very low magnitude of the current for STT-MRAM, resulting in less power consumption.

As a case study, we consider the emerging non-volatile memory technology, i.e., STT-MRAM, to store the DNN parameters. After the network is trained on the high-performance computing platform, it can be deployed in the low power accelerators or edge devices for inferencing [156]. For such accelerators, we assume that the parameters are stored in the STT-MRAM. Despite having low power, low latency, and no refreshing requirement, it suffers from thermal problems [159]. The thermal stability of STT-MRAM cell at the temperature (T) be

$$\Delta = \frac{E_b}{k_B * T}, \tag{4.15}$$

where E_b is the energy of the barrier and k_B is the Boltzmann constant. The data bits stored in the STT-MRAM cells flip with the probability

$$P_{sw} = 1 - e^{\left(-\frac{\tau_{pw}}{\tau_0} * e^{\left(-\Delta\left(1-\frac{I_{rd}}{I_{c0}}\right)\right)}\right)}, \tag{4.16}$$

where τ_{pw} is a pulse width of the read operation, τ_0 is the period, I_r is the read current, and I_{c0} is the switching current at 0 K. By using the parameters described in [160], we plot the equations (4.15) and (4.16) for increasing values of the temperature.

From Figure 4.2, we observe that the thermal stability of the STT-MRAM cell reduces with the temperature, which results in an exponential increase in the flipping probability. As the temperature increases from 30°C to 100°C, the flipping probability increases by 7328794.80%. This temperature range coincides with the typical vehicle operating temperature conditions [38]. It clearly indicates that the data stored in the STT-MRAM cell is subject to significant random variations at high temperatures. In the next section, we present the models for finding the average error of the parameters stored in STT-MRAM cells.

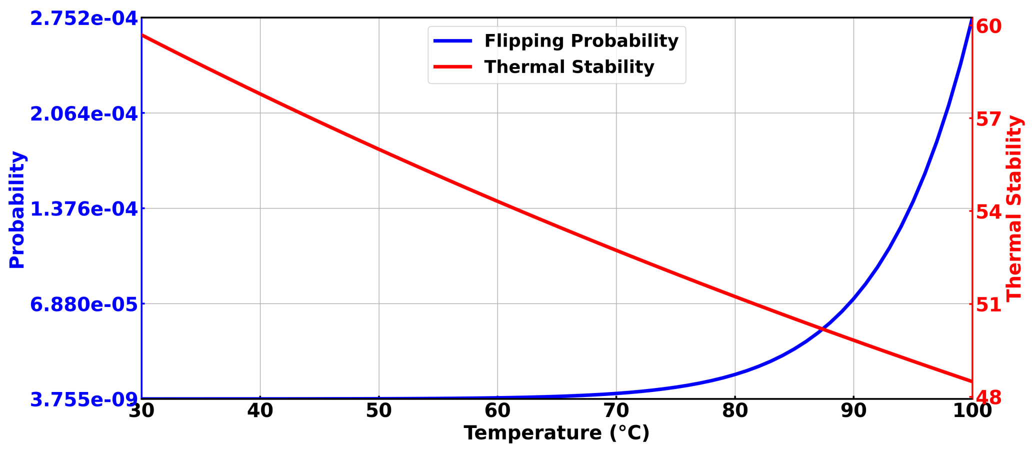


Figure 4.2: Temperature impact on STT-MRAM's Thermal Stability and Flipping Probability. As the temperature rises, there is a gradual decrease in the thermal stability, and flipping probability exponentially increases.

4.3.3 Floating-point and Fixed-point Number Representation

The DNN parameters are stored in STT-MRAM in binary format, but internally they follow a suitable representation technique for data processing. Floating-point and fixed-point number systems are widely used in computing platforms. The choice of a particular representation style depends on many factors such as precision and dynamic range of the numbers, performance, power, and cost.

The floating-point number representation is shown in Figure 4.3a. It has one sign bit ($s = 0$ for +ve and $s = 1$ for -ve), q exponent bits $(e_0, ..., e_{q-1})$ and p mantissa bits $(d_0, ..., d_{p-1})$. Generally, computing systems use IEEE 754 32-bit single-precision format for storing numbers, where $q = 8$ and $p = 23$. In general, for a 32-bit binary number, we can find its value as [161]

$$value = (-1)^s \cdot 2^{(E-127)} \cdot \left(1 + \sum_{i=0}^{22} 2^{22-i} \cdot d_i\right), \tag{4.17}$$

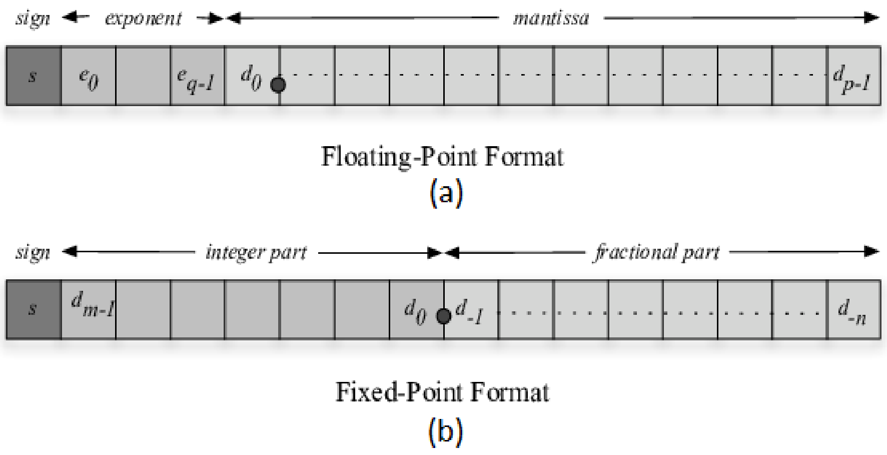


Figure 4.3: Fixed-point and floating-point number representation.

where $E = \sum_{i=0}^{7} 2^{7-i} \cdot e_i$. The floating-point numbers represent a wide range of numbers than the fixed-point counterpart. The maximum and minimum numbers in single-precision format are $\pm 3.4028235 \times 10^{38}$.

In a fixed-point number representation, signed numbers are stored in 2's complement format. The fixed-point number representation format is as shown in Figure 4.3b, where s is the sign bit (1-negative, 0-positive), m bits for the integer part, and n bits for the decimal part, such that $b = 1 + m + n$. Once the values of m and n are chosen, the position of the decimal point is fixed. The minimum and maximum values represented by fixed-point format are $-2^{b-1}/2^n$ and $(2^{b-1} - 1)/2^n$, respectively. If x is the binary number of length b, then we can find the value as [162]

$$value = 2^{-n} \left[-2^{b-1} \cdot x_{b-1} + \sum_{i=0}^{b-2} 2^i \cdot x_i \right]. \tag{4.18}$$

Different combinations of b and n are used to represent the fixed-point numbers while adopting them in the AI frameworks to balance the neural network's classification accuracy, power, and computing resources [163–165].

Since floating-point numbers render a large scale of numbers than the fixed-point, eventually, floating-point numbers introduce large errors when there are random bit flips in the data.

4.3.4 Errors for Parameter Representation

The deep neural network parameters are stored in the accelerator memory and retrieved during inference operation. The fixed-point numbers are preferred in the DNN accelerators due to energy saving capability [166]. Also, as shown in Figures 4.4 and 4.5 later in Section-4.4.1, there is a severe temperature impact on the accuracy of the DNN parameters stored as 32-bit floating-point numbers than 16-bit fixed-point. Therefore, in this work, we assume that the DNN parameters and activations use fixed-point numbers stored in STT-MRAM cells. The expectation of such i^{th} DNN parameter due to a bit flipping be

$$E(\epsilon_i) = \sum_{l=1}^{n_{word}} (w_{il}^* - w_i)\ P_{sw}\ (1 - P_{sw})^{l-1}, \tag{4.19}$$

where w_{il}^* is obtained by flipping the l^{th} bit of the parameter w_i, and n_{word} are the total number of bits in a fixed-point representation. Also, $w_i \in \boldsymbol{W}_{\mathcal{N}_a}$ and $\boldsymbol{W}_{\mathcal{N}_a}$ is a weight set associated with neuron $\mathcal{N}_a$. Here, we assume that only a single bit is flipped at a time for any DNN parameter, and as $P_{sw} \ll 0$, the $(1 - P_{sw})$ term can be safely ignored from the above equation. Therefore, we get

$$E(\epsilon_i) = \sum_{l=1}^{n_{word}} (w_{il}^* - w_i)\ P_{sw}. \tag{4.20}$$

Similarly, the expectation of the square of the i^{th} DNN parameter perturbation be

$$E(\epsilon_i^2) = \sum_{l=1}^{n_{word}} (w_{il}^* - w_i)^2\ P_{sw}. \tag{4.21}$$

The $E(\epsilon_i)$ and $E(\epsilon_i^2)$ are required in the sensitivity estimation framework. In the next section, we present a model to relate the bit flipping location with flipping probability.

4.3.5 Random Error Model

The temperature-dependent flipping probability of the STT-MRAM cell is described in equation (4.16). For a given DNN of N parameters with n_{word} bits per parameter, we generate random uniform distribution $U(0,1)^{N,n_{word}}$ [167]. The l^{th} bit of the i^{th} parameter is flipped, if

$$U[i,l] \leq P_{sw}. \tag{4.22}$$

According to the model, as the flipping probability increases with the temperature, more bit flips occur.

With all the system models discussed in this section, we present a motivation for sensitivity analysis and propose our sensitivity estimation framework in the next section.

4.4 Sensitivity Estimation

In this section, first, we study the impact of temperature-induced random bit flips in the STT-MRAM array on classification accuracy, followed by the proposed sensitivity estimation methodology.

4.4.1 Motivation

From Figure 4.2 in Section-4.3.2, we observe that the flipping probability increases exponentially with the temperature and results in the random variations in the

DNN parameters stored in the STT-MRAM. We use the model in Section-4.3.5 to study the effect of random parameter perturbations on classification accuracy for various deep neural network architectures and datasets. The experiment setup is in Section-4.5.1. We compare the temperature effect on the single-precision floating-point representation and 16-bit fixed-point representation.

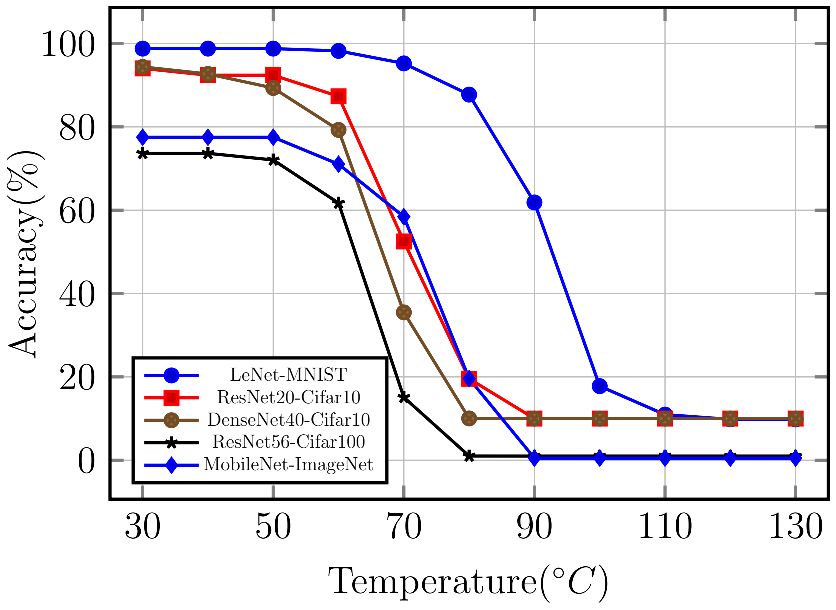


Figure 4.4: Effect of temperature on classification accuracy for parameters stored as a single-precision floating-point representation. The classification accuracy of all the DNNs under consideration drops sharply with increasing temperature.

As observed in Figure 4.4 and Figure 4.5, the DNNs sustain their baseline accuracy at lower temperatures. However, there is a sharp decrease in the accuracy as the temperature increases. Also, the inflection point of accuracy drop is unique for different networks, which may depend upon numerous factors, such as the type and number of layers, type of activation function, number of parameters, parameter number representation format, and input dataset. In comparison with the fixed-point format (Figure 4.5), the accuracy drops at lower temperatures for the single-precision floating-point format (Figure 4.4). This motivational example

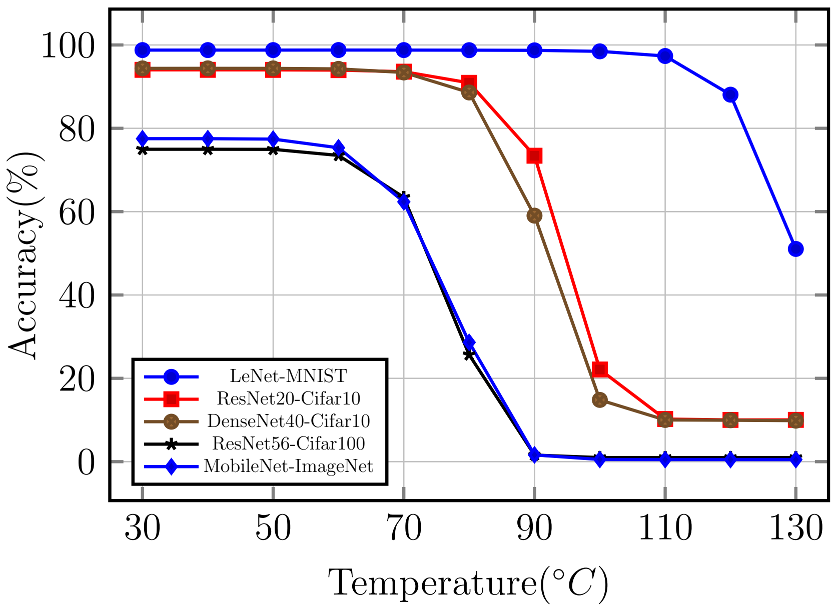


Figure 4.5: Effect of temperature on classification accuracy for parameters stored as a 16-bit fixed-point representation. The classification accuracy of all the DNNs under consideration drops sharply with increasing temperature.

clearly highlights: (i) single-precision floating-point numbers introduce large data errors resulting in sharp accuracy drop at the lower temperatures; (ii) the importance of DNN parameters' protection from thermal perturbations to maintain the classification accuracy, especially in automotives as their peak ambient temperature reaches $90°C - 150°C$ [38].

The accurate sensitivity estimation plays a crucial role in maintaining the baseline accuracy. As found in the existing works, we can find sensitivity at the bit, parameter, neuron, or layer level. The sensitivity estimation at bit level is too complex as it may have very large dimensions if the network size is huge, as shown in Figure 1.10 in Section- 4.1. Therefore, the next obvious choice is the sensitivity at the parameter level. However, ranking parameters based on parameter-level sensitivity may not guarantee the structural order when parameters are stored in the memory. Hence, the obvious choice is the sensitivity at the neuron level. In the following section, we present a more accurate sensitivity estimation framework based on the first-order and second-order components.

4.4.2 Neuron-level Sensitivity Estimation

The neuron is a basic computational element in the DNN. The errors in neuron outputs can propagate through subsequent network layers and may misclassify the input data. However, it is not necessary that all the neuron outputs can impact the classification accuracy. The sensitivity analysis reveals the potential of neuron perturbations on the classification accuracy. Hence, the precise sensitivity model is desirable. We formally define the neuron sensitivity as a definition 4.4.1.

Definition 4.4.1. *Let $\mathcal{C}(\boldsymbol{\mathcal{N}})$ is an original cost function of the deep neural network neuron set $\boldsymbol{\mathcal{N}}$ over the entire dataset. Let the parameter perturbations in a neuron $\mathcal{N}_a$ be $\boldsymbol{\epsilon}_{\mathcal{N}_a} \in \mathbb{R}^{s_a}$, and the cost function of the network due to a distorted neuron is $\mathcal{C}(\mathcal{N}_a + \boldsymbol{\epsilon}_{\mathcal{N}_a})$. By assuming other neurons remain unchanged, the sensitivity of the neuron be*

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| = |\mathbb{E}[\mathcal{C}(\boldsymbol{\mathcal{N}}) - \mathcal{C}(\mathcal{N}_a + \boldsymbol{\epsilon}_{\mathcal{N}_a})]|. \tag{4.23}$$

We use bold letters to represent vector/matrix if unless specified otherwise. In what follows, we analytically derive the sensitivity of the neuron in equation (4.23). By considering first-order and second-order terms of the Taylor expansion and neglecting higher-order terms [168], the modified cost function be

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}[\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot \mathcal{H}_{\mathcal{N}_a} \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a}]|, \tag{4.24}$$

where $\mathcal{G}_{\mathcal{N}_a}$ and $\mathcal{H}_{\mathcal{N}_a}$ represent the first-order and second-order derivatives of the $\mathcal{C}(\mathcal{N}_a)$, respectively, commonly known as the gradient vector and hessian matrix. As $\mathcal{H}_{\mathcal{N}_a}$ is a real symmetric matrix, it can be described as $Q^T\Lambda Q$, where Λ is a diagonal matrix with eigenvalues $(\lambda_1, \lambda_2, ..., \lambda_{s_a})$ as the diagonal elements in the descending order and the rows of Q are the corresponding eigenvectors $(q_1, q_2, ..., q_{s_a})$.

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}[\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot Q^T \Lambda Q \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a}]| \tag{4.25}$$

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}[\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}(Q \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a})^T \Lambda (Q \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a})]| \tag{4.26}$$

With $\boldsymbol{\psi}_{\mathcal{N}_a} = Q \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a}$, we have

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}[\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\boldsymbol{\psi}_{\mathcal{N}_a}^T \Lambda \boldsymbol{\psi}_{\mathcal{N}_a}]| \tag{4.27}$$

Since Λ is a diagonal matrix with eigenvalues in the declining order, we have $\lambda_1 \geq \lambda_2 \geq ...\lambda_{s_a}$. We can rewrite equation (4.27) as

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}[\boldsymbol{\epsilon}_{\mathcal{N}_a}^T \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\sum_{i=1}^{s_a} \psi_i^2 \lambda_i]| \tag{4.28}$$

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}(\boldsymbol{\epsilon}_{\mathcal{N}_a}^T) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\sum_{i=1}^{s_a} \mathbb{E}(\psi_i^2) \lambda_i| \tag{4.29}$$

As the rows of the Q are eigenvectors and $\boldsymbol{\psi}_{\mathcal{N}_a} = Q \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a}$, we have

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}(\boldsymbol{\epsilon}_{\mathcal{N}_a}^T) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\sum_{i=1}^{s_a} \lambda_i \ \mathbb{E}[(q_i \cdot \boldsymbol{\epsilon}_{\mathcal{N}_a})^2]| \tag{4.30}$$

After expanding the inner product of eigenvectors and parameter perturbations,

$$\begin{aligned} |\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}(\boldsymbol{\epsilon}_{\mathcal{N}_a}^T) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\sum_{i=1}^{s_a} \lambda_i \Bigg(\sum_{j=1}^{s_a} q_{ij}^2 \ \mathbb{E}(\epsilon_j^2) \\ + \sum_{j=1}^{s_a} \sum_{k=j+1}^{s_a} 2 \ q_{ij} \ q_{ik} \ \mathbb{E}(\epsilon_j) \ \mathbb{E}(\epsilon_k) \Bigg) | \end{aligned}$$

For a fixed-point number, $\mathbb{E}(\epsilon_j), \ \mathbb{E}(\epsilon_k) < 1$ and most of the elements of the eigenvectors, i.e., $q_{ij}, \ q_{ik} < 1$. Therefore, we get

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}(\boldsymbol{\epsilon}_{\mathcal{N}_a}^T) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\sum_{i=1}^{s_a} \lambda_i \sum_{j=1}^{s_a} q_{ij}^2 \ \mathbb{E}(\epsilon_j^2)| \tag{4.31}$$

We choose top-n eigenvalues as most of them are close to zero [169], where $n \ll s_a$.

$$|\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| \approx |\mathbb{E}(\boldsymbol{\epsilon}^T_{\mathcal{N}_a}) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2}\sum_{i=1}^{n}\lambda_i \sum_{j=1}^{s_a} q_{ij}^2 \ \mathbb{E}(\epsilon_j^2)| \tag{4.32}$$

As the neurons in a network have different sizes, we normalize, and finally, the sensitivity of the neuron is

$$\mathbb{S}^{\mathcal{N}_a} \equiv |\frac{1}{s_a}\mathbb{E}(\boldsymbol{\epsilon}^T_{\mathcal{N}_a}) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2s_a}\sum_{i=1}^{n}\lambda_i \sum_{j=1}^{s_a} q_{ij}^2 \ \mathbb{E}(\epsilon_j^2)| \tag{4.33}$$

The sensitivity estimation by equation (4.33) incorporates first and second-order terms of the Taylor expansion resulting in more accurate sensitivity of the DNN neurons than the state-of-the-art methods. The proposed method of neuron sensitivity requires finding the eigenpairs for each neuron. Obviously, higher the neuron's sensitivity, perturbations in its parameters will bring significant classification accuracy loss.

As later shown in Section-4.5.2, the proposed sensitivity metric performs better than the state-of-the-art methods for LeNet, ResNet20, ResNet56, and DenseNet40 networks; but it is inferior to MobileNet. For such cases, we present the effect of regularization on the proposed sensitivity metric in the following section.

4.4.3 Regularization Impact on Sensitivity

The sensitivity metric presented in the previous section considers Taylor approximation of the change in a cost function. The error terms in equation (4.24), i.e., $\boldsymbol{\epsilon}_{\mathcal{N}_a}$ should have low variance due to bit flipping for accurate approximation of the cost function. Otherwise, an inaccurate approximation may result in the developed sensitivity metric inconsistencies. In our case, regularization can play a critical role in lowering the error variance.

While model training, the model shows significant accuracy on the training dataset but performs poorly on the test dataset due to an overfitting problem. The regularization is widely applied to avoid overfitting problems, which helps to generalize the models [170, 171]. The L1 regularization and L2 regularization are commonly used to minimize the generalization error and overfitting [172]. The regularization modifies the cost function with a penalty term. In particular, for L1 regularization, the original cost function is modified as

$$cost\ function' = cost\ function + \alpha \sum_{i \in N} ||w_i||, \tag{4.34}$$

where α is a regularization parameter. Similarly, a cost function for L2 regularization is

$$cost\ function' = cost\ function + \alpha \sum_{i \in N} ||w_i||^2. \tag{4.35}$$

Since the objective of the training is to minimize the cost function, the α, which is a hyper-parameter, has a significant impact on the resulting trained DNN parameters. Therefore, the L1 regularization results in a sparse DNN, whereas the L2 regularization produces the DNN with parameter values close to zero.

Regularization penalizes the parameters with large values. Hence, the DNN parameter distribution narrows down. As a result, the parameter variance due to bit flipping given by equations (4.20) and (4.21) is minimized. Such narrowed parameter variance helps to achieve accurate Taylor approximation in the proposed sensitivity metric. Since the pre-trained MobileNet shows inferior performance for the proposed sensitivity metric, we train the MobileNet using L2 regularization and evaluate its impact on the sensitivity.

In the next section, we present our experiment setup and discuss the results in detail.

4.5 Experimental Results

In this section, first, we present the experimental set up followed by the results and observations.

4.5.1 Experimental Set-up

We use standard pre-trained deep neural networks, such as LeNet [173], ResNet20, ResNet56 [174], DenseNet40 [175], MobileNet [176] and standard datasets like, MNIST [177], Cifar10, Cifar100 [178], and ImageNet [179] to validate the proposed sensitivity estimation framework. The eigenpair computation of the entire network is computationally intractable. However, the approximation methods proposed in [139, 140] efficiently calculate the top-eigenpairs using randomized numerical linear algebra methods. These methods apply the oracle to calculate hessian without forming the hessian matrix using random vectors, which have time complexity the same as the back-propagation. We use the utility provided in [180] to calculate the neuron-level top eigenvalue and eigenvector pairs, which uses the hessian approximation algorithm in [139]. For each neuron, $n = 5$ top eigenpairs are considered, as they guarantee accurate approximation of the hessian matrix [63]. As there are few bias terms in the DNN architecture, we do not include them in the eigenvalue computation, and we assume that they are always protected from random perturbations. We consider the protection of 5% parameters associated with the most sensitive neurons to compare the accuracy of different sensitivity methods [63]. The four neuron-level sensitivity methods used for comparison are listed below:

- **Magnitude-based Sensitivity Estimation (Mag)**: This is an L1 norm-based sensitivity method [135]. The sensitivity is estimated as the absolute value of the mean of the DNN parameters in a neuron as:

$$\mathbb{S}_{mag}^{\mathcal{N}_a} = \frac{1}{s_a} |\sum_{i=1}^{s_a} w_i|, \quad w_i \in \boldsymbol{W}_{\mathcal{N}_a} \tag{4.36}$$

- **Gradient-based Sensitivity Estimation (Grad)**: In this approach, we consider only the first-order derivative term of the Taylor expansion [137] as:

$$\mathbb{S}_{grad}^{\mathcal{N}_a} = \frac{1}{s_a} |\mathbb{E}(\boldsymbol{\epsilon}_{\mathcal{N}_a}^T) \cdot \mathcal{G}_{\mathcal{N}_a}| \tag{4.37}$$

- **Hessian-based Sensitivity Estimation (Hess)**: The second-order derivative term of the Taylor expansion [63] is considered in this approach as:

$$\mathbb{S}_{hess}^{\mathcal{N}_a} = \frac{1}{2s_a} |\sum_{i=1}^{n} \lambda_i \sum_{j=1}^{s_a} q_{ij}^2 \; \mathbb{E}(\epsilon_j^2)| \tag{4.38}$$

- **Combined Sensitivity Estimation (Comb)**: It is the proposed sensitivity estimation framework consisting of the first-order and second-order derivative terms of the Taylor expansion, given by the equation (4.33).

We consider a signed 16-bit fixed-point number representation for the DNN parameters. Table 4.1 describes the number of integer and decimal bits used for various DNN architectures-datasets along with their baseline accuracy. To evaluate regularization impact on the developed sensitivity metric, we train the MobileNet on ImageNet dataset using L2 regularization with $\alpha = 1e^{-3}$. This trained network (MobileNet-ImageNet Reg) has lower baseline accuracy than the pre-trained MobileNet-ImageNet counterpart, as shown in Table 4.1.

In the following section, we present and discuss our experiment results in detail to validate the effectiveness of the proposed sensitivity method.

Table 4.1: Integer and decimal bits in the signed 16-bit fixed-point representation

Architecture-Dataset	Max. parameter	Min. parameter	Num. Int bits	Num. Dec bits	Baseline accuracy
LeNet-MNIST	1.143516	-1.528422	1	14	98.78%
ResNet20-Cifar10	1.666870	-1.472656	1	14	96.47%
DenseNet40-Cifar10	1.920898	-1.757446	1	14	94.39%
ResNet56-Cifar100	2.431884	-1.184448	2	13	75.12%
MobileNet-ImageNet	3.999877	-2.617187	2	13	77.51%(Top5)
MobileNet-ImageNet Reg	2.650756	-0.664550	2	13	69.80%(Top5)

4.5.2 Experimental Results and Discussion

From Figure 4.5 in Section-4.4.1, we observe that each DNN sustains the perturbations at a lower temperature range, and the region in which accuracy drops significantly is different for each of them. Accordingly, we choose different temperature ranges to study the effect of random bit flipping for each DNN. We collect classification accuracy results of 50 randomized trials for each temperature value for the sensitivity methods mentioned in Section-4.5.1.

For the temperature range of 120-150°C, as shown in Figure 4.6, the accuracy of the *Mag* [135] method is significantly lower than the remaining three methods for the LeNet-MNIST configuration. Whereas the accuracy due to the *Grad* [137] method is improved by 2-17% in comparison with the *Mag* for a temperature range of 120-140°C. For the same temperature range, the accuracy with the *Hess* method is improved by 2-18.3% compared with the *Grad* method. At last, the proposed

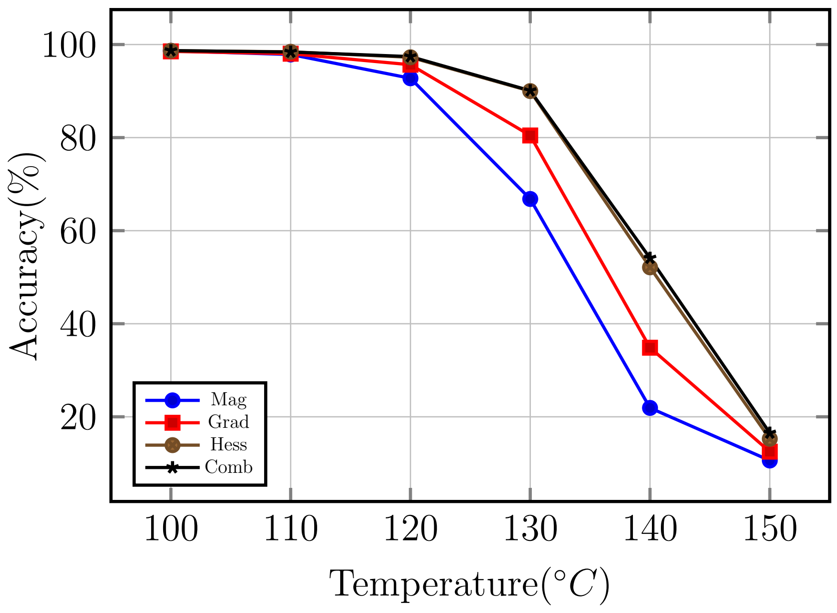


Figure 4.6: Accuracy with the protection of top-5% sensitive parameters for LeNet-MNIST.

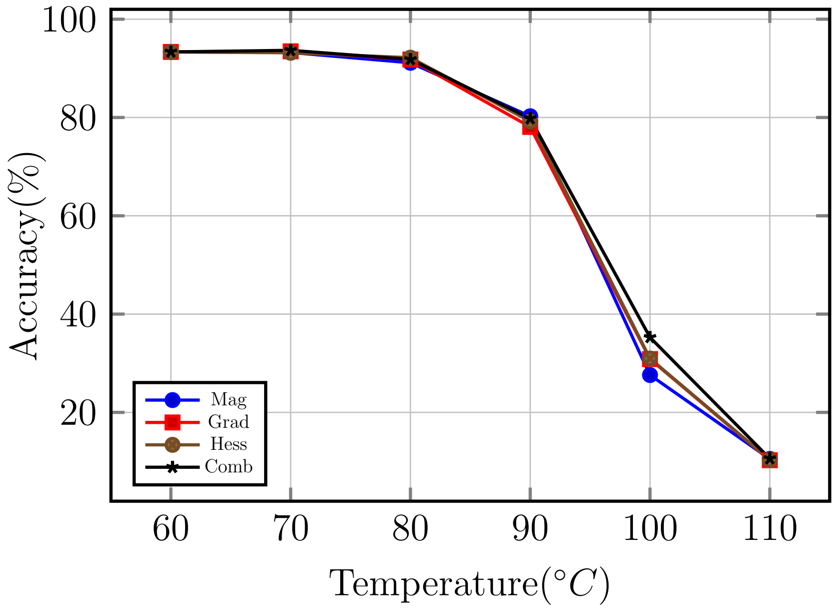


Figure 4.7: Accuracy with protecting top-5% sensitive parameters for ResNet20-Cifar10.

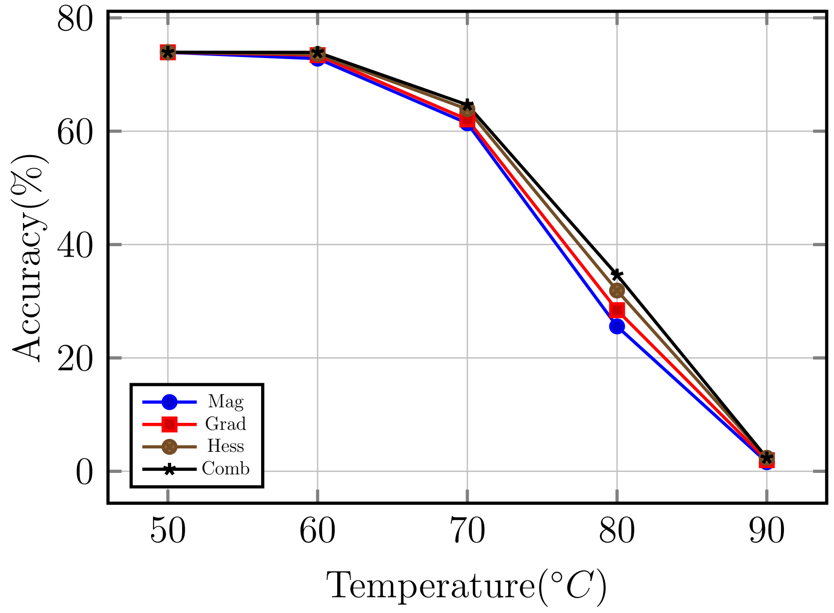


Figure 4.8: Accuracy with protecting top-5% sensitive parameters for ResNet56-Cifar100.

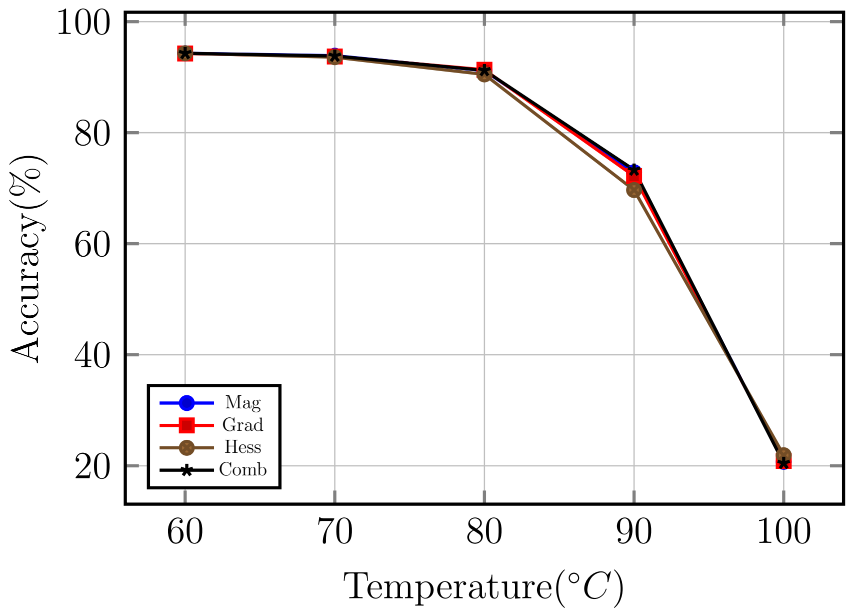


Figure 4.9: Accuracy with protecting top-5% sensitive parameters for DenseNet40-Cifar10.

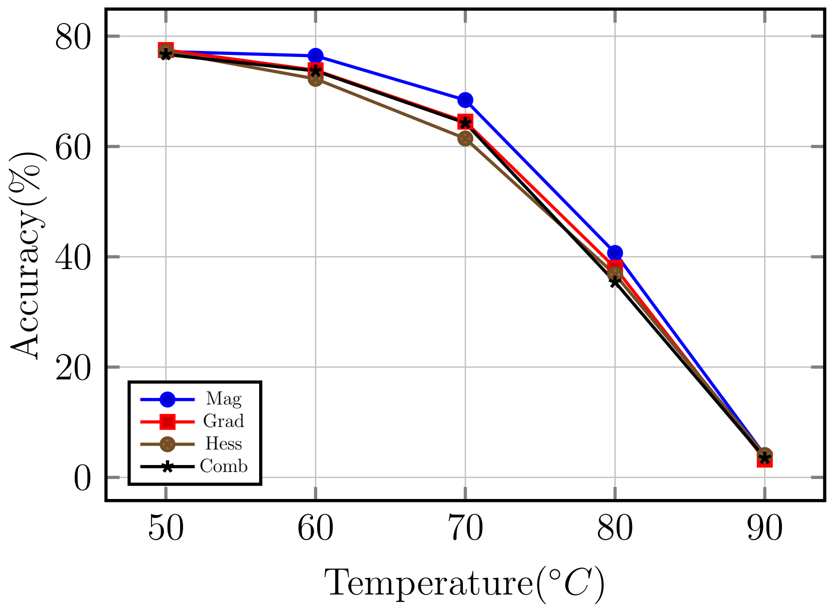


Figure 4.10: Accuracy with protecting top-5% sensitive parameters for MobileNet-ImageNet.

Comb method improves the classification accuracy by 0.5-2% for the temperature range of 130-150$^\circ C$ than the *Hess* [63] method.

Meanwhile, for ResNet20-Cifar10 configuration in Figure 4.7, all four methods perform roughly the same for the temperature below 80$^\circ C$. On the other hand, at 90$^\circ C$, all methods perform equally except *Grad* [137], which shows an accuracy reduction of 2%. For the temperature range higher than 90$^\circ C$, the proposed *Comb* method outperforms all other state-of-the-art methods by a maximum of 4.2% at 100$^\circ C$. We observe a similar performance improvement by the proposed *Comb* method for ResNet56-Cifar100 configuration in Figure 4.8 for the temperature range of 60-90$^\circ C$. Here, we get the accuracy improvement by 0.5-3.1% than the *Hess* [63] method. These results highlight the significance of the proposed sensitivity estimation framework at a high-temperature range.

For the configurations in Figure 4.6-4.8, overall, among the existing sensitivity estimation frameworks, *Hess* [63] performs better than the *Mag* [135] and *Grad* [137]

Table 4.2: Parameter Distribution

Architecture-Dataset	Total Weights	≤ -2	$[-2,-1]$	$[-1,0]$	$[0,1]$	$[1,2]$	$[2,3]$	$3 >$
LeNet-MNIST	61706	0	44	34009	27651	2	0	0
ResNet20-Cifar10	272474	0	10	149437	122969	58	0	0
DenseNet40-Cifar10	599050	0	19	314830	284161	40	0	0
ResNet56-Cifar100	861620	0	3	453344	408148	123	2	0
MobileNet-ImageNet	470072	6	118	253593	215776	312	88	179
MobileNet-ImageNet Reg	470072	0	0	324525	145384	103	60	0

methods. Unlike the previous three configurations, *Hess* performs poorly among the existing sensitivity estimation methods for DenseNet40-Cifar100 configuration in Figure 4.9, as low as 2.5% than the *Grad* at $90°C$. Whereas for DenseNet40-Cifar100, we observe 0.2-1% accuracy improvement by the proposed *Comb* method than the *Grad* [137] for a temperature range higher than $70°C$. This result verifies the fact that neither a first-order nor second-order derivative is always a good estimate for the sensitivity, and *Comb* methodology exploits the advantage of the first-order and second-order derivatives to achieve higher accuracy.

The MobileNet-ImageNet configuration in Figure 4.10 is an exception, where *Mag* [135] performs consistently better than the existing *Grad* [137], *Hess* [63], and the proposed *Comb* methods. The *Mag* method achieves an accuracy improvement with counterparts by 0.3-4.1% for the temperature range higher than $50°C$. Therefore, we analyze all neural networks' parameter and their variance distribution.

Table 4.3: Parameter Disturbance Distribution

Variance range	LeNet-MNIST	ResNet20-Cifar10	DenseNet40-Cifar10	ResNet56-Cifar100	MobileNet-ImageNet	MobileNet-ImageNet Reg
$[-8, -7]$	0	0	0	0	10	0
$[-7, -6]$	0	0	0	0	169	0
$[-6, -5]$	0	0	0	0	78	2
$[-5, -4]$	0	0	0	2	10	58
$[-4, -3]$	0	6	24	15	41	86
$[-3, -2]$	2	52	16	108	271	17
$[-2, -1]$	649	243	97	903	3124	190
$[-1, 0]$	27009	123202	284949	407960	219330	258898
$[0, 1]$	31881	148792	313856	452354	245825	210820
$[1, 2]$	2121	169	89	275	1090	1
$[2, 3]$	43	10	13	3	99	0
$[3, 4]$	1	0	6	0	19	0
$[4, 5]$	0	0	0	0	5	0
$[5, 6]$	0	0	0	0	1	0

As shown in Table 4.1, the MobileNet-ImageNet has maximum and minimum parameter values 3.999877 and -2.617187, respectively, which is considerably higher than the other networks. This results in a wide parameter distribution, as shown in Table 4.2, where the MobileNet-ImageNet configuration has a significant number of parameters higher than 1. Whereas all other networks have all the parameters roughly in a narrow range of [-1,1]. Consequently, this again broadens the parameter disturbance distribution, as shown in Table 4.3. Notably, the MobileNet-ImageNet configuration has a wide variance space compared with its counterparts. Such high values of the variances violate the condition of Taylor expansion and cause inconsistencies in the equations (4.33), (4.37), and (4.38), resulting in inaccurate sensitivity estimation. Hence, we observe that the *Mag* outperforms other methods in Figure 4.10.

To study the contradictory result of the MobileNet-ImageNet configuration, we perform sensitivity estimation on other neural network configurations by using single-precision floating-point numbers instead of 16-bit fixed-point representation under the same settings described in Section-4.5.1.

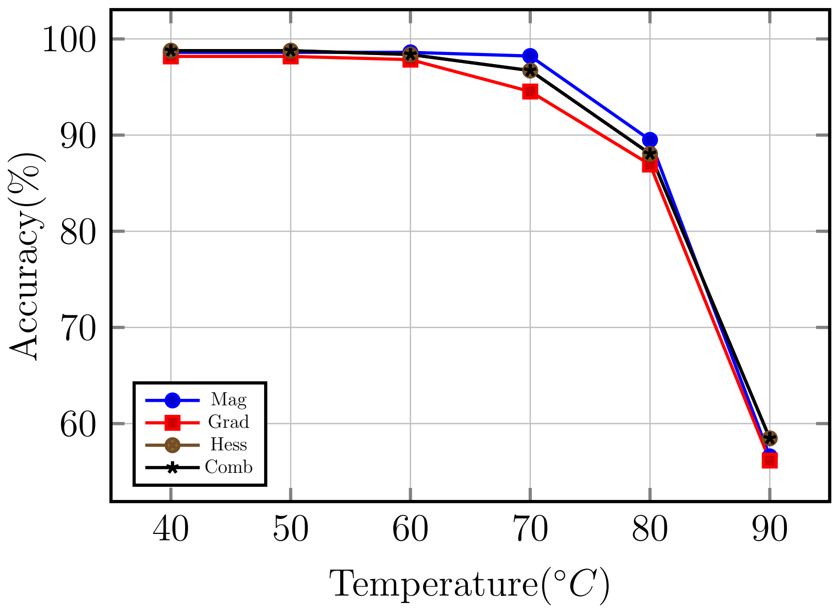


Figure 4.11: Accuracy with protecting top-5% sensitive parameters for LeNet-MNIST using single-precision floating-point representation.

As shown in Figures 4.11 and 4.13, for LeNet-MNIST and DenseNet-Cifar10 configurations, respectively, the *Mag* [135] performs better than all other sensitivity methods roughly for the entire temperature range of 40-90$^\circ C$. Whereas we observe an inconsistent accuracy trend for all the methods in Figure 4.12 for ResNet20-Cifar10 configuration. Also, the accuracy by *Hess* [63] and *Comb* methods is exactly the same for all temperature values. The reason is pessimistically large values of the average errors in equations (4.20) and (4.21), approximately in the range of 10^{37} and 10^{75}, respectively. These experimental results confirm the reason for *Mag* outperforming other sensitivity methods in MobileNet-ImageNet configuration (Figure 4.10).

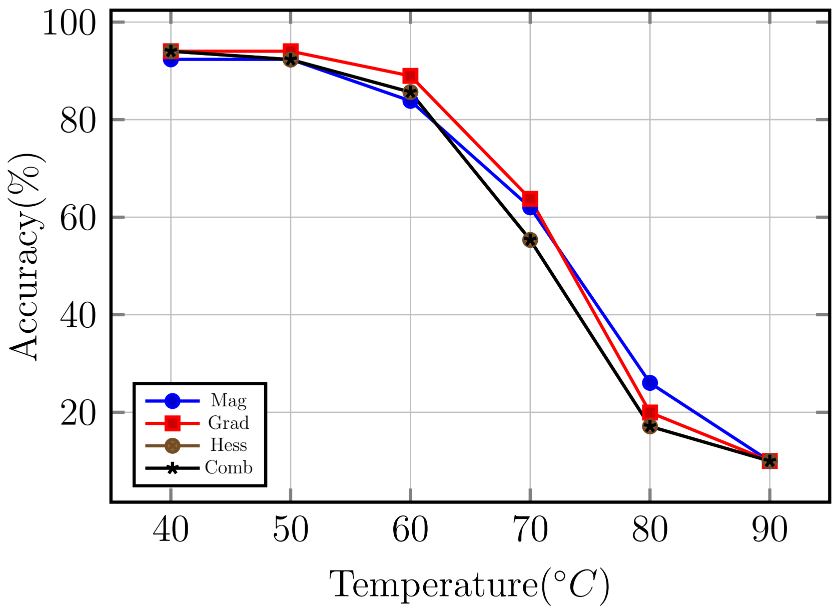


Figure 4.12: Accuracy with protecting top-5% sensitive parameters for ResNet20-Cifar10 using single-precision floating-point representation.

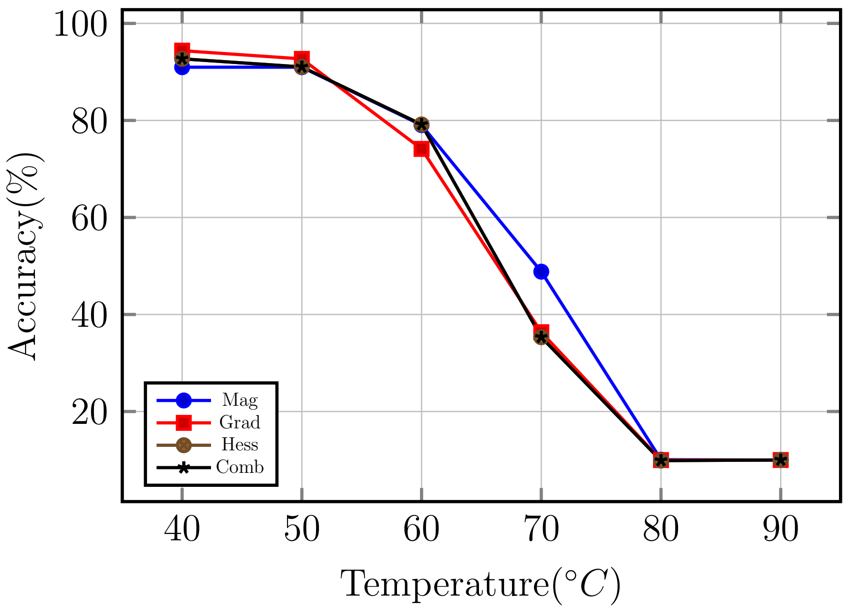


Figure 4.13: Accuracy with protecting top-5% sensitive parameters for DenseNet40-Cifar10 using single-precision floating-point representation.

Verification of Regularization Impact on Sensitivity

The MobileNet-ImageNet Reg has significantly narrower parameter distribution than the pre-trained MobileNet-ImageNet, as shown in Table 4.2. Such a narrowed parameter distribution for MobileNet-ImageNet Reg results in a narrower disturbance distribution than the pre-trained MobileNet-ImageNet, as shown in Table 4.3. We perform sensitivity estimation on MobileNet-ImageNet Reg configuration using 16-bit fixed-point representation under the same settings described in Section-4.5.1.

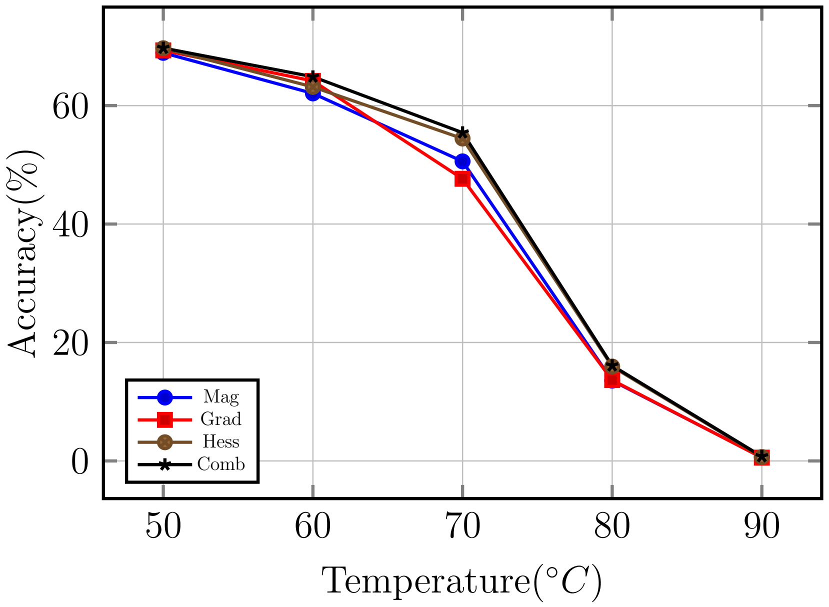


Figure 4.14: Accuracy with protecting top-5% sensitive parameters for MobileNet-ImageNet Reg.

As shown in Figure 4.14, for the temperature range of 50-80$^{\circ}C$, the *Hess* [63] method performs better than the *Mag* [135] by 1-3.8%. Whereas the *Grad* [137] method performs better than the *Mag* for the temperatures lower than 60$^{\circ}C$, but is inferior at higher temperatures. Moreover, the developed *Comb* performs better than the others for the entire temperature range of 50-80$^{\circ}C$. This result indicates that the developed sensitivity metric performs better than the state-of-the-art methods for low parameter disturbance distribution.

4.6 Conclusion

In this chapter, we first study the impact of temperature-induced random bit flipping in STT-MRAM cells on deep neural networks' classification accuracy. For DNNs under consideration, the accuracy gradually decreases with the rise in the temperature. In order to prevent accuracy loss, we require to protect the most sensitive neurons from thermal perturbations. Therefore, we propose a novel neuron-level sensitivity estimation methodology employing the first-order and second-order derivative information of the cost function. Our experimental results demonstrate that overall, the proposed method of sensitivity estimation improves the classification accuracy by 0.5-4.2% at a high-temperature range for LeNet, ResNet20, ResNet56 and DenseNet40 networks. Later, we present the limitation of the proposed sensitivity metric for MobileNet, where the wide parameter disturbance results in inferior performance.

CHAPTER 5

THERMAL AWARE MAPPING OF DEEP NEURAL NETWORK NEURONS ON 3D MEMORY OF ECS

The growing complexity and memory requirement of the deep neural networks introduce performance and reliability challenges for the processing units, which need to be addressed during AV design to ensure safety. Emerging 3D IC technologies, such as the hybrid memory cube (HMC), can assist in mitigating the memory constriction problem [181], but 3D ICs present significant thermal issues [182]. On the other hand, compared to DRAM, the emerging memory, such as STT-MRAM, has non-volatility features and low access latency, low power, and high density [47], desirable for alleviating memory problems due to the growing size of the DNNs. However, the classification accuracy of AI applications degrades at high temperatures when DNN parameters are stored in STT-MRAM, as observed in Chapter 4. Therefore, the thermal impacts need to be addressed during DNN application design for ECS to ensure safe and reliable AV operation.

In this chapter, we incorporate the thermal-aware sensitivity framework developed in Chapter 4 into the judicious mapping of DNN neuron parameters on memory banks for the ECS platform. First, we study the thermal impact problem on mapping DNN neuron parameters onto the memory banks. We employ the sensitivity metric developed in Chapter 4 and present a bin-packing-based strategy to map neuron parameters to memory banks with different temperature profiles to maximize the AI classification accuracy. From our experimental results, for state-of-the-art DNN architectures and datasets, we show significant improvement in the accuracy due to thermal aware mapping than thermal ignorant mapping. Second, given the constant energy budget for DNN execution, we develop a bin-packing and convex optimization-based approach to find the optimal temperature profile of the memory system, which maximizes the accuracy of AI applications. We compare the developed method with uniform temperature profiles generated by state-of-the-art workload distribution techniques [109, 183, 184] and a local search-based strategy to explore different temperature profiles based on uniform temperatures. From our experiment results, the developed method performs better than others at the low-temperature range. Besides, we observe inconsistent behavior of the developed method at high temperatures due to discrepancies in the sensitivity method. Also, we find that the uniform temperature profile of the memory system, which has been shown to be highly effective in reducing the peak temperature and power/energy consumption, is not necessarily an optimal way to maximize the AI application accuracy.

5.1 Introduction

Deep neural networks are the brain of autonomous vehicles. Their intelligence is developed during the training phase, where DNNs extract features automatically from the underlying data, which is distinct from other machine learning algorithms [23]. The artificial intelligence-enabled autonomous driving relies on DNNs for processing data from various sensors, like, RADAR, LiDAR, Imaging, etc., to control the vehicle maneuvers. DNNs are deployed in AVs to perform prime tasks, such as perception, localization, planning, control, etc., due to their ability to take complex decisions like the human brain [25]. Therefore, the performance of DNNs in terms of execution latency on the processing unit and classification accuracy is critical for safe and reliable vehicle operation.

High temperatures impact the DNN performance, especially with growing computational and memory capacity demand. The AVs substitute human beings with

DNNs for taking complex decisions on the road, and there is a stringent performance requirement for mission-critical tasks to ensure safety. Figure 1.9 of Chapter 1 shows that the DNN models are enormously complicated to achieve high accuracy. Similarly, as shown in Figure 1.10 of Chapter 1, the number of DNN parameters has scaled exponentially, but it is difficult for the traditional 2D memory system to accommodate them. Moreover, due to an increase in memory refresh rate, the performance of traditional memory, i.e., DRAM, suffers as the temperature rises [46]. And, the temperature of the ECS can vary as high as $90 - 155°C$ [38]. Therefore, exponentially growing memory capacity demand and high temperatures together can severely impact the performance of DNN execution.

The emerging non-volatile memory devices can be utilized to address the growing memory demand for DNN storage, but they are affected significantly by the temperature. The emerging memory devices, such as ReRAM and STT-MRAM, eliminate the requirement for refreshing while also providing high density and decreased read latency to alleviate the memory bottleneck problem [47]. In particular, for STT-MRAM, read access latency is reduced by 8.4% than the conventional DRAM [48]. However, temperature-induced random errors affect the emerging non-volatile memories. For example, the classification accuracy of AI applications running on ReRAM-based accelerators drops at high temperatures due to variations in cell conductances [49]. Similarly, as the temperature rises, the switching probability of STT-MRAM cells rises, lowering the classification accuracy of AI applications, as seen in the motivation example in Chapter 4. Therefore, it is crucial to incorporate thermal aspects of the memory system in ECS design to address the safety and reliability of AVs.

The 3D IC technology, such as the HMC, can help address the memory performance problem, but the thermal issues in 3D IC are well-known problems. The emerging 3D IC technology can assist in mitigating the performance problems caused due to memory bottleneck by bringing computation close to the memory, resulting in reduced data access time [52, 185]. For example, 3D IC based on emerging non-volatile STT-MRAM improves the performance by 34.74% than its equivalent 2D IC [186]. However, 3D IC limits the heat dissipation capability due to stacking of the layers [187]. Moreover, 3D ICs have high power density, resulting in high temperature and thermal gradients [42], which can degrade the reliability and even damage the chip. On the other hand, the memory bandwidth can be throttled to manage the operating temperature of the memory, resulting in performance reduction during run-time [182], which can be catastrophic for AVs.

In this chapter, we study two problems for mapping DNN neuron parameters on memory banks. First, we study the importance of thermal awareness for mapping DNN neuron parameters on memory banks to maximize the accuracy under a given temperature profile. To address this problem, we use the thermal-aware neuron-level *Comb sensitivity* method developed in Chapter 4 and bin-packing. Then, we compare the performance of the developed thermal-aware mapping method with the temperature oblivious mapping scheme. Second, given the constant energy utilization, we find the optimal temperature profile of the memory banks while judiciously mapping the DNN neuron parameters to maximize the accuracy. In particular, our contributions are listed as follows:

- We present a bin-packing-based algorithm to map the neuron parameters on memory banks utilizing a sensitivity method developed in Chapter 4 and the thermal profile of the memory banks. Our experiment results for the thermal-aware mapping algorithm improve the classification accuracy by 0.18-47.91% than the thermal oblivious method for popular networks, such as LeNet, Conv6, ResNet20, ResNet56, and DenseNet40, which highlights the significance of thermal awareness.

- For a given energy specification, the total temperature across the IC remains the same [188]. Hence, we develop a bin-packing and convex optimization-based method to find the optimal temperature profile of the memory banks while mapping DNN neuron parameters to maximize the accuracy under a constant energy consumption. We use a local search and uniform temperature-based methods to compare the performance of the developed method. Experiment results with state-of-the-art DNNs and datasets show that the developed method performs well at low temperatures. But, the developed method is inferior to exploring thermal profiles at high temperatures due to discrepancies in the sensitivity framework. Besides, we show that the uniform temperature profile of the memory banks is not an effective way to maximize AI application accuracy.

The rest of the chapter is organized as follows. In Section-5.2, we present related work. In Section-5.3, we present our preliminaries, i.e., DNN and 3D IC architectures, along with formulated problems. Then, we describe a method to study the importance of thermal awareness in mapping neuron parameters on memory banks in Section-5.4. Next, we present a convex optimization-based heuristic to find an optimal temperature map of memory banks to maximize the accuracy under constant energy budget in Section-5.5. We present our experiment set-up details and the results with discussion in Section-5.6, followed by a conclusion in Section-5.7.

5.2 Related Work

There are several works on neural network mapping for ReRAM based neuromorphic computing platforms, such as [49, 189–191]. Nevertheless, there is no work related to neural network neuron parameter mapping on the 3D memory banks to the best of our knowledge. However, the most closely related work in [49] considers the magnitude of the DNN parameters as a sensitivity metric to facilitate the thermal-aware mapping on the ReRAM crossbar. In this approach, the index tracking is cumbersome for large networks, as the parameter level sensitivity is considered. Moreover, magnitude is not the best choice of sensitivity metric, as later seen in the experiment results, which occasionally performs worse than the random estimate.

As a common practice, various algorithms for managing thermal problems in the PUs attempt to construct uniform temperatures across its processing cores [109, 183, 184]. For example, Sha et al. presented the M-oscillating method to maximize the performance by guaranteeing the peak temperature, which uniformly spreads the workload/power temporally [184]. The best way to reduce the peak temperature and enhance the reliability, which depends on temperature, is to keep the power density uniform across the processing unit, resulting in a uniform temperature profile. Moreover, the uniform temperature of the processing cores assists in minimizing the spatial and temporal thermal gradients, which improves the lifetime reliability of the PUs [109]. Therefore, in this chapter, we employ this analogy and study if the uniform temperature profile of the memory banks helps to maximize the AI application accuracy.

5.3 Preliminary

In this section, first, we present the deep neural network architecture and emerging 3D IC architecture in detail. Later, we formally define our problem statement.

5.3.1 DNN Architecture

We consider a deep neural network architecture presented in Section-2.4 of Chapter 2. Specifically, we use a class of convolutional neural networks, as the developed sensitivity metric in Chapter 4 can be applied to it. Moreover, training of the neural networks is computationally intense, requiring significant memory and energy [192]. Hence, due to limited resources for training inside AVs, we consider pre-trained neural networks are deployed in AVs [193] with their baseline accuracies readily available.

Tracking the DNN neurons simplifies the access complexity, enabling the ease of development of an accuracy maximization framework under thermal impact. The DNN is stored in the memory and later repeatedly retrieved during the inference phase. The sensitivity analysis is an instrumental tool to rank the DNN parameters/neurons based on their saliency, which can be used to protect the DNNs from accuracy degradation [63]. For such protection, when sensitivity is established at the parameter level, the index tracking becomes complex for large DNNs. On the other hand, index-tracking at the neuron level is simple [141], which facilitates structural ease of access, as all the parameters associated with a neuron are guaranteed to be in adjacent memory locations. Therefore, in this chapter, we consider the thermal-aware mapping of the DNNs on memory banks at the neuron level.

The structure of the neuron is described in Section-2.4 of Chapter 2, where the neuron set is $\boldsymbol{\mathcal{N}} = \{\mathcal{N}_1, \mathcal{N}_2, ...\mathcal{N}_{n_e}\}$ consisting n_e neurons of sizes $s_1, s_2, ..., s_{n_e}$, respectively. In particular, the size of the neuron depends on its associated number of parameters. Moreover, we consider a 16-bit fixed-point representation for parameters stored in the memory, as explained in Section-4.3.3 of Chapter 4. Accordingly, we define the memory footprint of the neuron as a number of bytes required for storage, as $2 * neuron\ size$.

5.3.2 3D IC Architecture

We consider the 3D IC architecture similar to that in [52, 194], which consists of a logic layer at the bottom and multiple memory banks on the upper layers. The logic layer is in contact with a heat sink. Furthermore, the inter-layer communication between processing cores in the logic layer and memory banks facilitates using through-silicon vias (TSVs). Without losing generality, we define the k^{th} 3D IC platform as

$$PU_k = \{\mathcal{P}; \mathcal{Y}|_{\mathcal{B}}\}, \tag{5.1}$$

where $\mathcal{P}$ be the processing cores at the logic layer, and $\mathcal{Y}$ be the number of memory layers, each containing $\mathcal{B}$ symmetric banks of size z bytes. Hence, the total number of memory banks be $\mathcal{Y} * \mathcal{B}$. We assume that the set of banks on consecutive vertical aligned memory layers form a vault. Thus, the 3D IC has $\mathcal{B}$ vaults, each containing $\mathcal{Y}$ memory banks. Also, we assume that the DNN neuron parameters are mapped on $\mathcal{Y}|_{\mathcal{B}}$, and data movement between processing cores ($\mathcal{P}$) and memory system ($\mathcal{Y}|_{\mathcal{B}}$) is not the focus of this research. However, we deal with finding temperature profiles maximizing the classification accuracy through thermal-aware mapping. The illustrative 3D IC architecture is shown in Figure 5.1, which has 16 processing cores at the logic layer, and there are 4 memory layers, each containing 16 memory banks. Therefore, it has 16 vaults, each having 4 memory banks.

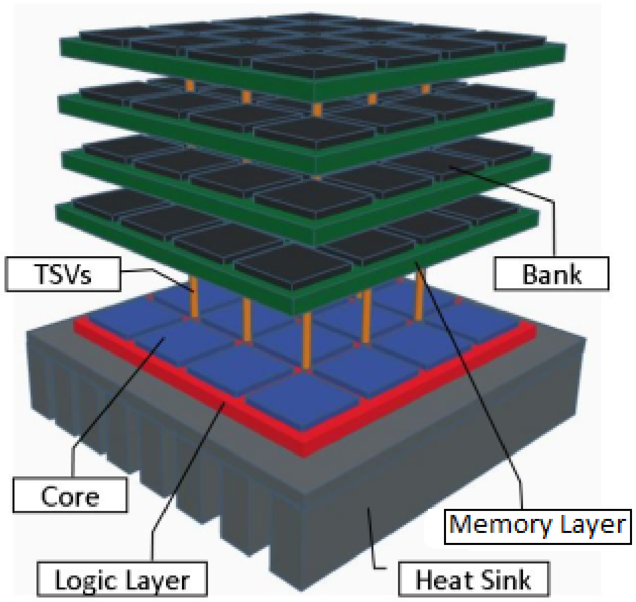


Figure 5.1: 3D IC Architecture [194]. The logic layer with processing cores is at the bottom, and memory banks are on the upper layers.

We consider the thermal behavior of 3D IC the same as in the existing work [194,195]. For a 3D platform, the thermal correlation is low in memory banks of the same layer. Their intra-layer thermal resistance is substantially higher than their inter-layer thermal resistance. Moreover, heat dissipation in the vertical direction is more significant than in the lateral direction [196]. Hence, we assume that memory banks in a particular vault have a similar temperature, and memory vaults have different temperatures. We consider the temperature of the memory banks in stable status as a temperature profile of the memory system.

After presenting our preliminaries, next, we formally formulate our problems.

5.3.3 Problems Formulation

To study the importance of thermal awareness while mapping DNN neuron parameters on memory banks, we assume that all neurons are accessed with the same frequency, which would make all memory bank accesses at the same frequency. Hence, different memory allocation strategies do not affect the temperature profiles. Also, we assume that the memory banks of $\mathcal{Y}|_{\mathcal{B}}$ have a unique temperature profile ($T_{\mathcal{Y}*\mathcal{B}}$) irrespective of the corresponding memory vault temperatures, and the mapping problem is formulated as:

Problem 5.3.1. *Given a deep neural network of neuron set $\boldsymbol{\mathcal{N}}$, memory system $\mathcal{Y}|_{\mathcal{B}}$, and a temperature profile $T_{\mathcal{Y}*\mathcal{B}}$, judiciously map all neuron parameters in $\boldsymbol{\mathcal{N}}$ to $\mathcal{Y}|_{\mathcal{B}}$ to maximize the classification accuracy of the AI application.*

For the next problem, we explore different temperature profiles, and to perform a fair comparison; we assume overall energy consumption is the same. Moreover, for a 3D IC platform with emerging non-volatile memory technologies, most of the energy is utilized in the computations in the logic layer [186,197]. Hence, we assume that memory access consumes negligible energy. Moreover, for 3D IC with fewer memory layers, the temperature of memory banks in a particular memory vault is approximately similar, and there is a significant temperature difference among the memory vaults [196, 198]. Therefore, we assume that the temperature of the memory vault is similar to the corresponding processing core, and we can distribute the workload among processing cores to change the thermal profile of the memory vaults. With these assumptions, we define the next problem as:

Problem 5.3.2. *Given a deep neural network of neuron set $\boldsymbol{\mathcal{N}}$ and memory system $\mathcal{Y}|_{\mathcal{B}}$, map all neuron parameters in $\boldsymbol{\mathcal{N}}$ to $\mathcal{Y}|_{\mathcal{B}}$, and find the optimal temperature profile of $\mathcal{Y}|_{\mathcal{B}}$ under a constant energy utilization, such that the classification accuracy of the AI application is maximized.*

In the next section, we present our approaches to deal with the thermal-aware mapping problems formulated above.

5.4 Thermal Aware Mapping of Neuron Parameters on Memory Banks

In this section, we consider the problem of mapping DNN neuron parameters on memory banks under different thermal profiles to maximize the accuracy of AI applications.

It is evident that the computations lead to thermal hotspots across the processing units [184, 199], including processing cores and memory systems, and their impact must be incorporated in the design of safety-critical applications. Additionally, 3D ICs elevate the thermal gradients in multitude due to high transistor density and chip geometry. In this chapter, we consider that the DNN is stored in the $\mathcal{Y}|_{\mathcal{B}}$ of PU_k, which has STT-MRAM cells for holding the data. As observed in the motivational example in Chapter 4, there is a significant degradation in the accuracy of the AI applications when temperatures rise. But, instead of a single temperature as in the motivational example, the temperatures are spatially different inside the PU_k, which can lead to a severe impact on the accuracy. Therefore, the thermal profile of the PUs needs to be considered while mapping DNN parameters on the memory banks.

Traditionally, mapping objects on available resources is an NP-hard problem, e.g., mapping the tasks on processing units for minimizing the latency [39, 107]. Problem 5.3.1 can be solved using various methods, including mathematical programming [107], simulated annealing [62], genetic algorithm [108], and convex optimization [85]. However, the computational cost of applying these methods to solve Problem 5.3.1 is prohibitively significant due to the massive size of DNNs required to achieve high accuracy [1]. Hence, we resort to a bin-packing-based heuristic to solve

Algorithm 2 Mapping of the DNN neuron parameters on memory banks.

```
    Inputs: Neuron set $\boldsymbol{\mathcal{N}}$ with their sizes; memory system $\mathcal{Y}|_{\mathcal{B}}$; the size of the
    memory bank $z$; temperature profile of the memory banks $T_{\mathcal{Y}*\mathcal{B}}$.
    Output: Mapping of $\boldsymbol{\mathcal{N}}$ on $\mathcal{Y}|_{\mathcal{B}}$; accuracy.
 1: $\mathbb{S}$: calculate the sensitivity of neurons in $\boldsymbol{\mathcal{N}}$;
 2: Rank the neurons based on $\mathbb{S}$ in descending order and store them in queue $Q$;
 3: Rank the memory banks in ascending order of their temperatures in $\mathcal{Q}$;
 4: for $n$ in $Q$ do
 5:     for $m$ in each memory bank of $\mathcal{Q}$ do
 6:         if $s_1 + ... + s_n \leq z$ then
 7:             Map the $n^{th}$ neuron parameters in $Q$ to $m$;
 8:         else
 9:             Choose the next memory bank;
10:         end if
11:     end for
12: end for
13: Evaluate the accuracy of the DNN.
```

Problem 5.3.1. We use Algorithm 2, employing the sensitivity method developed in Chapter 4, to map the neuron parameters on memory banks judiciously.

The errors of the parameters stored in STT-MRAM increase exponentially with the temperature, and temperature-induced errors in the sensitive neuron parameters significantly reduce the accuracy. Hence, we map the sensitive neuron parameters to the low-temperature memory banks to minimize the thermal impact in Algorithm 2. As shown in Algorithm 2, first, we calculate the sensitivity of neurons (line 1). In particular, we incorporate the sensitivity method (equation (4.33)) developed in Chapter 4 to estimate the sensitivity of the neurons using a maximum temperature of the memory system. Afterward, we rank the neurons in descending order based on sensitivity information (line 2). Similarly, in line 3, we rank the memory banks in ascending order of their temperatures. Then, we map the ranked neuron parameters on the ranked memory banks until the bank is filled up to its capacity using the first-fit bin-packing heuristic, otherwise select the next memory bank (line 4 - line 12). In this way, we map neurons to all the memory banks iteratively. At last, we evaluate the accuracy of the DNN with mapped neuron parameters on memory banks and corresponding temperature profiles (line 13). Line 2 and line 3 of the algorithm use a linear-complexity sorting algorithm. Hence, the complexity of Algorithm 2 is $O(n * m)$, where n and m are the number of neurons and number of memory banks, respectively. Algorithm 2 finds the mapping of neuron parameters under a given temperature profile, which may not be optimal to maximize the accuracy. Hence, in the next section, we present a convex optimization-based mathematical program and a heuristic to map the neuron parameters on memory banks, finding the optimal temperature profile of the memory system to maximize accuracy.

5.5 Optimal Temperature Profile of the Memory Banks

It is a common practice to distribute the workload among the processing cores of the PU and adjust their temperature profile to reduce the peak temperature, power consumption, and enhance the lifetime reliability [39,109,183,184]. In the case of 3D ICs with non-volatile memory, the processing cores consume most of the energy [186, 197]. Moreover, due to significant vertical heat dissipation [196], the temperature of the memory vault is similar to the corresponding core when the number of layers is less, and there is a significant temperature difference among the memory vaults [198]. Therefore, the temperature profile of the memory banks can be controlled by distributing the workload among the processing cores of the 3D platform. On the other hand, performing the exact computation does not always result in the same energy utilization. However, we are exploring different temperature profiles; hence, we assume that the total energy utilization is the same to perform a fair comparison. We use these notions to study Problem 5.3.2, i.e., finding the optimal temperature profile of the memory banks to maximize the accuracy of AI application execution under a constant energy budget. First, we present a convex-optimization-based mathematical program, and following, bin-packing and convex-optimization-based heuristic.

5.5.1 Convex Optimization-Based Approach

The formulated mapping Problem 5.3.2 is NP-hard in nature, and different approaches can be applied to solve it, such as mathematical programming [200], simulated annealing [62], genetic algorithm [108], and analytical approach [71]. In what follows, we incorporate the accuracy maximization of AI applications into the convex optimization-based mathematical program as it guarantees the optimal solution.

We use the thermal-aware sensitivity metric developed in Chapter 4 for this approach. For a^{th} neuron of size s_a, let the gradient vector of associated parameters be $\mathcal{G}_{\mathcal{N}_a}$, and the vector of the expected values of the parameter errors be $\mathbb{E}(\boldsymbol{\epsilon}^T_{\mathcal{N}_a})$. Similarly, we assume top-n eigenvalue (λ) and eigenvector (q) pairs for the neuron. If we have the expectation square of the neuron parameters as $\mathbb{E}(\epsilon^2)$, then the sensitivity of a^{th} neuron is

$$\mathbb{S}^{\mathcal{N}_a} \equiv |\frac{1}{s_a}\mathbb{E}(\boldsymbol{\epsilon}^T_{\mathcal{N}_a}) \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2s_a}\sum_{i=1}^{n} \lambda_i \sum_{j=1}^{s_a} q_{ij}^2 \ \mathbb{E}(\epsilon_j^2)|. \tag{5.2}$$

Note that only $\mathbb{E}(\boldsymbol{\epsilon}^T_{\mathcal{N}_a})$ and $\mathbb{E}(\epsilon_j^2)$ in equation (5.2) depends on the temperature, described as equations (4.20) and (4.21), respectively, in Chapter 4. We separate the temperature-independent terms of the equations (4.20) and (4.21), respectively, as

$$E(\epsilon_i) = \left(\sum_{l=1}^{n_{word}} (w_{il}^* - w_i)\right) \ P_{sw} = \Delta w_i \ P_{sw}, \tag{5.3}$$

and

$$E(\epsilon_i^2) = \left(\sum_{l=1}^{n_{word}} (w_{il}^* - w_i)^2 \right) \; P_{sw} = \Delta w_i^2 \; P_{sw}. \tag{5.4}$$

We rewrite equation (5.2) as

$$\mathbb{S}^{\mathcal{N}_a} \equiv |\frac{1}{s_a} \boldsymbol{\Delta w}_{\mathcal{N}_a}^T \cdot \mathcal{G}_{\mathcal{N}_a} + \frac{1}{2s_a} \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{s_a} q_{ij}^2 \; \Delta w_i^2| \; P_{sw}. \tag{5.5}$$

As the spatial temperature of a memory bank is the same, we have the same flipping probability, i.e., P_{sw}, for all neurons mapped on the corresponding memory bank. Moreover, the temperature of all memory banks in a vault is the same; hence, P_{sw} is the same for all neurons of a specific memory vault. If r neurons are mapped to the b^{th} memory bank, we define the overall change in the cost function of the memory bank as

$$|\mathbb{E}[\Delta \mathcal{C}(\boldsymbol{\mathcal{N}}^b)]| = \sum_{a=1}^{r} \mathbb{S}^{\mathcal{N}_a} = \Phi_b \; P_{sw}(T_b), \tag{5.6}$$

where Φ_b is the temperature-independent term obtained by adding sensitivities of all neurons mapped to the memory bank, and $P_{sw}(T_b)$ is the switching probability of b^{th} memory bank having temperature T_b.

Next, for neuron mapping on memory banks, we define the decision variables x_{ab} as

$$x_{ab} = \begin{cases} 1, & \text{if } \mathcal{N}_a \text{ is assigned to } \mathcal{B}_b; \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

Each neuron $\mathcal{N}_a$ must be mapped to only one memory bank as constrained by

$$\sum_{b=1}^{\mathcal{B}} x_{ab} = 1, \quad \forall \mathcal{N}_a \in \boldsymbol{\mathcal{N}}. \tag{5.8}$$

The size of neurons and memory banks can be considered as

$$\sum_{a=1}^{\boldsymbol{\mathcal{N}}} s_a * x_{ab} \leq z, \quad \forall b \in \mathcal{B}_b. \tag{5.9}$$

According to equation (5.6), the overall sensitivity of the memory banks with temperature-independent terms only is given as

$$\sum_{a=1}^{\boldsymbol{\mathcal{N}}} \mathbb{S}^{\mathcal{N}_a} * x_{ab} = \Phi_b, \quad \forall b \in \mathcal{B}_b. \tag{5.10}$$

The total energy consumption of the processing unit in the interval $[t_q,\ t_{q+1}]$ can be given as [188]

$$Energy_q = \Delta t_q\ \boldsymbol{\Psi} + \boldsymbol{\varphi} \sum_{i=t_q}^{t_{q+1}} T_i,$$

where $\boldsymbol{\Psi}$ and $\boldsymbol{\varphi}$ are processor-dependent constants, and Δt_q is the interval length. Therefore, the energy consumption of the PU depends on the total temperature. Contrarily, if the energy consumption is fixed, then we can infer that the total temperature of the PU must remain the same. Hence, under given energy consumption, we use the following constraint for the temperature profile of the memory banks

$$\sum_{b=1}^{\mathcal{B}} T_b = T_{tot}, \tag{5.11}$$

where T_{tot} is the overall temperature corresponding with constant energy consumption. To evaluate the accuracy of the AI applications under a certain temperature range, we restrict the memory bank temperatures using the following constraint

$$T_{min} \leq T_b \leq T_{max}, \quad \forall b \in \mathcal{B}_b, \tag{5.12}$$

where T_{min} and T_{max} are the minimum and maximum temperature values across the memory system. The goal of Problem 5.3.2 is to maximize the accuracy of AI applications, which can be accomplished by minimizing the change in a cost function under thermal impact. Therefore, we define the objective function as

$$minimize\ |\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| = \sum_{b=1}^{\mathcal{B}} \Phi_b * P_{sw}(T_b). \tag{5.13}$$

The convex optimization problem containing the constraints given by equations (5.8) to (5.12) provides a temperature profile of the memory banks, which is used to evaluate AI applications' accuracy under temperature-induced random bit flips. The computational cost of this approach is prohibitively high, as later found in Section-5.6.2. Hence, next, we develop a bin-packing and convex-optimization-based heuristic.

5.5.2 Bin-Packing and Convex Optimization-Based Heuristic

The application of convex optimization formulation to solve Problem 5.3.2 can provide an optimal solution, but it would be computationally intractable for large DNNs. Hence, we use a combination of the bin-packing heuristic and convex optimization to tackle Problem 5.3.2.

In particular, to reduce the computational complexity of the mapping problem, we employ the first-fit bin-packing strategy to map the neurons on different memory banks, described as Algorithm 2. For this algorithm, we arbitrarily assume the initial temperature profile of the memory banks and use the sensitivity metric developed in Chapter 4 by considering the maximum temperature. Then, we use convex optimization to find the optimal temperature profile of memory banks to maximize the AI application accuracy under a constant energy utilization described as the objective

$$minimize \; |\mathbb{E}[\Delta\mathcal{C}(\boldsymbol{\mathcal{N}})]| = \sum_{b=1}^{\mathcal{B}} \Phi_b * P_{sw}(T_b). \tag{5.14}$$

Since, Algorithm 2 provides mapping of $\boldsymbol{\mathcal{N}}$ on $\mathcal{Y}|_{\mathcal{B}}$, we can find Φ_b in equation (5.14) for each memory bank using equation (5.6). We use the following constraint to consider the effect of constant energy consumption on overall temperature dynamics of the memory banks as

$$\sum_{b=1}^{\mathcal{B}} T_b = T_{tot}. \tag{5.15}$$

Finally, we limit the minimum and maximum memory bank temperatures as

$$T_{min} \leq T_b \leq T_{max}, \quad \forall b \in \mathcal{B}_b. \tag{5.16}$$

The computational complexity of this amalgamated approach depends on Algorithm 2, as explained in Section-5.4, and the convex optimization model, which depends on the number of memory banks in a system as decision variables. The formulated convex optimization problem can find only the optimal temperature profile of the memory banks under the given mapping of the neurons. Therefore, the performance of this approach mainly depends on the bin-packing heuristic, i.e., Algorithm 2, which depends on the developed sensitivity metric.

After presenting our approaches to address the formulated problems, next, we discuss our experimental set-up and results in detail.

5.6 Experimental Results

In this section, first, we present the experimental set-up followed by the results and observations.

5.6.1 Experimental Set-up

We use state-of-the-art pre-trained deep neural networks, such as LeNet [173], Conv6 [201,202], ResNet20, ResNet56 [174], DenseNet40 [175], and standard datasets like, MNIST [177], Cifar10, Cifar100 [178] to evaluate the approaches for thermal aware mapping problems. We consider that numbers are stored in the memory in a 16-bit fixed-point format as described in Chapter 4. To better understand the thermal

Table 5.1: Memory Bank Size (z) for DNNs

Architecture-Dataset	Total Parameters	Bank Size (Bytes)	$T_{max}(^{\circ}C)$
LeNet-MNIST	61706	31300	145
Conv6-Cifar10	2262602	1140000	120
ResNet20-Cifar10	272474	136400	110
DenseNet40-Cifar10	599050	300000	100
ResNet56-Cifar100	861620	431000	90
DenseNet40-Cifar100	622360	311500	100

behavior, we consider that the memory system has one layer, i.e., $\mathcal{Y} = 1$, and it has $\mathcal{B} = 4$ memory banks, hence, 4 memory vaults. Accordingly, we use the symmetric size of the memory banks (z), which is different for each DNN architecture, as shown in Table 5.1. We choose minimum temperature as $T_{min} = 25$, but different maximum temperature (T_{max}) for each DNN architecture, as shown in Table 5.1, to cover its entire range according to Figure 4.5 of Chapter 4. We use additional settings of the sensitivity method from Section 4.5.1 of Chapter 4.

We use the following approaches for comparing the thermal impact on mapping of DNN neuron parameters to memory banks while maximizing the AI application accuracy:

- **Thermal Aware Mapping using Combined Sensitivity (TA-Comb)**: This approach utilizes Algorithm 2 to map the neuron parameters on memory banks, which employs the sensitivity method developed in Chapter 4.
- **Thermal Aware Mapping using Magnitude Sensitivity (TA-Mag)**: It uses a state-of-the-art magnitude-based sensitivity method [49] in Algorithm 2 (line 1) for mapping neuron parameters on memory banks.
- **Thermal Oblivious Random Mapping (TO-Rand)**: In this approach, the neuron parameters are mapped randomly to the memory banks without considering the thermal impact. In particular, we randomly generate the num-

bers between the interval $[0, 1]$ as a sensitivity of the neurons in line 1 of Algorithm 2, which are further mapped on the memory banks using Algorithm 2.

On the other hand, we use the following approaches for analyzing the performance of bin-packing-based convex optimization, which finds the temperature profile of the memory banks while mapping DNN neuron parameters to maximize the accuracy under constant energy consumption:

- **Bin-packing-based Convex Optimization Method (CVX)**: It uses the framework in Section 5.5, which maps the neuron parameters using a bin-packing algorithm and finds optimal temperature profile using convex optimization.
- **Uniform Temperature Method (UNI)**: We utilize a general phenomenon of workload distribution across processing cores to generate uniform temperatures [109, 183, 184]. We apply a bin-packing algorithm to map the neuron parameters on memory banks by assuming a uniform temperature profile under a constant energy consumption constraint (T_{tot}), such that equation (5.11) is satisfied.
- **Local Search Method (LOC)**: We develop a heuristic based on a local search strategy [203]. First, we use a bin-packing algorithm to map the neuron parameters on memory banks by assuming a uniform temperature profile under a constant energy consumption constraint (T_{tot}). Next, we randomly explore the temperature profiles in the neighborhood of the uniform temperature profile to maximize the accuracy of the AI applications, satisfying constant energy consumption.

We use PyTorch 1.8.1 framework [204] to set up the experiments and Pyomo [205, 206] to evaluate the convex optimization-based model. We collect classification accuracy results of 50 randomized trials for AI applications, and average accuracy values are reported if not specified explicitly. In the following section, we present and discuss our experiment results in detail to validate the developed methods to tackle the formulated problems.

5.6.2 Experimental Results and Discussion

In this section, first, we verify the effectiveness of thermal-aware mapping of neuron parameters on memory banks to maximize classification accuracy. Then, we evaluate the method of finding the optimal temperature profile for mapping DNN neuron parameters on memory banks to maximize the accuracy under constant energy utilization.

Verification of Thermal Aware Mapping of Neuron Parameters on Memory Banks

The ECS temperature can vary as high as $90 - 150°C$, and emerging 3D ICs have large thermal gradients, which together can trigger random bit flipping of the data in the memory cells of the STT-MRAM. To apply this thermal impact on the mapping problem, first, we randomly generate temperatures of the memory banks in the interval $[T_{min}, T_{max}]$. Then, under these random temperatures, we evaluate the accuracies of the methods for solving Problem 5.3.1, i.e., *TA-Comb*, *TA-Mag*, and *TO-Rand*.

As shown in Table 5.2, for LeNet-MNIST configuration, the *TA-Comb* method outperforms *TA-Mag* [49], and *TO-Rand*, respectively, in the range 0.25-47.91%

and 0.21-39.64%. Similarly, for Conv6-Cifar10 configuration in Table 5.3, *TA-Comb* performs better than *TA-Mag* [49] and *TO-Rand* by 0.11-44.52% and 0.1-43.9%. We observe a similar trend for ResNet20-Cifar10 in Table 5.4, where *TA-Comb* consistently performs better than *TA-Mag* [49] and *TO-Rand* by 4.3-32.08% and 5.61-30.78%. Whereas, for DenseNet40-Cifar10 configuration in Table 5.5, *TA-Comb* achieves superior accuracy than *TA-Mag* [49] and *TO-Rand* in the range 0-1.99% and 0.01-10.87%, respectively. Also, *TA-Comb* achieves superior accuracy than *TA-Mag* [49] and *TO-Rand* by 3.87-12.94% and 1.22-10.9% for ResNet56-Cifar100 in Table 5.6. Moreover, the DenseNet40-Cifar100 configuration reflects the same phenomenon in Table 5.7, where *TA-Comb* performs better than *TA-Mag* [49] and *TO-Rand* by 0.72-14.45% and 0.18-17.95%, respectively. As *TA-Comb* consistently outperforms the *TO-Rand* for considered DNNs, it highlights the importance of thermal awareness while mapping neurons on memory banks for 3D ICs.

The performance of three testing candidates, i.e., *TA-Comb*, *TA-Mag*, and *TO-Rand*, depends on the sensitivity method and temperature of the memory banks. As *TA-Comb* significantly performs better than *TA-Mag* [49] in Table 5.2-5.7, it validates the superior accuracy of the sensitivity method developed in Chapter 4 than the state-of-the-art magnitude sensitivity. Algorithm 2 maps the sensitive neuron parameters of the DNN to the coolest memory banks, which helps to minimize the thermal impact for sensitive neurons, resulting in accuracy maximization. As observed in Tables 5.2-5.7, when temperatures of the memory banks are high, we can observe significant improvement in the accuracy by the *TA-Comb* method, as high as 47.91% for LeNet-MNIST. But, when the temperature of the memory banks are low, there are very few random perturbations of the data, which may not be enough to pinnacle the importance of thermal awareness as observed in highlighted cells

Table 5.2: LeNet-MNIST accuracy with the random temperature profile of the 4 memory banks

Temperatures($^\circ C$)	**TA-Comb Accuracy(%)**	**TA-Mag Accuracy(%)**	**TO-Rand Accuracy(%)**
[30, 35, 79, 145]	85.56	37.65	45.92
[28, 33, 59, 134]	96.99	83.42	85.99
[50, 65, 96, 104]	98.72	98.47	98.51
[27, 96, 122, 125]	97.70	95.14	92.70
[88, 94, 111, 117]	98.47	97.35	96.92
[55, 94, 135, 142]	79.80	44.80	39.87
[63, 72, 101, 141]	92.13	56.09	63.26
[65, 110, 110, 141]	91.62	54.95	60.13
[69, 107, 126, 140]	89.99	56.99	60.58
[63, 107, 134, 138]	86.69	59.02	56.12

of Tables 5.5 and 5.6, where *TA-Mag* [49] and/or *TO-Rand* performs better than *TA-Comb* method.

Table 5.3: Conv6-Cifar10 accuracy with the random temperature profile of the 4 memory banks

Temperatures($^\circ C$)	**TA-Comb Accuracy(%)**	**TA-Mag Accuracy(%)**	**TO-Rand Accuracy(%)**
[28, 33, 59, 96]	85.77	84.73	84.19
[30, 35, 72, 79]	86.10	85.99	86.00
[42, 55, 64, 97]	85.69	84.49	83.49
[55, 65, 94, 104]	84.68	78.15	76.79
[50, 69, 96, 107]	84.06	73.24	71.27
[27, 88, 94, 117]	76.34	41.67	27.10
[26, 27, 99, 118]	75.18	35.63	21.69
[80, 90, 92, 101]	83.98	79.11	77.95
[63, 63, 101, 107]	83.71	68.59	65.37
[65, 110, 110, 111]	62.48	17.96	18.58

Table 5.4: ResNet20-Cifar10 accuracy with the random temperature profile of the 4 memory banks

Temperatures($^{\circ}C$)	**TA-Comb Accuracy(%)**	**TA-Mag Accuracy(%)**	**TO-Rand Accuracy(%)**
[42, 55, 90, 97]	88.86	76.43	75.52
[41, 44, 64, 93]	92.67	88.37	87.06
[28, 33, 59, 96]	92.19	84.43	83.16
[55, 65, 94, 104]	75.89	54.97	51.44
[50, 69, 96, 107]	66.07	35.50	37.11
[27, 88, 94, 110]	48.09	23.87	24.45
[35, 65, 79, 110]	60.81	28.73	30.03
[30, 63, 72, 101]	88.33	73.09	72.25
[26, 63, 99, 107]	62.56	33.02	33.32
[27, 80, 92, 101]	82.51	65.89	63.16

Table 5.5: DenseNet40-Cifar10 accuracy with the random temperature profile of the 4 memory banks

Temperatures($^{\circ}C$)	**TA-Comb Accuracy(%)**	**TA-Mag Accuracy(%)**	**TO-Rand Accuracy(%)**
[50, 55, 65, 94]	90.34	88.35	86.34
[28, 33, 69, 96]	88.46	86.56	82.04
[27, 59, 88, 96]	83.23	82.10	77.89
[35, 65, 79, 94]	89.20	87.57	85.18
[30, 63, 63, 72]	94.11	94.07	94.18
[26, 27, 92, 99]	73.39	72.81	62.52
[42, 55, 80, 90]	90.79	**91.02**	89.84
[44, 64, 93, 97]	75.33	74.89	68.92
[38, 41, 53, 59]	94.39	94.39	94.38
[34, 53, 73, 76]	93.91	93.89	93.60

Table 5.6: ResNet56-Cifar100 accuracy with the random temperature profile of the 4 memory banks

Temperatures($^{\circ}C$)	**TA-Comb Accuracy(%)**	**TA-Mag Accuracy(%)**	**TO-Rand Accuracy(%)**
[50, 55, 65, 83]	58.86	45.92	51.26
[27, 28, 33, 59]	74.55	**75.03**	**74.86**
[35, 65, 79, 88]	32.09	22.12	25.10
[30, 63, 63, 72]	70.06	64.86	68.82
[26, 27, 80, 90]	25.62	13.57	18.97
[27, 28, 33, 84]	57.81	46.72	49.64
[42, 44, 59, 76]	69.28	62.75	67.65
[34, 53, 73, 76]	66.21	56.18	61.58
[38, 41, 63, 85]	52.80	40.28	41.90
[25, 27, 43, 72]	71.96	68.09	70.74

Table 5.7: DenseNet40-Cifar100 accuracy with the random temperature profile of the 4 memory banks

Temperatures($^{\circ}C$)	**TA-Comb Accuracy(%)**	**TA-Mag Accuracy(%)**	**TO-Rand Accuracy(%)**
[50, 55, 65, 94]	37.80	23.35	19.85
[28, 33, 69, 96]	28.07	16.14	13.37
[27, 59, 88, 96]	17.58	9.66	7.46
[35, 65, 79, 94]	33.60	20.22	16.54
[30, 63, 63, 72]	73.06	72.34	72.50
[26, 27, 92, 99]	8.18	4.58	3.42
[42, 55, 80, 90]	45.47	34.86	30.46
[44, 64, 93, 97]	8.56	5.15	3.73
[38, 41, 53, 59]	74.94	74.79	74.76
[34, 53, 73, 76]	69.15	67.41	67.66

Verification of Methods for Finding Optimal Temperature Profile to Maximize the Accuracy

First, we check the computational cost of the convex optimization-based mathematical program. Figure 5.2 shows the CPU time required to solve the convex optimization mathematical program described in Section-5.5.1 for ResNet20-Cifar10 configuration. As shown in the figure, the computational cost increases exponentially with increasing the number of memory banks; hence, difficult to solve large-size problems.

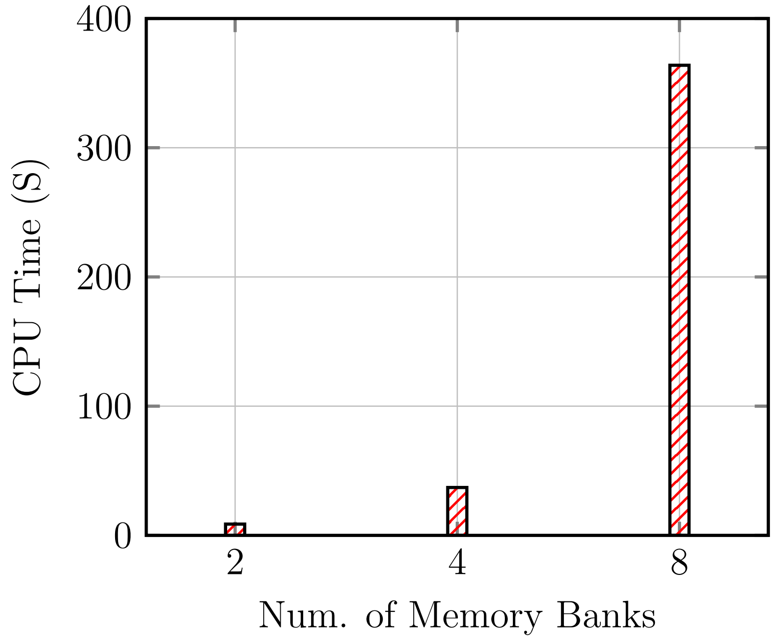


Figure 5.2: Computational cost of the Convex Optimization Approach for ResNet20-Cifar10. The computational cost increases exponentially with the number of memory banks.

Next, we evaluate the performance of the bin-packing and convex optimization-based heuristic for solving Problem 5.3.2 by increasing the T_{tot}. However, from Figure 4.5 of Chapter 4, different neural network architectures have different inflection points, where accuracy drops suddenly from their baselines; accordingly, they have different temperature ranges to evaluate the performance. Hence, we choose different T_{tot} for different DNNs and evaluate their accuracies. For example, in Figure 5.3 for LeNet-MNIST configuration, T_{tot} is indicated on the x-axis, in the interval $[300°C, 500°C]$ with increments of $40°C$. Similar reasoning applies to other architectures in Figures 5.4-5.8.

As shown in Figure 5.3 for LeNet-MNIST configuration, at lower T_{tot}, the CVX , *UNI* [109, 184], and *LOC* [203] perform marginally the same. At $T_{tot} = 500°C$, the *LOC* outperforms the uniform temperature method, i.e., *UNI*, by 9.74%. For Conv6-Cifar10 in Figure 5.4, CVX performs well for $T_{tot} \leq 260°C$. For $T_{tot} \geq 300°C$, *UNI* [109, 184] performs better than CVX except for $T_{tot} = 420°C$, where CVX outperforms *UNI* [109,184] by 4.5%. This result shows that the uniform temperature profile does not always guarantee better accuracy than the developed heuristic, i.e., CVX.

We obtain a similar trend for ResNet20-Cifar10 in Figure 5.5 as other configurations, where CVX performs well for $T_{tot} \leq 300°C$. But, for $T_{tot} \geq 340°C$, the *UNI* [109, 184] performs better than CVX by 2.94-13.45%. As shown in Figure 5.6, the CVX performs well for $T_{tot} \leq 240°C$. But, a local search, i.e., *LOC* [203], performs better than CVX and *UNI* by 0.7-15.12% and 0.7-2.56% for $T_{tot} = [280°C, 360°C]$. On the other hand, *LOC* performs inferior to CVX and *UNI* [109, 184] by 1.16% at $T_{tot} = 400°C$. This result highlights the randomness of the *LOC* [203] method.

Similarly, for the DenseNet40-Cifar100 configuration in Figure 5.7, CVX performs better than others by 0.6% for $T_{tot} \leq 230°C$. As the temperature increases, the CVX is inferior to others due to inconsistencies in the developed sensitivity metric. Whereas CVX performs marginally the same as others for a lower temperature range, i.e., $T_{tot} \leq 240°C$, and *LOC* [203] performs better than *UNI* [203] by 0.5-1.8% at $T_{tot} \geq 280°C$ for ResNet56-Cifar100 in Figure 5.8.

Overall, the uniform temperature-based method, i.e., *UNI*, does not guarantee high accuracy, which is a contradictory result to the existing thermal-aware methods, where it assists in minimizing the peak temperature, power, and improving the reliability [109,183,184]. On the other hand, the developed bin-packing-based convex optimization method, i.e., *CVX*, performs comparable at the low-temperature range but is inferior at high temperatures due to discrepancies in the sensitivity method developed in Chapter 4.

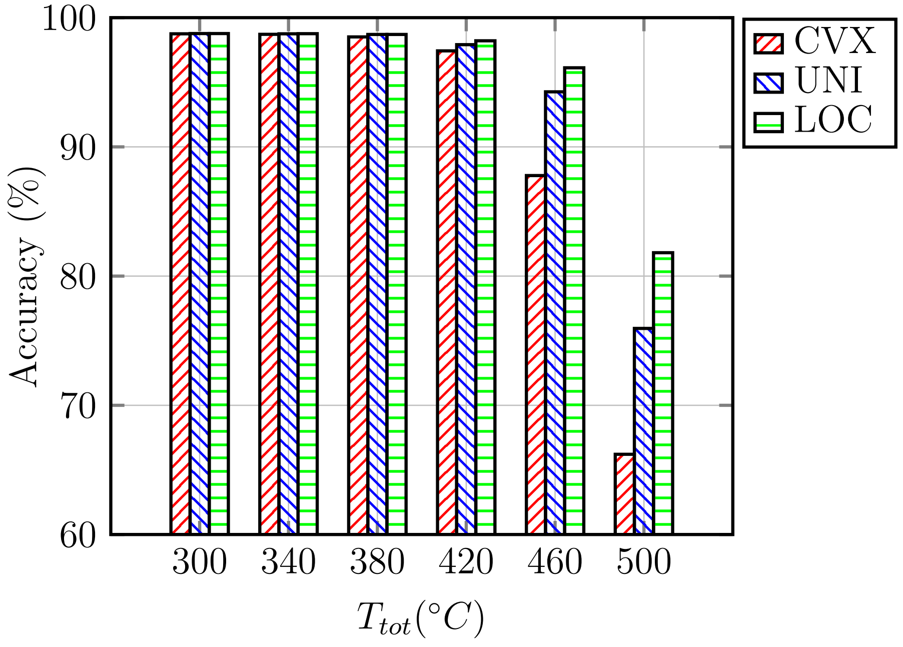


Figure 5.3: Accuracy of LeNet-MNIST for increasing values of the total temperature of the memory system.

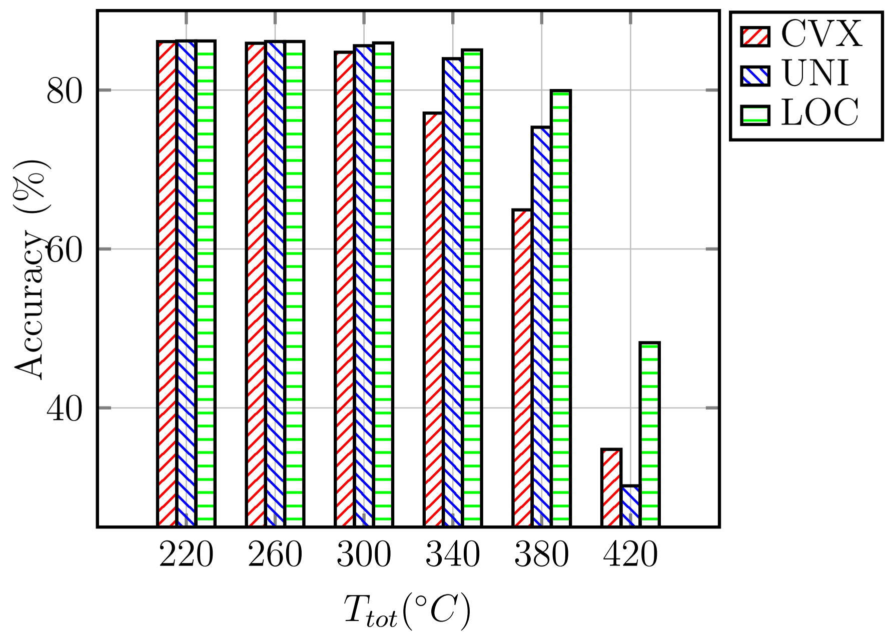


Figure 5.4: Accuracy of Conv6-Cifar10 for increasing values of the total temperature of the memory system.

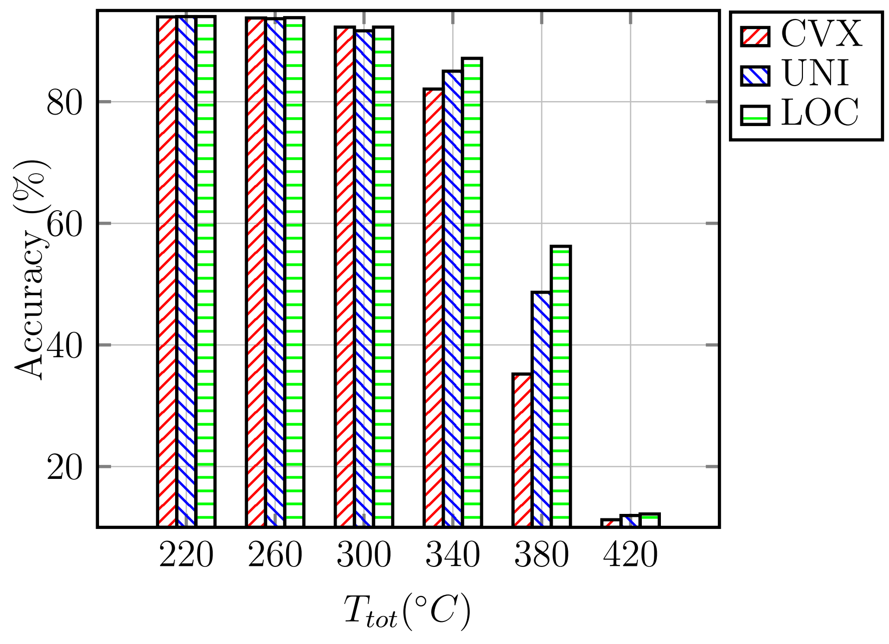


Figure 5.5: Accuracy of ResNet20-Cifar10 for increasing values of the total temperature of the memory system.

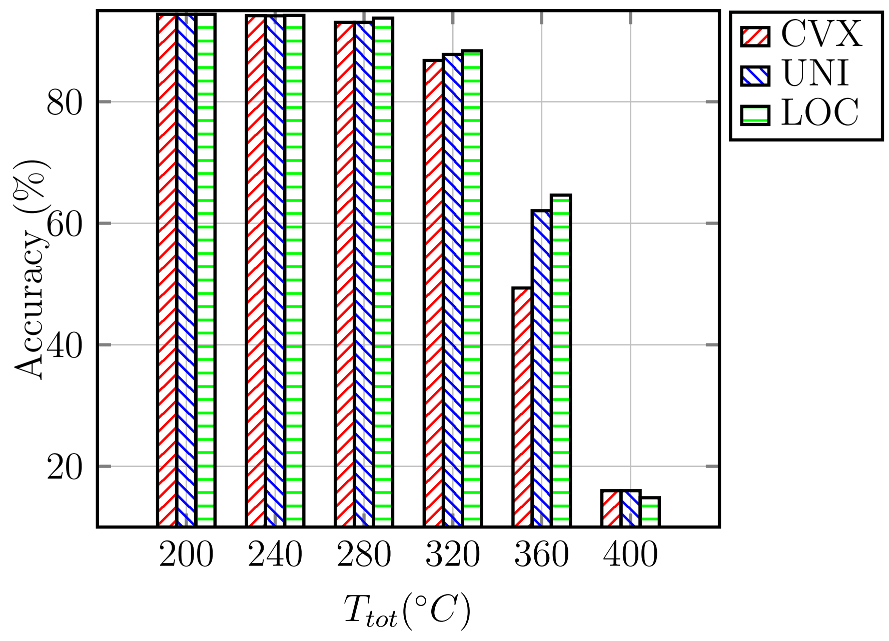


Figure 5.6: Accuracy of DenseNet40-Cifar10 for increasing values of the total temperature of the memory system.

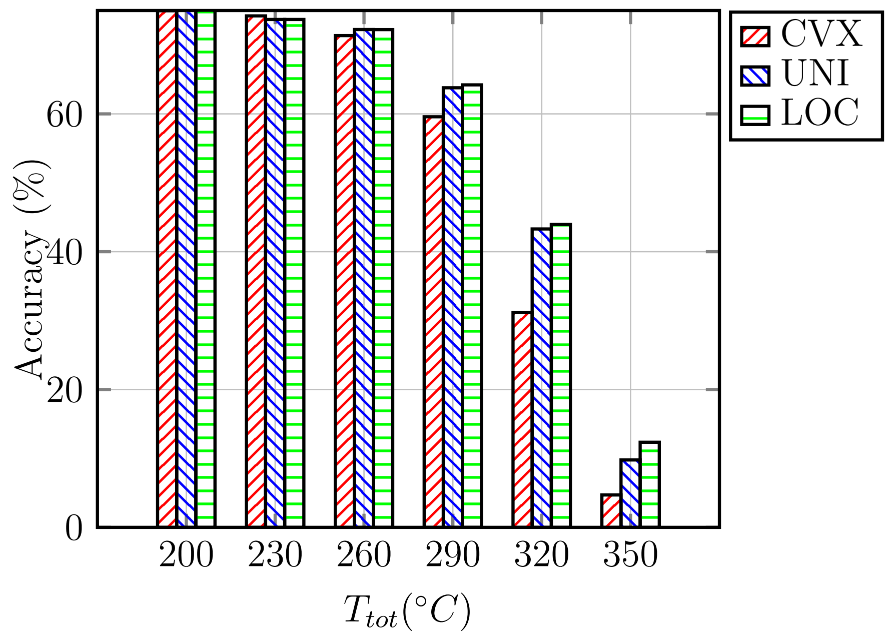


Figure 5.7: Accuracy of DenseNet40-Cifar100 for increasing values of the total temperature of the memory system.

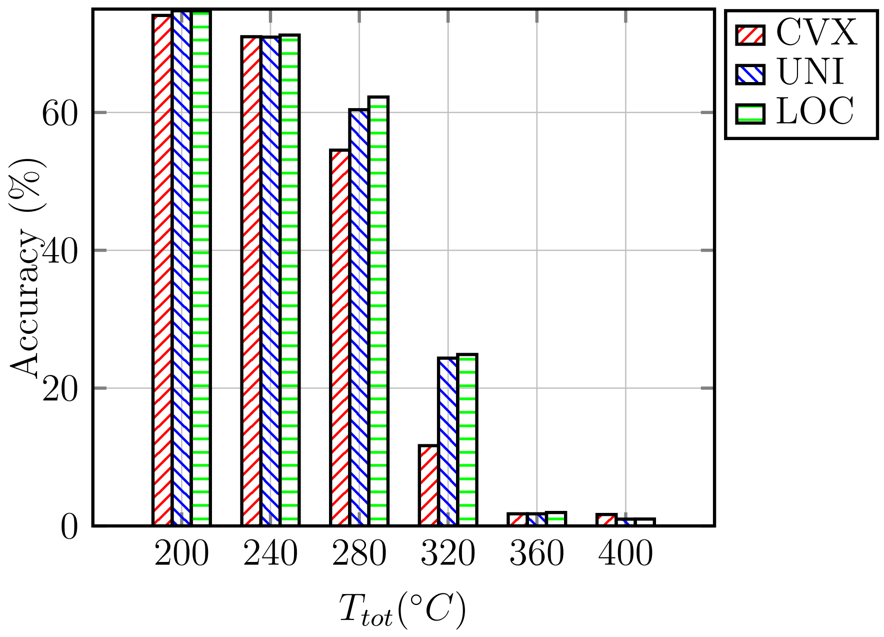


Figure 5.8: Accuracy of ResNet56-Cifar100 for increasing values of the total temperature of the memory system.

5.7 Conclusion

The emerging 3D ICs have a great potential to alleviate the memory bottleneck problem of growing DNN sizes. However, 3D ICs introduce large amplitudes of thermal gradients than their conventional counterparts. The growing thermal map of the processing units degrades their performance in terms of accuracy for executing AI applications, which should be closely characterized in the ECS design.

Therefore, in this chapter, first, we study the importance of thermal awareness by incorporating the sensitivity metric developed in Chapter 4 to map DNN neuron parameters on memory banks. Our experimental results show that the thermal-aware mapping can achieve 0.18-47.91% accuracy improvement than the thermal ignorant for state-of-the-art architectures, like LeNet, Conv6, ResNet20, ResNet56, and DenseNet40. Then, we develop a bin-packing-based convex optimization method to find the optimal temperature profile of the memory system to maximize the accuracy under constant energy utilization. We compare the developed method with uniform temperature profiles generated by workload distribution among processing cores and temperature profiles explored through local search in the neighborhood of uniform temperatures. From experiment results, we show that the developed method has comparable performance at the low temperatures but is inferior at high temperatures due to discrepancies in the sensitivity metric. It highlights the future work of this dissertation to design a more accurate sensitivity metric for quantifying thermal impact. Besides, we show that the uniform temperature profile of the memory system does not guarantee the high accuracy of AI applications for DNN execution.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

We begin this chapter by summarizing our contributions to the dissertation. Then we discuss probable research directions for the future.

6.1 Summary

The AV technology development has drawn significant attention from the automotive manufacturers due to its social and economic revolutionary capabilities. AVs have the ability to minimize road accidents caused by human errors and save precious lives. It can offer new mobility options for the elderly, the disabled, and children. AVs can reduce traffic congestion and parking problems in large cities, which can help cut down the emissions, thereby helping the environment. The automotive industry has invested significant resources in AV development to achieve mass production by considering these impacts. However, in the current status, there are numerous safety and reliability challenges for the successful deployment of AV technology.

Thermal issues in the ECS are one of the main barriers to the reliability of the AVs. Due to the miniaturization of the ICs, there is a soaring power density in the processing units. Additionally, the ECS of vehicles undergoes harsh thermal conditions due to the surrounding environment and heat produced by the neighboring mechanical components. Altogether, the operating temperature of the ECS elevates rapidly. First, such high temperatures can degrade the performance of the PUs either by throttling the clock speed or shutting down, which can impact the latency of mission-critical tasks causing catastrophic events. Second, high temperatures deteriorate the lifetime of the PUs. Third, high temperatures can perturb the data stored in the memory of the PUs, which can misclassify the objects in vision-related AI applications leading to hazardous consequences. Therefore, the thermal-aware design of the ECS can optimize its performance, reliability, and accuracy.

In this dissertation, step-by-step, we develop techniques to minimize the thermal challenges of the ECS. First, we address the challenge of satisfying the peak temperature, bounding the peak operating temperature, and maximizing the system-wide reliability. We develop a mathematical program to minimize the latency of the applications by satisfying the thermal constraints. Then, we present an algorithm to effectively bound the peak temperature of the periodic applications under variable execution time scenarios. Also, we present two analytical methods for compute-efficient MTTF formulation. Afterward, we use a sophisticated genetic algorithm-based approach to explore the design space using a developed temperature bounding algorithm and compute-efficient MTTF formulation to minimize the latency, reduce the peak temperature and enhance the MTTF by satisfying the thermal constraint. Our experiment results for two automotive benchmarks demonstrate that the proposed peak temperature bounding method is accurate by $17°C$ and $40°C$, which results in 81% and 260% more accurate MTTF estimation. Moreover, the compute-efficient MTTF formulation is faster by $17\times$ and $191\times$.

Second, we address the challenge of quantifying the thermal impact on the AI applications' accuracy by developing a more accurate sensitivity framework. We consider emerging non-volatile memory technology, e.g., STT-MRAM for storing DNNs, which can mitigate the memory bottleneck presented due to the ever-growing complexity and size of the DNNs to achieve high accuracies. However, data stored in the STT-MRAM flips with increasing temperature. Hence, we present a sensitivity estimation framework to accurately identify delicate neurons of the DNNs, which significantly affect the accuracy under thermal impact. Our experimental results for LeNet, ResNet20, ResNet56, and DenseNet40 by protecting sensitive neurons achieve 0.5-4.2% better accuracy than state-of-the-art methods. Later, we present the limitation of the developed sensitivity framework when errors are large.

Third, we study the challenge of mapping DNN neuron parameters on memory banks under the thermal impact for ECS design. The emerging 3D ICs have large thermal gradients than their traditional counterparts due to increased device density and chip geometry. We incorporate a developed sensitivity method in the bin-packing-based heuristic to map the neuron parameters on memory banks and evaluate the thermal awareness by comparing it with the temperature oblivious method. We achieve 0.18-47.91% better performance for thermal-aware mapping than the thermal oblivious approach at high temperatures for standard DNN architectures, like LeNet, Conv6, ResNet20, ResNet56, and DenseNet40. Afterward, we develop a heuristic based on bin-packing and convex optimization to find the optimal temperature profile of the memory banks to maximize the accuracy under constant energy utilization while mapping DNN neuron parameters. We compare the developed method with a general phenomenon of uniform temperature profile and local search-based methods. Overall, experiment results demonstrate that the developed method performs well at low temperatures and show inconsistencies at high temperature due to discrepancies in the sensitivity method for state-of-the-art DNNs. Besides, we show that uniform temperatures are not the optimal way of maximizing the accuracy of DNNs.

Altogether, in this dissertation, we address the performance, lifetime reliability, and accuracy impacts due to high-temperature dynamics of the ECS design to enhance the safety of AVs.

6.2 Future Work

The resource investment in the AV industry will undoubtedly keep growing in the coming years due to its socio-economic transforming potential. Growing attention to AV development will shift the computing paradigm to more complex and strict real-time systems. The emerging 3D IC technology has tremendous scope for future computing systems. The rapidly growing complexity in the underlying hardware [1, 44] and size of the software [19, 43] will definitely emit more heat, introducing a new set of thermal challenges for the AV design.

The resource management at the processing cores and memory system together should be considered to manage the thermal problems of the future complex computing systems. In our current research, we target the thermal problems of the processing cores and memory systems separately. In particular, first, we map the vehicle applications on the heterogeneous architecture of the PUs to satisfy the design constraints, such as latency, peak temperature, reliability. Then, we focus on thermal aware mapping of the DNN on the memory system to improve the accuracy. We believe that the integrated approach of the processing cores and memory system design will be inevitable to manage the thermal problems of the complex ECS.

The sensitivity analysis plays a crucial role in detecting delicate parameters of the neural network, which drastically degrades the accuracy of AI applications under unnoticeable perturbations [63]. The sensitivity analysis developed in Chapter 4 utilizes the first-order and second-order derivatives of the cost function, which improves the accuracy significantly for specific DNN architectures at a high-temperature range. Our experiment results from Chapter 5 highlight the limitation of the developed sensitivity method, which motivates a more accurate metric development to quantify the thermal impact. Therefore, it is expected that the integration of the third-order derivative term in the sensitivity framework would yield even better accuracy, which will need a compute-efficient mechanism to approximate the third-order component of the cost function.

It is mandatory for the automotive industry to apply the ISO-26262 standard into the vehicle design to ensure different automotive safety integrity levels (ASILs) [81, 207–209]. Our research framework can be used to integrate the system-level reliability into ASILs. The different hardware units in the ECS have different ASIL requirements. Accordingly, we can achieve ASILs for processing units of the ECS through application mapping/scheduling. We can implement safety mechanisms and satisfy ASILs through effective resource management of the PUs and memory systems.

BIBLIOGRAPHY

[1] A. Gholami, Z. Yao, S. Kim, M. W. Mahoney, and K. Keutzer, “AI and Memory Wall,” *RiseLab Medium Post*, 2021.

[2] M. Awatramani, “Workload-Aware Scheduling Techniques for General Purpose Applications on Graphics Processing Units,” Ph.D. dissertation, Iowa State University, 2017.

[3] World Health Organization (WHO), “Global Status Report on Road Safety 2018,” Tech. Rep., Dec. 2018. [Online]. Available: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/

[4] A. Haghi, D. Ketabi, M. Ghanbari, and Rajabi, “Assessment of human errors in driving accidents; Analysis of the causes based on aberrant behaviors,” *Life Science Journal*, vol. 11, no. 9, pp. 414–420, 2014.

[5] U.S. National Highway Traffic Safety Administration, “Automated Vehicles for Safety,” 2020. [Online]. Available: https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[6] S. Curto, A. Severino, S. Trubia, F. Arena, and L. Puleo, “The effects of autonomous vehicles on safety,” *AIP Conference Proceedings*, vol. 2343, no. 1, p. 110013, 2021.

[7] P. A. Hancock, I. Nourbakhsh, and J. Stewart, “On the future of transportation in an era of automated and autonomous vehicles,” *Proceedings of the National Academy of Sciences*, vol. 116, no. 16, pp. 7684–7691, 2019.

[8] Audi, “Audi is researching the use of time in the robot car,” Jul. 2017. [Online]. Available: https://www.audi-mediacenter.com/en/press-releases/audi-is-researching-the-use-of-time-in-the-robot-car-9120

[9] R. Shanker et al., “Autonomous Cars: Self-Driving the New Auto Industry Paradigm,” *Morgan Stanley Blue Paper*, p. 109, 2013.

[10] M. Massar, I. Reza, S. M. Rahman, S. M. H. Abdullah, A. Jamal, and F. S. Al-Ismail, “Impacts of Autonomous Vehicles on Greenhouse Gas Emissions—Positive or Negative?” *International Journal of Environmental Research and Public Health*, vol. 18, no. 11, 2021.

[11] A. Millard-Ball, "The autonomous vehicle parking problem," *Transport Policy*, vol. 75, pp. 99–108, 2019.

[12] H. Claypool, A. Bin-Nun, and J. Gerlach, "Self-Driving Cars: The Impact on People with Disabilities," *Ruderman Family Foundation*, Jan. 2017.

[13] C. Metz, "The Costly Pursuit of Self-Driving Cars Continues On. And On. And On." *The New York Times*, May 2021. [Online]. Available: https://www.nytimes.com/2021/05/24/technology/self-driving-cars-wait.html

[14] L. Day, "The Current State Of Play In Autonomous Cars," *Hackday*, Dec. 2021. [Online]. Available: https://hackaday.com/2021/12/29/the-current-state-of-play-in-autonomous-cars/

[15] S. Singh and B. S. Saini, "Autonomous cars: Recent developments, challenges, and possible solutions," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, p. 012028, Jan. 2021.

[16] J. Cusack, "How driverless cars will change our world," *BBC Future*, Nov. 2021. [Online]. Available: https://www.bbc.com/future/article/20211126-how-driverless-cars-will-change-our-world

[17] C. Johnson and J. Walker, "PEAK CAR OWNERSHIP: The market opportunity of electric automated mobility services," *Rocky Mountain Institute*, Mar. 2017.

[18] Statista DossierPlus, "Autonomous Vehicles ," no. did-69417-1, p. 40, 2019.

[19] V. Haydin, "The Emerging Future of Autonomous Vehicles," *Intellias: Intelligent Software Engineering*, 2021. [Online]. Available: https://intellias.com/the-emerging-future-of-autonomus-driving/

[20] Yole Development, "Sensors and data management for autonomous vehicles," Tech. Rep., 2015.

[21] M. Haenlein and A. Kaplan, "A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence," *California Management Review*, vol. 61, no. 4, pp. 5–14, 2019.

[22] R. Anyoha, "The History of Artificial Intelligence," *Blog, Special Edition on Artificial Intelligence: Harvard University*, 2017. [Online]. Available: https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

[23] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[24] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep Learning–Based Text Classification: A Comprehensive Review," *ACM Comput. Surv.*, vol. 54, no. 3, apr 2021.

[25] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Research*, vol. 43, no. 4, pp. 244–252, 2019.

[26] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, p. 303–314.

[27] Đorđe Petrović, R. Mijailović, and D. Pešić, "Traffic Accidents with Autonomous Vehicles: Type of Collisions, Manoeuvres and Errors of Conventional Vehicles' Drivers," *Transportation Research Procedia*, vol. 45, pp. 161–168, 2020.

[28] M. Harris, "Google reports self-driving car mistakes: 272 failures and 13 near misses," *The Guardian*, 2016.

[29] Clifford Law, "Driverless Car Accidents – Who's at Fault?" *National Law Review, Volume XI, Number 176*, vol. XI, no. 176, 2021.

[30] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Networks*, vol. 90, p. 101823, 2019.

[31] A. Chowdhury, G. Karmakar, J. Kamruzzaman, A. Jolfaei, and R. Das, "Attacks on Self-Driving Cars and Their Countermeasures: A Survey," *IEEE Access*, vol. 8, pp. 207 308–207 342, 2020.

[32] European Union Agency for Cybersecurity (ENISA) and the Joint Research Centre (JRC), "Cybersecurity Challenges in the Uptake of Artificial Intelligence in Autonomous Driving," Tech. Rep., Feb. 2021.

[33] P. A. Hancock, T. Kajaks, J. K. Caird, M. H. Chignell, S. Mizobuchi, P. C. Burns, J. Feng, G. R. Fernie, M. Lavallière, I. Y. Noy, D. A. Redelmeier, and B. H. Vrkljan, "Challenges to Human Drivers in Increasingly Automated Vehicles," *Human Factors*, vol. 62, no. 2, pp. 310–328, 2020.

[34] G. Brannon, "Vehicle Technology Survey- Phase II," *American Automobile Association, Inc*, 2017.

[35] A. Danise, "Women Say No Thanks to Driverless Cars, Survey Finds; Men Say Tell Me More," *Nerdwallet*, 2015.

[36] A. Alhashimi, D. Varagnolo, and T. Gustafsson, "Joint Temperature-Lasing Mode Compensation for Time-of-Flight LiDAR Sensors," *Sensors*, vol. 15, no. 12, pp. 31 205–31 223, 2015.

[37] Laird Thermal Systems Application Note, "Cooling Solutions for Autonomous Systems," Tech. Rep.

[38] M. Krüger, S. Straube, A. Middendorf, D. Hahn, T. Dobs, and K. D. Lang, "Requirements for the application of ECUs in e-mobility originally qualified for gasoline cars," *Microelectronics Reliability*, vol. 64, pp. 140–144, 2016.

[39] A. S. Bankar, S. Sha, V. Chaturvedi, and G. Quan, "Thermal Aware Lifetime Reliability Optimization for Automotive Distributed Computing Applications," *2020 IEEE 38th International Conference on Computer Design (ICCD)*, pp. 498–505, 2020.

[40] ZVEI-German Electrical and Electronic Manufacturers' Association, "Handbook for Robustness Validation of Automotive Electrical / Electronic Modules," Frankfurt, Germany, Tech. Rep., 2013.

[41] S. Heinrich and L. Motors, "Flash Memory in the emerging age of autonomy," in *Flash Memory Summit*, 2017.

[42] J.-H. Han, K. Torres-Castro, R. E. West, N. Swami, and M. Stan, "Thermal Analysis of Microfluidic cooling in Processing-in-3D-Stacked Memory," in *2021 22nd International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE)*, 2021, pp. 1–6.

[43] U.S. Government Accountability Office, “Vehicle Cybersecurity: DOT and Industry Have Efforts Under Way, but DOT Needs to Define Its Role in Responding to a Real-world Attack,” *GAO-16-350*, p. 61, 2016.

[44] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, “Compute and Energy Consumption Trends in Deep Learning Inference,” *arXiv:2109.05472 [cs.LG]*, Sep. 2021.

[45] S. Pagani, P. D. Manoj, A. Jantsch, and J. Henkel, “Machine Learning for Power, Energy, and Thermal Management on Multicore Processors: A Survey,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 101–116, 2020.

[46] A. Rahmati, M. Hicks, D. Holcomb, , and K. Fu, “Refreshing Thoughts on DRAM: Power Saving vs. Data Integrity,” *Workshop on Approximate Computing Across the System Stack (WACAS)*, 2014.

[47] S. Sethuraman, V. K. Tavva, K. Rajamani, C. K. Subramanian, K. H. Kim, H. C. Hunter, and M. B. Srinivas, “Temperature Aware Adaptations for Improved Read Reliability in STT-MRAM Memory Subsystem,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4635–4644, 2020.

[48] M. Jung, Y. Jin, and M. Shihab, “Area, Power and Latency Considerations of STT-MRAM to Substitute for Main Memory,” in *41st International Symposium on Computer Architecture (ISCA-41)*, 06 2014.

[49] M. V. Beigi and G. Memik, “Thermal-Aware Optimizations of ReRAM-Based Neuromorphic Computing Systems,” in *Proceedings of the 55th Annual Design Automation Conference*, 2018.

[50] R. Viswanath, V. Wakharkar, A. Watawe, and V. Lebonheur, “Thermal Performance Challenges from Silicon to Systems,” *Intel Technology Journal Q3*, pp. 1–16, 2000.

[51] M. Valad Beigi and G. Memik, “THOR: THermal-aware Optimizations for extending ReRAM Lifetime,” in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2018, pp. 670–679.

[52] W. Lu, P.-T. Huang, H.-M. Chen, and W. Hwang, “An Energy-Efficient 3D Cross-Ring Accelerator With 3D-SRAM Cubes for Hybrid Deep Neural Net-

works," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 4, pp. 776–788, 2021.

[53] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: Multi-Purpose Log-Quantized DNN Inference Engine Stacked on 96-MB 3-D SRAM Using Inductive Coupling Technology in 40-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 186–196, 2019.

[54] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, p. 380–392, jun 2016.

[55] H. Martin, G. Przemyslaw, W. Bernhard, and R. Sven, "Affordable and Safe High Performance Vehicle Computers with Ultra-Fast On-Board Ethernet for Automated Driving," *Advanced Microsystems for Automotive Applications*, pp. 56–68, 2019.

[56] J. Korta, P. Skruch, and K. Holon, "Reliability of Automotive Multidomain Controllers: Advancements in Electronics Cooling Technologies," *IEEE Vehicular Technology Magazine*, vol. 16, pp. 86–94, 2021.

[57] S. G. Kandlikar and A. Ganguly, "Fundamentals of Heat Dissipation in 3D IC Packaging and Thermal-Aware Design," in *3D Microelectronic Packaging: From Architectures to Applications.* Springer Singapore, 2021, pp. 369–395.

[58] C. Mandalapu, I. Abdel-Motaleb, S. Hong, and R. Patti, "Design, Fabrication, and Testing of a Liquid Cooling Platform for High Power 3D-ICs," in *2019 8th International Symposium on Next Generation Electronics (ISNE)*, 2019, pp. 1–4.

[59] S. Islam and I. Abdel-Motaleb, "Thermal Stress Analysis of Liquid-Cooled 3D-ICs," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, 2020, pp. 132–135.

[60] A. Mutapcic, S. Boyd, S. Murali, D. Atienza, G. De Micheli, and R. Gupta, "Processor Speed Control with Thermal Constraints," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 9, pp. 1994–2008, 2009.

[61] S. Sha, A. S. Bankar, X. Yang, W. Wen, and G. Quan, "On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, no. 2, Mar. 2020.

[62] H. Lin, Y. Feng, and X. Qiang, "Lifetime reliability-aware task allocation and scheduling for MPSoC platforms," *Proceedings -Design, Automation and Test in Europe, DATE*, pp. 51–56, 2009.

[63] S. Dash and S. Mukhopadhyay, "Hessian-Driven Unequal Protection of DNN Parameters for Robust Inference," *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2020.

[64] G. Xie, H. Peng, Z. Li, and J. Song, "Reliability Enhancement Toward Functional Safety Goal Assurance in Energy-Aware Automotive Cyber-Physical Systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, pp. 5447–5462, 2018.

[65] G. Xie, G. Zeng, Y. Liu, J. Zhou, R. Li, and K. Li, "Fast Functional Safety Verification for Distributed Automotive Applications during Early Design Phase," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 5, pp. 4378–4391, 2018.

[66] J. Kocić, N. Jovičić, and V. Drndarević, "An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms," *Sensors (Basel)*, vol. 19, no. 9, 2019.

[67] W. Liao, L. He, and K. Lepak, "Temperature and supply Voltage aware performance and power modeling at microarchitecture level," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1042–1053, 2005.

[68] G. Quan and V. Chaturvedi, "Feasibility Analysis for Temperature-constraint Hard Real-Time Periodic Tasks," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 3, pp. 329–339, 2010.

[69] V. Hanumaiah, S. Vrudhula, and K. S. Chatha, "Performance Optimal Online DVFS and Task Migration Techniques for Thermally Constrained Multi-Core Processors," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1677–1690, 2011.

[70] M. Bao, A. Andrei, P. Eles, and Z. Peng, "On-line thermal aware dynamic voltage scaling for energy optimization with frequency/temperature depen-

dency consideration," in *2009 46th ACM/IEEE Design Automation Conference*, 2009, pp. 490–495.

[71] Y. Lee, H. S. Chwa, K. G. Shin, and S. Wang, "Thermal-Aware Resource Management for Embedded Real-Time Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2857–2868, 2018.

[72] Q. Han, M. Fan, O. Bai, S. Ren, and G. Quan, "Temperature-Constrained Feasibility Analysis for Multicore Scheduling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 12, pp. 2082–2092, 2016.

[73] S. Pagani, J. J. Chen, M. Shafique, and J. Henkel, "MatEx: Efficient Transient and Peak Temperature Computation for Compact Thermal Models," *Proceedings -Design, Automation and Test in Europe, DATE*, vol. 2015-April, pp. 1515–1520, 2015.

[74] M. Fan, V. Chaturvedi, S. Sha, and G. Quan, "An analytical solution for multi-core energy calculation with consideration of leakage and temperature dependency," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2013, pp. 353–358.

[75] W. Guan, M. G. Moghaddam, and C. Ababei, "Quantifying the impact of uncertainty in embedded systems mapping for NoC based architectures," *Microprocessors and Microsystems*, vol. 80, p. 103503, 2021.

[76] J. R. Black, "Electromigration—A Brief Survey and Some Recent Results," *IEEE Transactions on Electron Devices*, vol. 16, no. 4, pp. 338–347, 1969.

[77] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[78] C. Banerjee, T. Mukherjee, and E. Pasiliao, "An Empirical Study on Generalizations of the ReLU Activation Function," in *Proceedings of the 2019 ACM Southeast Conference*, 2019, p. 164–167.

[79] Research and Markets, "Global Autonomous Vehicle ECU Market 2020-2024," 2020.

[80] J. Scheuring and U. Bernhard, "Time-Sensitive Networking in the Automotive Industry," vol. 15, no. 9, pp. 54–58, 2020.

[81] I. . Standard, “Road Vehicles-Functional Safety,” Tech. Rep., 2018.

[82] A. James, “Reliability, Availability and Maintainability Aspects of Automobiles,” *Life Cycle Reliab Safety Engineering*, vol. 10, pp. 81–89, 2021.

[83] STOUT, “Automotive Defect and Recall Report,” Tech. Rep., 2019.

[84] S. Hamdioui, D. Gizopoulos, G. Guido, M. Nicolaidis, A. Grasset, and P. Bonnot, “Reliability Challenges of Real-Time Systems in Forthcoming Technology Nodes,” *Proceedings -Design, Automation and Test in Europe, DATE*, pp. 129–134, 2013.

[85] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. . Rosing, “WARM: Workload-Aware Reliability Management in Linux/Android,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 9, pp. 1557–1570, 2017.

[86] Y. Ma, T. Chantem, R. P. Dick, and X. S. Hu, “Improving System-Level Lifetime Reliability of Multicore Soft Real-Time Systems,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1895–1905, 2017.

[87] L. Niu, “Reliability-Aware Energy-Efficient Scheduling for (m,k)-Constrained Real-Time Systems Through Shared Time Slots,” *Microprocessors and Microsystems*, vol. 77, p. 103110, 2020.

[88] A. Taherin, M. Salehi, and A. Ejlali, “Reliability-Aware Energy Management in Mixed-Criticality Systems,” *IEEE Transactions on Sustainable Computing*, vol. 3, no. 3, pp. 195–208, 2018.

[89] M. A. Haque, H. Aydin, and D. Zhu, “On Reliability Management of Energy-Aware Real-Time Systems Through Task Replication,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 813–825, 2017.

[90] D. Zhu, “Reliability-Aware Dynamic Energy Management in Dependable Embedded Real-Time Systems,” *Transactions on Embedded Computing Systems*, vol. 10, no. 2, pp. 1–27, 2010.

[91] J. Huang, R. Li, X. Jiao, Y. Jiang, and W. Chang, “Dynamic DAG Scheduling on Multiprocessor Systems: Reliability, Energy, and Makespan,” *IEEE*

Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 39, no. 11, pp. 3336–3347, 2020.

[92] I. Ukhov, P. Eles, and Z. Peng, "Temperature-Centric Reliability Analysis and Optimization of Electronic Systems under Process Variation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 11, pp. 2417–2430, 2015.

[93] L. Schor, I. Bacivarov, H. Yang, and L. Thiele, "Worst-Case Temperature Guarantees For Real-Time Applications on Multi-core Systems," *Real-Time Technology and Applications - Proceedings*, pp. 87–96, 2012.

[94] A. L. Da Silva, A. L. Del Mestre Martins, and F. Gehm Moraes, "Fine-grain Temperature Monitoring for Many-core Systems," *Proceedings - 32nd Symposium on Integrated Circuits and Systems Design, SBCCI 2019*, 2019.

[95] V. Chaturvedi, H. Huang, S. Ren, and G. Quan, "On the Fundamentals of Leakage Aware Real-Time DVS Scheduling for Peak Temperature Minimization," *Journal of Systems Architecture*, vol. 58, no. 10, pp. 387–397, 2012.

[96] K. C. Chen, H. W. Tang, Y. H. Liao, and Y. C. Yang, "Temperature Tracking and Management With Number-limited Thermal Sensors for Thermal-Aware NoC Systems," *IEEE Sensors Journal*, vol. 20, no. 21, pp. 13 018–13 028, 2020.

[97] M. Sojka, O. Benedikt, Z. Hanzalek, and P. Zaykov, "Testbed for Thermal and Performance Analysis in MPSoC Systems," *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems, FedCSIS 2020*, vol. 21, pp. 683–692, 2020.

[98] J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma, "Reliability and Temperature Constrained Task Scheduling for Makespan Minimization on Heterogeneous Multi-core Platforms," *Journal of Systems and Software*, vol. 133, pp. 1–16, 2017.

[99] K. Ergun, X. Yu, N. Nagesh, L. Cherkasova, P. Mercati, R. Ayoub, and T. Rosing, *RelIoT: Reliability Simulator for IoT Networks*, W. Song, K. Lee, Z. Yan, L.-J. Zhang, and H. Chen, Eds., Cham, 2020.

[100] M. I. bin Bandan, S. Bhattacharjee, S. K. Jali, and D. K. Pradhan, "Instantaneous Mean-Time-To-Failure (MTTF) Estimation for Checkpoint Interval Computation at Run Time," *Microelectronics Reliability*, vol. 98, pp. 69–77, 2019.

[101] K. N. Tu and A. M. Gusak, "A Unified Model of Mean-Time-To-Failure for Electromigration, Thermomigration, and Stress-migration Based on Entropy Production," *Journal of Applied Physics*, vol. 126, no. 7, 2019.

[102] S. S. Sahoo, B. Veeravalli, and A. Kumar, "CL(R) Early : An Early-stage DSE Methodology for Cross-Layer Reliability-Aware Heterogeneous Embedded Systems," *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020.

[103] H. Salamy, "Energy-Aware Schedules under Chip Reliability Constraint for Multi-Processor Systems-on-a-Chip," *Journal of Circuits, Systems and Computers*, vol. 29, no. 9, 2020.

[104] J. Karlsson and P. Khosravi, "Evaluation of Security Mechanisms for an Integrated Automotive System Architecture," Master's thesis, Chalmers University of Technology, 2017.

[105] D. Reinhardt, U. Dannebaum, M. Scheffer, and M. Traub, "High Performance Processor Architecture for Automotive Large Scaled Integrated Systems within the European Processor Initiative Research Project," *WCX SAE World Congress Experience*, 2019.

[106] H. F. Sheikh, I. Ahmad, Z. Wang, and S. Ranka, "An Overview and Classification of Thermal-Aware Scheduling Techniques for Multi-core Processing Systems," *Sustainable Computing: Informatics and Systems*, vol. 2, no. 3, pp. 151–169, 2012.

[107] A. Ait El Cadi, O. Souissi, R. Ben Atitallah, N. Belanger, and A. Artiba, "Mathematical Programming Models for Scheduling in a CPU/FPGA Architecture with Heterogeneous Communication Delays," *Journal of Intelligent Manufacturing*, vol. 29, no. 3, pp. 629–640, 2018.

[108] F. A. Omara and M. M. Arafa, "Genetic Algorithms for Task Scheduling Problem," *Journal of Parallel and Distributed Computing*, vol. 70, no. 1, pp. 13–22, 2010.

[109] A. Iranfar, M. Kamal, A. Afzali-Kusha, M. Pedram, and D. Atienza, "TheSPoT: Thermal Stress-Aware Power and Temperature Management for Multiprocessor Systems-on-Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1532–1545, 2018.

[110] K. Skadron, K. Sankaranarayanan, S. Velusamy, D. Tarjan, M. R. Stan, and W. Huang, "Temperature-Aware Microarchitecture: Modeling and Implementation," *ACM Transactions on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.

[111] S. Sharifi, D. Krishnaswamy, and T. S. Rosing, "PROMETHEUS: A Proactive Method For Thermal Management of Heterogeneous MPSoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 1110–1123, 2013.

[112] R. Rao and S. Vrudhula, "Fast and Accurate Prediction of the Steady-State Throughput of Multicore Processors Under Thermal Constraints," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 1559–1572, 2009.

[113] M. Faizan and A. S. Pillai, "Dynamic Task Allocation and Scheduling for Multicore Electronics Control Unit (ECU)," *Proceedings of the 3rd International Conference on Electronics and Communication and Aerospace Technology, ICECA 2019*, pp. 821–826, 2019.

[114] B. Andersson, H. Kim, D. D. Niz, M. Klein, R. R. Rajkumar, and J. Lehoczky, "Schedulability Analysis of Tasks with Corunner-Dependent Execution Times," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 3, May 2018.

[115] L. Schor, H. Yang, I. Bacivarov, and L. Thiele, "Worst-Case Temperature Analysis for Different Resource Models," *IET Circuits, Devices and Systems*, vol. 6, no. 5, pp. 297–307, 2012.

[116] S. K. Roy, R. Devaraj, A. Sarkar, S. Sinha, and K. Maji, "Optimal Scheduling of Precedence-constrained Task Graphs on Heterogeneous Distributed Systems With Shared Buses," *Proceedings - 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing, ISORC 2019*, pp. 185–192, 2019.

[117] F. Qi and C. P. Chen, "A Complete Monotonicity Property of the Gamma Function," *Journal of Mathematical Analysis and Applications*, vol. 296, no. 2, pp. 603–607, 2004.

[118] Y. Lee, E. Kim, and K. G. Shin, "Efficient Thermoelectric Cooling for Mobile Devices," *Proceedings of the International Symposium on Low Power Electronics and Design*, 2017.

[119] S. Speaks, "Reliability and MTBF Overview," Vicor Reliability Engineering, Tech. Rep., 2014.

[120] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A Survey of Deep Learning Applications to Autonomous Vehicle Control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2021.

[121] B. Zheng, H. Liang, Q. Zhu, H. Yu, and C.-W. Lin, "Next Generation Automotive Architecture Modeling and Exploration for Autonomous Driving," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2016, pp. 53–58.

[122] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 267–278.

[123] Y. Ibrahim, H. Wang, J. Liu, J. Wei, L. Chen, P. Rech, K. Adam, and G. Guo, "Soft errors in DNN accelerators: A comprehensive review," vol. 115, p. 113969, 2020.

[124] M. A. Hanif, F. Khalid, R. V. W. Putra, S. Rehman, and M. Shafique, "Robust Machine Learning Systems: Reliability and Security for Deep Neural Networks," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, 2018, pp. 257–260.

[125] F. Yao, A. S. Rakin, and D. Fan, "DeepHammer: Depleting the intelligence of deep neural networks through targeted chain of bit flips," in *29th USENIX Security Symposium (USENIX Security 20)*, Aug. 2020, pp. 1463–1480.

[126] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017.

[127] X. Liu, M. Zhou, T. S. Rosing, and J. Zhao, "HR3AM: A Heat Resilient Design for RRAM-based Neuromorphic Computing," in *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1–6.

[128] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556 [cs.CV]*, Apr. 2015.

[129] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten, “Exploring the Limits of Weakly Supervised Pretraining,” *arXiv:1805.00932 [cs.CV]*, May 2018.

[130] Y. Wang, S. Liang, S. Yao, Y. Shan, S. Han, J. Peng, and H. Luo, “RECONFIGURABLE PROCESSOR FOR DEEP LEARNING IN AUTONOMOUS VEHICLES,” *ITU Journal: ICT Discoveries, Special Issue*, vol. 1, no. 1, p. 13, 2017.

[131] J. Zhang, M. Jung, and M. Kandemir, “FUSE: Fusing STT-MRAM into GPUs to Alleviate Off-Chip Memory Access Overheads,” in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2019, pp. 426–439.

[132] A. Anwar, A. Raychowdhury, R. Hatcher, and T. Rakshit, “XBAROPT - Enabling Ultra-Pipelined, Novel STT MRAM Based Processing-in-Memory DNN Accelerator,” in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2020, pp. 36–40.

[133] K. Mishty and M. Sadi, “Designing Efficient and High-performance AI Accelerators with Customized STT-MRAM,” *arXiv:2104.02199 [cs.AR]*, Apr. 2021.

[134] L. Wu, M. Taouil, S. Rao, E. J. Marinissen, and S. Hamdioui, “Survey on STT-MRAM Testing: Failure Mechanisms, Fault Models, and Tests,” *arXiv:2001.05463 [cs.ET]*, Jan. 2020.

[135] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning Filters for Efficient ConvNets,” *arXiv:1608.08710 [cs.CV]*, Mar. 2017.

[136] A. Khalid, A. Sundararajan, I. Acharya, and A. I. Sarwat, “Prediction of Li-Ion Battery State of Charge Using Multilayer Perceptron and Long Short-Term Memory Models,” in *2019 IEEE Transportation Electrification Conference and Expo (ITEC)*, 2019, pp. 1–6.

[137] W. Choi, D. Shin, J. Park, and S. Ghosh, “Sensitivity Based Error Resilient Techniques for Energy Efficient Deep Neural Network Accelerators,” in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019.

[138] J. Pizarroso, J. Portela, and A. Muñoz, "NeuralSens: Sensitivity Analysis of Neural Networks," *arXiv:2002.11423 [cs.LG]*, Feb. 2021.

[139] N. Golmant, Z. Yao, A. Gholami, M. Mahoney, and J. Gonzalez, "pytorch-hessian-eigenthings: efficient PyTorch Hessian eigendecomposition," Oct. 2018.

[140] Z. Yao, A. Gholami, K. Keutzer, and M. W. Mahoney, "PyHessian: Neural Networks Through the Lens of the Hessian," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 581–590.

[141] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[142] L. Xiang, X. Zeng, Y. Niu, and Y. Liu, "Study of Sensitivity to Weight Perturbation for Convolution Neural Network," *IEEE Access*, vol. 7, pp. 93 898–93 908, 2019.

[143] L. Li, X. Qi, T. Xie, and B. Li, "SoK: Certified Robustness for Deep Neural Networks," *arXiv:2009.04131 [cs.LG]*, Dec. 2021.

[144] P. A. Kowalski and M. Kusy, "Sensitivity Analysis for Probabilistic Neural Network Structure Reduction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1919–1932, 2018.

[145] I. M. Sobol, "Sensitivity estimates for nonlinear mathematical models," vol. 1, no. 4, 1990, p. 407—414.

[146] Y. Long, E. Lee, D. Kim, and S. Mukhopadhyay, "Q-PIM: A Genetic Algorithm based Flexible DNN Quantization Method and Application to Processing-In-Memory Platform," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[147] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both Weights and Connections for Efficient Neural Networks," *arXiv:1506.02626 [cs.NE]*, Oct. 2015.

[148] X. Sun, X. Ren, S. Ma, and H. Wang, "meProp: Sparsified Back Propagation for Accelerated Deep Learning with Reduced Overfitting," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 06–11 Aug 2017, pp. 3299–3308.

[149] G. Xu, H. Li, H. Ren, J. Sun, S. Xu, J. Ning, H. Yang, K. Yang, and R. H. Deng, "Secure and Verifiable Inference in Deep Neural Networks," in *Annual Computer Security Applications Conference*, 2020, p. 784–797.

[150] S. Yu, Z. Yao, A. Gholami, Z. Dong, S. Kim, M. W. Mahoney, and K. Keutzer, "Hessian-Aware Pruning and Optimal Neural Implant," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, January 2022, pp. 3880–3891.

[151] C. Jiang, G. Li, C. Qian, and K. Tang, "Efficient DNN Neuron Pruning by Minimizing Layer-wise Nonlinear Reconstruction Error," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, 7 2018, pp. 2298–2304.

[152] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[153] A. S. Hassan, T. Arifeen, and J.-A. Lee, "Data Footprint Reduction in DNN Inference by Sensitivity-Controlled Approximations with Online Arithmetic," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, 2020, pp. 534–541.

[154] T. Wu, X. Li, D. Zhou, N. Li, and J. Shi, "Differential Evolution Based Layer-Wise Weight Pruning for Compressing Deep Neural Networks," *Sensors*, vol. 21, no. 3, 2021.

[155] P. Panda, "QUANOS: Adversarial Noise Sensitivity Driven Hybrid Quantization of Neural Networks," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 2020, p. 187–192.

[156] Z. He, T. Zhang, and R. Lee, "Sensitive-Sample Fingerprinting of Deep Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[157] A. Jog, A. K. Mishra, C. Xu, Y. Xie, V. Narayanan, R. Iyer, and C. R. Das, "Cache revive: Architecting volatile STT-RAM caches for enhanced performance in CMPs," in *DAC Design Automation Conference 2012*, 2012, pp. 243–252.

[158] J. J. Kan, C. Park, C. Ching, J. Ahn, L. Xue, R. Wang, A. Kontos, S. Liang, M. Bangar, H. Chen, S. Hassan, S. Kim, M. Pakala, and S. H. Kang, "System-

atic validation of 2x nm diameter perpendicular MTJ arrays and MgO barrier for sub-10 nm embedded STT-MRAM with practically unlimited endurance," in *2016 IEEE International Electron Devices Meeting (IEDM)*, 2016, pp. 27.4.1–27.4.4.

[159] S. Sethuraman, V. K. Tavva, and M. B. Srinivas, "Techniques to Improve Write and Retention Reliability of STT-MRAM Memory Subsystem," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2021.

[160] L.-B. Faber, W. Zhao, J.-O. Klein, T. Devolder, and C. Chappert, "Dynamic compact model of Spin-Transfer Torque based Magnetic Tunnel Junction (MTJ)," in *2009 4th International Conference on Design Technology of Integrated Systems in Nanoscal Era*, 2009, pp. 130–135.

[161] M. A. Çavuşlu, C. Karakuzu, S. Şahin, and M. Yakut, "Neural network training based on FPGA with floating point number format and it's performance," *Neural Computing and Applications*, vol. 20, pp. 195–202, 2011.

[162] N. Lahbabi, S. S. K. C. Bulusu, J.-F. Hélard, and M. Crussiére, "Very Efficient Tone Reservation PAPR Reduction Fully Compatible With ATSC 3.0 Standard: Performance and Practical Implementation Analysis," *IEEE Access*, vol. 6, pp. 58 355–58 372, 2018.

[163] L. Lai, N. Suda, and V. Chandra, "Deep Convolutional Neural Network Inference with Floating-point Weights and Fixed-point Activations," *arXiv:1703.03073 [cs.LG]*, Mar. 2017.

[164] S. Kim and H. Kim, "Zero-Centered Fixed-Point Quantization With Iterative Retraining for Deep Convolutional Neural Network-Based Object Detectors," *IEEE Access*, vol. 9, pp. 20 828–20 839, 2021.

[165] H. Heaton, S. W. Fung, A. Gibali, and W. Yin, "Feasibility-based Fixed Point Networks," *arXiv:2104.14090 [cs.LG]*, Apr. 2021.

[166] S. Jain, S. Venkataramani, V. Srinivasan, J. Choi, P. Chuang, and L. Chang, "Compensated-DNN: Energy Efficient Low-Precision Deep Neural Networks by Compensating Quantization Errors," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018, pp. 1–6.

[167] D. Stutz, N. Chandramoorthy, M. Hein, and B. Schiele, "Bit Error Robustness for Energy-Efficient DNN Accelerators," *arXiv:2006.13977 [cs.LG]*, Apr. 2021.

[168] X. Dong, S. Chen, and S. J. Pan, “Learning to Prune Deep Neural Networks via Layer-wise Optimal Brain Surgeon,” *arXiv:1705.07565 [cs.NE]*, Nov. 2017.

[169] L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou, “Empirical Analysis of the Hessian of Over-Parametrized Neural Networks,” *arXiv:1706.04454 [cs]*, May 2018.

[170] A. Krogh and J. Hertz, “A Simple Weight Decay Can Improve Generalization,” in *Advances in Neural Information Processing Systems*, vol. 4, 1991.

[171] J. Kukačka, V. Golkov, and D. Cremers, “Regularization for Deep Learning: A Taxonomy,” *arXiv:1710.10686 [cs.CV]*, Oct. 2017.

[172] O. Demir-Kavuk, M. Kamada, and T. Akutsu, “Prediction using step-wise L1, L2 regularization and feature selection for small data sets with large number of features,” *BMC Bioinformatics*, vol. 12, Oct. 2011.

[173] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[174] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[175] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[176] 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:1704.04861v1 [cs.CV]*, Apr. 2017.

[177] Y. LeCun, C. Cortes, and C. Burges, “MNIST handwritten digit database,” *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, vol. 2, 2010.

[178] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” Tech. Rep., April 2009.

[179] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[180] Y. Wu, X. Zhu, C. Wu, A. Wang, and R. Ge, “Dissecting Hessian: Understanding Common Structure of Hessian in Neural Networks,” *arXiv:2010.04261v5 [cs.LG]*, Jun. 2021.

[181] J. Jeddeloh and B. Keeth, “Hybrid memory cube new DRAM architecture increases density and performance,” in *2012 Symposium on VLSI Technology (VLSIT)*, 2012, pp. 87–88.

[182] W.-H. Lo, K.-z. Liang, and T. Hwang, “Thermal-aware dynamic page allocation policy by future access patterns for Hybrid Memory Cube (HMC),” in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 1084–1089.

[183] H. Huang, M. Fan, and G. Quan, “Thermal aware overall energy minimization scheduling for hard real-time systems,” *Sustainable Computing: Informatics and Systems*, vol. 3, no. 4, pp. 274–285, 2013.

[184] S. Sha, W. Wen, S. Ren, and G. Quan, “M-Oscillating: Performance Maximization on Temperature-Constrained Multi-Core Processors,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 11, pp. 2528–2539, 2018.

[185] W. Wen, J. Yang, and Y. Zhang, “Optimizing power efficiency for 3D stacked GPU-in-memory architecture,” *Microprocessors and Microsystems*, vol. 49, pp. 44–53, 2017.

[186] L. Zhu, L. Bamberg, A. Agnesina, F. Catthoor, D. Milojevic, M. Komalan, J. Ryckaert, A. Garcia-Ortiz, and S. K. Lim, “Heterogeneous 3D Integration for a RISC-V System With STT-MRAM,” *IEEE Computer Architecture Letters*, vol. 19, no. 1, pp. 51–54, 2020.

[187] F. Wang, Y. Li, N. Yu, and Y. Yang, “Effectiveness of thermal redistribution layer in cooling of 3D ICs,” *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, vol. 34, no. 3, p. e2847, 2021.

[188] M. Fan, R. Rong, S. Liu, and G. Quan, “Energy calculation for periodic multicore scheduling in system thermal steady state with consideration of leakage

and temperature dependency," *The Journal of Supercomputing*, vol. 71, p. 2565–2584, 2015.

[189] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 19–24.

[190] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[191] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: Variation-Aware Training for Memristor X-Bar," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015.

[192] A. Goel, C. Tung, Y.-H. Lu, and G. K. Thiruvathukal, "A Survey of Methods for Low-Power Deep Learning and Computer Vision," in *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1–6.

[193] S. Grigorescu, T. Cocias, B. Trasnea, A. Margheri, F. Lombardi, and L. Aniello, "Cloud2Edge Elastic AI Framework for Prototyping and Deployment of AI Inference Engines in Autonomous Vehicles," *Sensors*, vol. 20, no. 19, 2020.

[194] G. A. Chaparro-Baquero, S. Sha, S. Homsi, W. Wen, and G. Quan, "Thermal-aware joint CPU and memory scheduling for hard real-time tasks on multicore 3D platforms," in *2017 Eighth International Green and Sustainable Computing Conference (IGSC)*, 2017, pp. 1–8.

[195] Y.-J. Chen, C.-L. Yang, P.-S. Lin, and Y.-C. Lu, "Opportunities of Synergistically Adjusting Voltage-Frequency Levels of Cores and DRAMs in CMPs with 3d-Stacked DRAMs for Efficient Thermal Control," *SIGAPP Appl. Comput. Rev.*, vol. 16, no. 1, p. 26–35, apr 2016.

[196] B. Goplen and S. Sapatnekar, "Thermal via Placement in 3D ICs," in *Proceedings of the 2005 International Symposium on Physical Design*, 2005, p. 167–174.

[197] F. Chen, L. Song, H. Li, and Y. Chen, "Marvel: A Vertical Resistive Accelerator for Low-Power Deep Learning Inference in Monolithic 3D," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 1240–1245.

[198] H. Han, J. Chung, and J.-S. Yang, "READ: Reliability Enhancement in 3D-Memory Exploiting Asymmetric SER Distribution," *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1193–1201, 2018.

[199] M. Rapp, M. B. Sikal, H. Khdr, and J. Henkel, "SmartBoost: Lightweight ML-Driven Boosting for Thermally-Constrained Many-Core Processors," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021, pp. 265–270.

[200] M. Delorme, M. Iori, and S. Martello, "Bin packing and cutting stock problems: Mathematical models and exact algorithms," *European Journal of Operational Research*, vol. 255, no. 1, pp. 1–20, 2016.

[201] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," *arXiv:1803.03635 [cs.LG]*, Mar. 2019.

[202] S. Park, J. Lee, S. Mo, and J. Shin, "Lookahead: A Far-Sighted Alternative of Magnitude-based Pruning," *arXiv:2002.04809 [cs.LG]*, Feb. 2020.

[203] M. Abdel-Basset, G. Manogaran, D. El-Shahat, and S. Mirjalili, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem," *Future Generation Computer Systems*, vol. 85, pp. 129–145, 2018.

[204] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[205] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, 3rd ed. Springer Science & Business Media, 2021, vol. 67.

[206] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in Python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[207] K.-L. Lu and Y.-Y. Chen, "ISO 26262 ASIL-Oriented Hardware Design Framework for Safety-Critical Automotive Systems," in *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, 2019, pp. 1–6.

[208] N. Das and W. Taylor, "Quantified fault tree techniques for calculating hardware fault metrics according to ISO 26262," in *2016 IEEE Symposium on Product Compliance Engineering (ISPCE)*, 2016, pp. 1–8.

[209] J. M. C. Tavares, "Automating ISO 26262 Hardware Evaluation Methodologies," Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2014.

VITA

AJINKYA BANKAR

2010	B.E., Electronics and Telecommunication Savitribai Phule Pune University Baramati, Maharashtra, India
2013	M.E., Electronics (Digital Systems) Savitribai Phule Pune University Baramati, Maharashtra, India
2022	Ph.D., Electrical and Computer Engineering Florida International University (FIU) Miami, Florida, USA

PUBLICATIONS

A. S. Bankar, S. Sha, V. Chaturvedi, and G. Quan, "Thermal Aware Neuron Sensitivity Analysis for Deep Neural Networks" (in preparation)

S. Sha, A. S. Bankar, X. Yang, W. Wen, and G. Quan, "Endurance-Aware Deep Neural Network Real-Time Scheduling on ReRAM Accelerators," *59th Design Automation Conference (DAC 2022)*, Jul. 2022. (under review)

A. S. Bankar, S. Sha, V. Chaturvedi and G. Quan, "Thermal Aware Lifetime Reliability Optimization for Automotive Distributed Computing Applications," *2020 IEEE 38th International Conference on Computer Design (ICCD)*, Dec. 2020, pp. 498-505.

S. Sha, A. S. Bankar, X. Yang, W. Wen, and G. Quan, "On Fundamental Principles for Thermal-Aware Design on Periodic Real-Time Multi-Core Systems," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 25, no. 2, Mar. 2020.