© 2021 Ameya D. Patil

HARNESSING NOISE TO ENHANCE ROBUSTNESS VS. EFFICIENCY TRADE-OFF IN MACHINE LEARNING

BY

AMEYA D. PATIL

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Professor Naresh R. Shanbhag, Chair
Assistant Professor Alexander Schwing
Professor Pavan Kumar Hanumolu
Professor Romit Roy Choudhury

ABSTRACT

While deep nets have achieved human-comparable accuracy in various classification tasks, they fall short significantly in terms of the robustness and cost metrics. For example, tiny engineered corruptions in deep net inputs can reduce their accuracy to zero. Furthermore, deep nets also require millions of trainable parameters, resulting in significant training and inference costs. These robustness and cost challenges are well recognized today. In response, there have been a plethora of works focusing on improving either the accuracy vs. robustness trade-off, or the accuracy vs. cost trade-off. However, simultaneous consideration of accuracy, robustness, and cost metrics is largely absent today, in part, because far fewer works have explored the robustness vs. cost trade-off. This dissertation aims to fill this gap by focusing explicitly on the robustness vs. cost trade-off in the presence of data noise, as well as hardware noise. Specifically, we explore how to harness the noise in order to enhance this trade-off. We characterize and improve robustness vs. cost trade-offs across diverse problem settings, ranging from beyond-CMOS hardware implementations of machine learning (ML) classifiers to efficient training of deep nets that are robust to multiple types of corruptions in their inputs. This dissertation can be roughly divided into two part, one focusing on *hardware noise* and the other on *data noise*.

In the first part, we start by focusing on harnessing noise in spintronic hardware implementations, where the logic gates become error prone when operated at lower switching energy/delay. We propose techniques to shape the resulting hardware noise distribution and to efficiently compensate it at the system-level output. As a result, we observe 1000× improvement in tolerance to gate-level switching error rates, while keeping the area/energy overhead of compensation circuits to as low as 15%. These robustness enhancements further enable 3× reduction in iso-throughput energy consumption of a binary ML classifier employed for EEG-based seizure detection.

Building on this work, we propose spintronic channel networks, exponential decay of spin current to efficiently realize multi-bit dot product computation. We employ error-prone nanomagnets as efficient stochastic slicers biased by spin currents proportional to the likelihood of the classification decision. We achieve 112×-to-22.5× and 14×-to-2.5× higher energy-efficiency over conventional spin-based and 20 nm CMOS designs, respectively, when realizing 10-to-100-dimensional binary classifiers. Furthermore, we also consider the impact of hardware noise originated from process variations and readout circuits in in-memory computing implementations employing non-volatile resistive crossbar arrays. Based on our analysis, we identify design configurations achieving the highest signal-to-noise ratio (SNR), and further estimate how such robustness trades off with the array energy consumption.

In the second part, we switch gears to improve the robustness vs. cost trade-off for deep nets in the presence of data noise. Specifically, we focus on the impact of adversarial perturbations in the deep nets inputs. We propose and validate the hypotheses about orientations of dominant subspaces of adversarial perturbations. We demonstrate how changes in the curvature of decision boundary of the deep nets affects the orientations of the adversarial perturbations. Based on these insights we demonstrate how shaped noise can be introduced as a feature to enhance robustness vs. cost trade-off in deep nets. Specifically, we propose *shaped noise augmented processing* (SNAP), a method to efficiently train deep nets that are robust to multiple types of adversarial perturbations, simultaneously. SNAP prepends a deep net with a shaped noise augmentation layer whose distribution is learned along with the network parameters using any established robust training framework. Based on extensive comparisons with nine state-of-the-art (SOTA) robust training frameworks, we show that SNAP achieves the best robustness vs. training cost trade-off. In particular, it enables 4× reduction in the training cost compared to the SOTA approach published just this last year. Furthermore, thanks to the computational simplicity of SNAP, it is the first technique of its kind that is scalable to large datasets, such as ImageNet.

We conclude by identifying potential directions for future research.

To my parents, for all their love and support.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Prof. Naresh Shanbhag. He gave me immense freedom to work on a diverse set of problems, was always available for discussions, and provided unwavering support when our ideas didn't work out as expected. I really appreciate his strategic mentorship in helping us make better research and career decisions. I am grateful for his guidance in making me a better researcher and teaching me to accept rejections in stride as a natural part of the process. I would also like to thank my dissertation committee members, Prof. Alex Schwing, Prof. Pavan Hanumolu, and Prof. Romit Roy Choudhury for providing helpful feedback for my research and dissertations. Prof. Schwing's passion and commitment for deep learning research was very inspiring, and I am grateful for all his help and support in our collaborative work in the field of adversarial robustness of deep nets. I also enjoyed interacting with Prof. Hanumolu regarding different aspects of circuit design for in-memory computing and I learned a lot from those. Finally, I really appreciated insightful discussions with Prof. Roy Choudhury about different aspects of our work.

I was fortunate to get a chance to collaborate with Sasi Manipatruni, Dmitri Nikonov, and Ian Young from Component Research group in Intel for our spintronic work and wish to sincerely thank them for their help. I also want to thank Ram Krishnamurthy, Greg Chen, Phil Knag, Huseyin Ekin Sumbul, and Raghavan Kumar for hosting me as an intern in Summer 2018 at Intel Labs, which was a wonderful experience. Being a part of Prof. Shanbhag's research group, it was great to know and collaborate with its excellent members over the years: Sujan Gonugondla, Charbel Sakr, Mingu Kang, Michael Tuttle, Hassan Dbouk, Saion Roy, Kuk-Hwan Kim, Yingyan Lin, Sai Zhang, Sungmin Lim, Yongjune Kim, Han-Mo Ou, and Hanfei Geng. I appreciate all our interactions which have shaped both my research, career, and personality.

The research in this dissertation was funded in part by SONIC, C-BRIC, AIHW programs supported by SRC and DARPA, as well as DARPA's FRANC program. I feel lucky to have been a part of the SRC community as a graduate student and interacting with other faculties and students from different universities that are part of these programs was a formative experience. Thank you to the ECE Illinois Department for providing me with this opportunity to pursue a PhD degree and recognizing my work through multiple fellowships and awards.

I am lucky to have met an amazing set of friends over the past decade of staying away from home. Thanks to Zaid Ahsan, Navjot Singh, Krishan Swaminathan-Gopalan, Mubin Khan, Shripad Gade, Anadi Chaman, Kushagra Singhal, and Anurup Ganguly for making my stay in Urbana-Champaign enjoyable. All our fun activities together – long discussions, soccer games, IPL, and cooking – helped me deal with the stress and grind of the graduate school. Pranav Madadi has been an amazing friend and a source of constant support over both IIT and UIUC years. Thank you Nachiket Dongre, Sumit Jadhav, Onkar Deshpande, Abhishek Gune, Niranjan Natekar, Piyush Kulkarni, Rutuparna Karandikar, Rohan Dutte, Omkar Waikar, Vibhav Bhave, Anad Bhattad, and Jayesh Tandale for making a point to stay in touch despite being scattered all over the world. I cherished our regular calls as well as Pune meet-ups, which I always found extremely rejuvenating.

Finally, and most importantly, I would like to thank my parents, Dhananjay and Madhura Patil for all their efforts and sacrifices in helping me to get where I am today. I cannot find words to describe their love for me. I feel devastated that my mother didn't make it to see me reaching this milestone, having lost her battle to cancer just over three months ago. I wish that her soul may rest in peace and thank her from the bottom of my heart in all my prayers.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Machine learning (ML) systems strive to perform recognition, synthesis and policy-making tasks on the given input data. The examples of such systems include, robots using cameras to learn navigation in unseen environments, voice/chat assistants, and others. Such systems, when perfected and deployed, have potential to revolutionize multiple industries. The last five years have seen a huge surge of interest in the research and development of ML systems. It stems from the recent success of deep nets (see Fig. 1.1) in achieving human-comparable accuracy across diverse well-defined tasks, such as image recognition [1], speech recognition [2], and strategy games including Chess and Go [3].

The above accuracy metric measures the probability of given ML systems making correct decisions on unseen inputs, under idealistic conditions, *i.e.* the training data distribution is the same as that of the test data and the underlying hardware provides high precision deterministic implementation. Achieving high accuracy is necessary, but not sufficient to enable wide-scale deployment of ML systems in the real world. This is because the real world, as we all are aware, is often messy, noisy, non-ideal, and at times, even adversarial. For example, the underlying hardware itself could be erroneous due to device/circuit-level variations and noise [4, 5, 6, 7, 8]. Similarly, the inputs to ML systems could be corrupted via natural or engineered perturbations [9, 10]. Hence, it is necessary for ML systems to preserve their accuracy in the presence of noisy/non-ideal conditions. In adversarial ML research area, the term *robust accuracy* is defined as the accuracy of the classifier in the presence of engineered perturbations in its inputs. Hence, in this dissertation, we extend it to define *robustness* of a given ML system as a measure of the probability of correct decisions under *non-ideal* and *noisy* conditions. While noise is omnipresent, it can be broadly classified into two categories based on its sources:

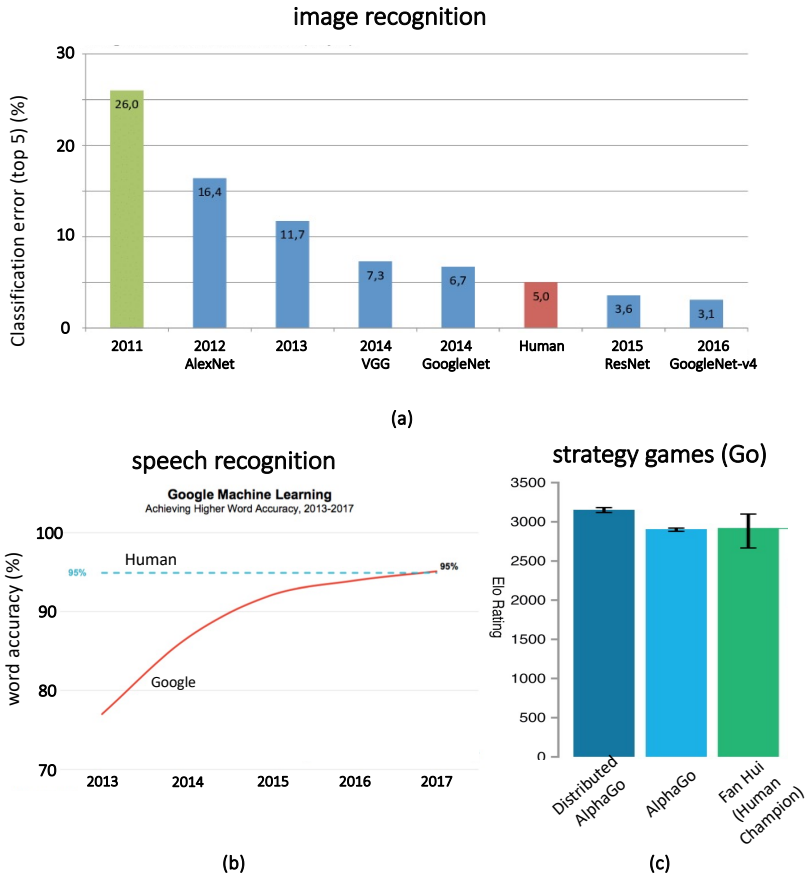


Figure 1.1: Recent success of deep nets: They have achieved state-of-the-art, human-comparable accuracy across diverse set of tasks, such as (a) image recognition [1], (b) speech recognition [2], and (c) strategy games [3].

- Data noise: this category includes natural weather-like corruptions, engineered adversarial transformations, and/or the noise in front-end sensing circuits of the ML systems.

- Hardware noise: this category includes switching errors in hardware logic gates, noise due to reduced precision and analog computation, and/or impact of process and voltage variations.

Unfortunately, human-comparable accuracy of state-of-the-art (SOTA) ML algorithms does not translate to equivalently high robustness. For example, the image recognition accuracy of deep nets gets severely lowered in the presence of common corruptions [10] (*e.g.* snow, fog, and others). Furthermore, even tiny, imperceptible perturbations in their inputs can fool deep nets to make decision errors on *every* single input in the test dataset [9, 11, 12, 13,

14]. Similarly, emerging beyond-CMOS devices suffer from increased vulnerability to hardware noise and variations [4, 8, 15, 16], causing significant drop in the inference accuracy. Hence, it is important to develop techniques that enhance robustness of ML systems, while preserving their accuracy.

The discussion of accuracy and robustness of ML systems is incomplete without the consideration of their efficiency. In this dissertation, we capture it via the cost metrics, which include both the time required to train the model (training cost), as well as energy and latency associated with producing single decision in deployment (inference cost per decision). Deep nets today require millions of trainable parameters to achieve their state-of-the-art accuracy [1, 17], resulting in significant training and inference costs. This challenge is further exacerbated due to current hardware limitations such as expensive memory accesses [18], slowing down of CMOS scaling [19, 20], and increased vulnerability to process variations in both CMOS [5] and beyond-CMOS [4, 21, 8] devices.

Today both robustness and cost challenges are well-recognized. In response, a large number of works have focused on improving either accuracy vs. robustness trade-off (see Sec. 1.1), or accuracy vs. cost trade-off (see Sec. 1.2). However, simultaneous consideration of accuracy, robustness, and cost metrics is largely absent today, in part, because far fewer works have explored robustness vs. cost trade-off. This dissertation aims to fill this gap by focusing explicitly on robustness vs. cost trade-off in the presence of data noise, as well as hardware noise (see Sec. 1.3). We outline dissertation organization in Sec. 1.4.

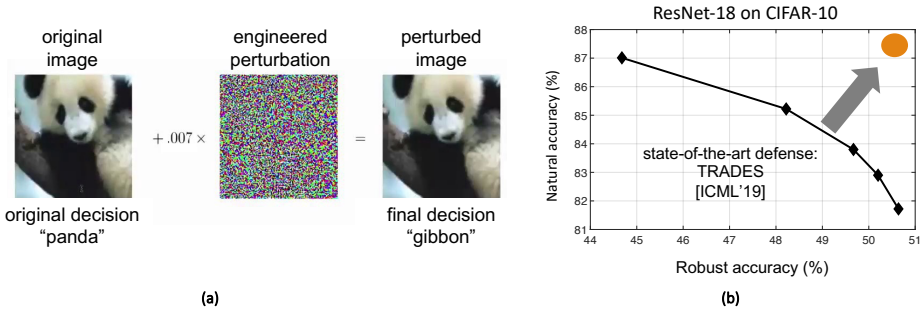


Figure 1.2: Adversarial vulnerability of deep nets: (a) example of imperceptible adversarial perturbation resulting in incorrect decision [9], and (b) trade-off between natural accuracy and robust accuracy observed for the state-of-the-art defense TRADES [22].

1.1 Accuracy vs. Robustness Trade-off

In spite of the high classification accuracy of deep nets [1], they are surprisingly easily fooled by corruptions in their inputs. For example, their accuracy reduces significantly in the presence of common weather-like corruptions [10], strange poses of familiar objects [23], and image rotations [24]. Furthermore, even tiny imperceptible changes in their inputs, *i.e. adversarial perturbations* [9], are sufficient to cause incorrect classification. Figure 1.2(a) shows an example of such perturbation, where despite being imperceptible, it changes the classification decision of a deep network AlexNet [9]. Similar perturbations can be found for any given deep net and any given input of that network. Furthermore, such a phenomenon has been observed across domains, *i.e.*, in image [9], speech [25], text [26], as well as, video games [27].

Multiple approaches to construct adversarial examples (attack methods) have been proposed [28, 11, 29, 12, 30, 31, 32, 33, 34]. While the basic idea is to move the input in the direction of increasing loss, stronger attacks [13, 12, 34, 30, 31] operate iteratively by perturbing the input in small increments. Almost all of these attacks succeed *deterministically*, *i.e.*, they cause classification errors for all inputs in the test dataset, against a naturally trained deep net. On the defense side, early attempts to provide adversarial defenses employed techniques such as introducing non-differentiable and/or randomized operations in the forward pass, ensemble of multiple networks [35, 36, 37, 38, 39, 40, 41]. However, such defenses were shown to be ineffective [42, 14], especially when the attacker takes explicit measures to evade that particular defense, *i.e.*, *adaptive* attacks. One example of adaptive attack is when an attacker employs expectation over transformation (EOT) to eliminate the impact of randomization in defense technique [42]. Thus, defending DNNs against adversarial attacks remains a formidable challenge partly due to a lack of in-depth understanding of the underlying cause of its vulnerability to adversarial perturbations.

Today *adversarial training* (AT) provides state-of-the-art (SOTA) empirical defense against adversarial perturbations. In AT, adversarial perturbations are computed during training to optimize a *robust* loss function [13]. After its initial success, multiple extensions and improvements in AT have been proposed [22, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]. In all these works, an improvement in robustness was always accompanied with some

reduction in natural accuracy, *i.e.*, all of them observe a trade-off between accuracy vs. robustness [55, 56]. For example, Fig. 1.2(b) shows such a trade-off for one of the SOTA AT framework called TRADES [22] on CIFAR-10 dataset. Notice that robust accuracy increases only at an expense of reduction in natural accuracy. Furthermore, robust accuracy is significantly (by $\sim$40%) lower than natural accuracy even for the state-of-the-art defense. We will discuss in Sec. 1.3 how any efforts in further improving the robustness requires one to start considering robustness vs. cost trade-off.

The second source of noise is the device/circuit-level non-determinism in hardware implementations. Today, all digital CMOS implementations require *deterministic* switching of their component switches and gates. The problem of robustifying computation in the presence of unreliable/noisy components was posed, as early as, in 1956 by John von Neumann in [57]. He defined robust logic network as one whose output exhibits a probability of error $p_e < 0.5$ when designed using ϵ-noisy logic gates, *i.e.*, gates whose outputs are in error with probability ϵ. It was further demonstrated that a robust logic network can be designed for any logic function provided $\epsilon \leq 0.0073$ and that it is impossible to do so if $\epsilon > \frac{1}{6}$. Later tighter upper bounds on ϵ were obtained in a series of papers [58, 59] culminating with those of Evans and Schulman [60]. In all these works, the overhead of additional gates required to achieve robustness was found to be impractically high, *i.e.* the cost associated with improving accuracy vs. robustness trade-off was too large.

More recently, a class of statistical error compensation (SEC) techniques was proposed to achieve robust computation in the presence of process/voltage induced delay variations in CMOS digital implementations of signal processing/ML applications [7, 61, 62]. In SEC, the hardware errors due to delay variations were compensated at the final system-level output to maintain the complexity of compensation circuits within 15%-to-20% of the main block computation [63]. As a result, up to 16$\times$ increase in error tolerance was achieved at a marginal drop in accuracy [7]. Such robustness gains were traded-off with up to 4$\times$ improvement in the energy-efficiency.

Finally, researchers explored modifying ML classifier training flow to enhance its robustness to hardware noise during inference. The examples include hardware-in-a-loop training of classifiers [64, 65] and hardware-aware noise injection during training [8, 16, 66, 15]. For emerging non-volatile memories, such approaches can also be combined with adaptive program-verify

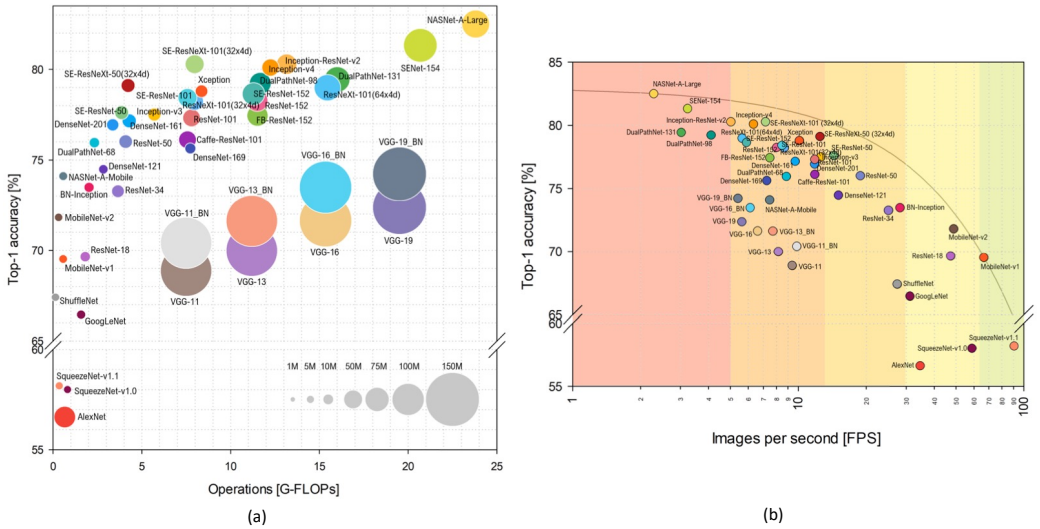


Figure 1.3: Accuracy vs. cost trade-off in deep nets [67]: classification accuracy vs. (a) number of floating-point operations and (b) images per second on NVIDIA Jetson TX1 GPU of the deep nets classifying Imagenet [68] 2012 ILSVRC dataset. The size of each ball in Fig. (a) denotes the number of parameters in the network. Millions of parameters and floating-point operations are required to achieve the SOTA accuracy.

write techniques employed to minimize the impact of write variations [21]. All these methods achieve certain improvements in the robustness to hardware noise and variations at a cost of small reduction in the accuracy.

1.2 Accuracy vs. Cost Trade-off

As shown in Fig. 1.1(a), deep nets achieved human-comparable accuracy in image classification tasks in 2014-15. Early proposals, such as VGGNet, were very expensive in terms of the number of network parameters and floating-point operations (FLOPs) required to produce a single decision. Since then, plethora of efforts have focused on improving accuracy vs. cost trade-off in deep nets as observed in Fig. 1.3(a). The proposals included both improved network architectures (*e.g.* [1, 17, 69, 70, 71, 72, 73, 74, 75]), as well as better training techniques (*e.g.* [76, 77, 78]) to achieve significant gains in accuracy vs. cost (FLOPs per inference) trade-off. The reduction in the parameter count and FLOPs also translates in improved throughput in hardware for image classification. For example, Fig. 1.3(b) shows corresponding accuracy vs. images classified per second for recently proposed deep nets running on

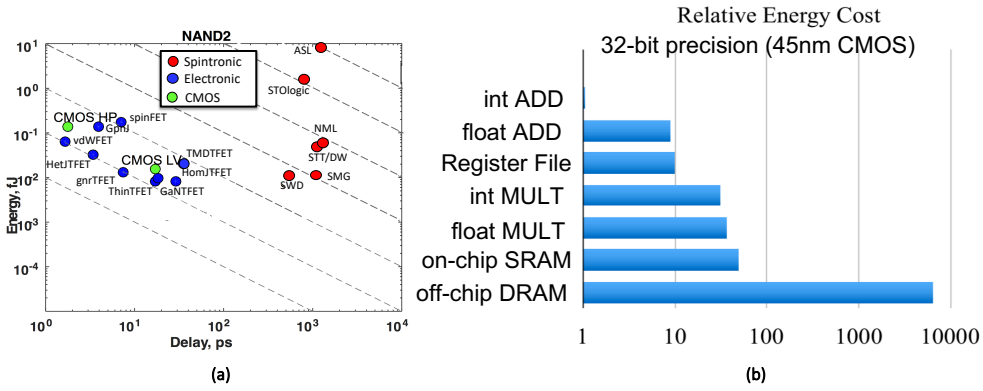


Figure 1.4: Hardware Challenges: (a) NAND2 gate switching energy vs. switching delay comparisons of the multiple beyond-CMOS devices under exploration, and their comparison with 14 nm CMOS [97, 98], (b) relative energy costs of different 32 bit operations for 45 nm CMOS implementations [18]. None of the beyond-CMOS devices on horizon significantly outperform CMOS and both on-chip and off-chip memory accesses are significantly costlier than compute operations in CMOS.

NVIDIA Jetson TX1 GPU. Researchers have also explored strategies, such as pruning [79, 80, 81, 82, 83, 84, 53, 85], aggressive quantization [86, 87, 88, 89], and neural architecture search (NAS) [90, 91, 92, 93, 94, 95, 54], to push the envelop of accuracy vs. robustness trade-off. Thanks to all these efforts, bringing the power of deep nets to the cost-constrained microcontroller-based edge devices is now a real possibility [96]. However, almost all these works do not focus on the robustness aspects discussed in Sec. 1.1. Only recently early exploration in this direction was started by combining adversarial training with pruning/NAS techniques [53, 85, 54].

All of the works discussed above improve accuracy vs. cost trade-off within the constraints of the current hardware. In order to push their envelope further, one needs to address today's hardware limitations. For instance, as the channel lengths continue to reduce beyond a few tens of nanometers, the energy and delay reductions due to CMOS scaling have stagnated. Furthermore, while multiple beyond-CMOS devices are under exploration, none of them significantly outperform CMOS in terms of gate-level switching energy and delay [97, 98], as evident in Fig. 1.4(a). Even in CMOS implementations, accessing data from memory is more expensive than doing computations on that data [18], as shown in Fig. 1.4(b). For example, fetching 32 bit word from register file or on-chip SRAM is almost 10× more energy-expensive

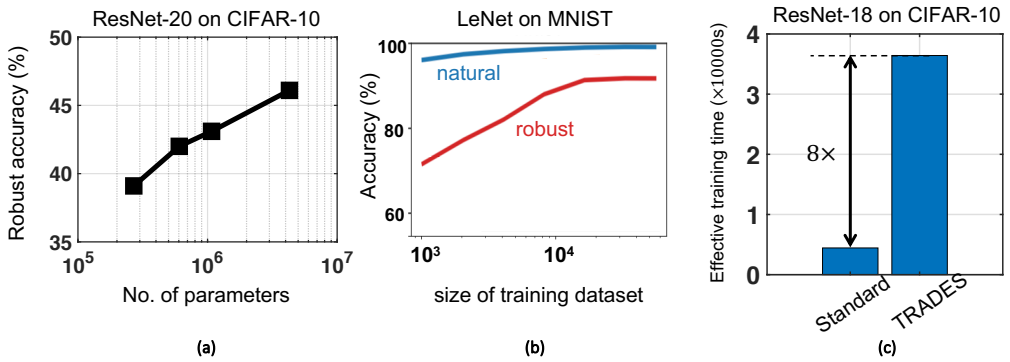


Figure 1.5: Interwoven robustness and cost challenges: Robust accuracy vs. (a) no. of model parameters [99] for ResNet-20 network trained on CIFAR-10 dataset, (b) natural and robust accuracy vs. size of training dataset for MNIST dataset [100], and (c) comparison of effective training time for standard training and TRADES adversarial training [22].

than 32 bit int ADD operation. Furthermore, off-chip DRAM accesses are almost 10× more energy-expensive than even iso-precision int MULT. This challenge is particularly exacerbated for deep net implementations due to their large parameter requirements. In the Sec. 1.3, we will discuss how the current approaches addressing these challenges, in fact, requires one to focus on robustness vs. cost trade-off due to hardware noise.

1.3 Dissertation Contribution: Improving Robustness vs. Cost Trade-off

As discussed in Sec. 1.1 and Sec. 1.2 the accuracy vs. robustness trade-off and accuracy vs. cost trade-off are primarily explored independently today. However, they start getting interwoven as one attempts to design ML implementations at the limits of accuracy, robustness, and efficiency. For instance, while pushing the envelop of accuracy vs. robustness trade-off via AT, one begins to run into prohibitive increase in training and inference costs as shown in Fig. 1.5, *i.e.* one needs bigger models [99, 104] (Fig. 1.5(a)), larger training dataset [100, 105] (Fig. 1.5(a)), and at times, even 8×-to-10× more complex training algorithms [22, 106, 107]. On the other hand, quest for achieving leaps of energy-efficiency in hardware requires addressing the robustness challenge in the presence of hardware noise [108, 65, 5]. For instance, prominent beyond-CMOS devices, such as RRAM and spin-based devices,

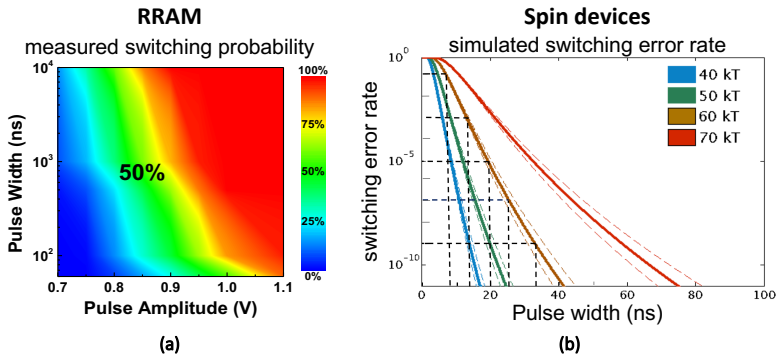


Figure 1.6: Intrinsically non-deterministic nature of emerging beyond CMOS devices: (a) measured switching probability vs. pulse width vs. pulse amplitude trade-off for RRAM devices [101], (b) simulated switching error rate vs. pulse width trade-off for spin-based devices with different energy barriers [102, 103].

are inherently non-deterministic in nature [102, 101, 103]. Their switching error rates increase with the reduction in their switching energy/delay as shown in Fig. 1.6. Similar trends are also observed in one of the promising approach called *in-memory computing* (IMC), where one strives to embed computation within the bitcell array (BCA) and its peripherals in order reduce the memory access costs. Due to severe area constraints in the BCA, such in-memory computation often needs to happen in the analog domain along bitlines (BLs) by using change in BL voltage ΔV_{BLB} to represent multi-bit information (see *e.g.* Fig. 1.7(a)). With decreasing BL discharge, a key knob for reducing energy cost of SRAM read, the variations in ΔV_{BLB} increase as shown in Fig. 1.7(b) [65]. It adversely affects the classifier accuracy (also shown in Fig. 1.7(b) [65]) indicating the need for improving robustness against hardware noise to achieve maximum energy benefits.

In summary, improving the accuracy vs. robustness trade-off in the presence of data noise exacerbates the need for cost-efficiency, while one needs to achieve robustness against hardware noise to further enhance the accuracy vs. cost trade-off. To address this predicament, we believe that one needs to focus explicitly on improving the robustness vs. cost trade-off. Specifically, we ask following fundamental questions:

> *How should the noise be managed in order to enhance robustness vs. cost trade-off in ML systems? Is noise always a problem that should be mitigated? Can it be exploited as a feature?*

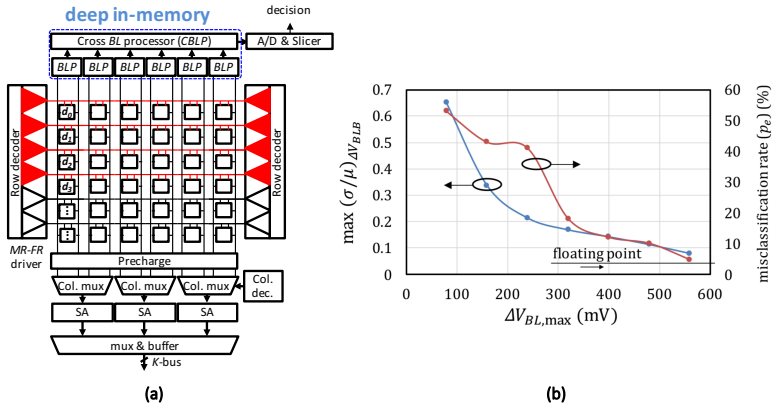


Figure 1.7: Need for robustness against hardware noise in in-memory computing: (a) deep in-memory architecture (DIMA), and (b) variations in ΔV_{BLB} and their impact on classifier accuracy as a function of maximum BL discharge, a key knob controlling energy consumption.

This dissertation attempts to address these questions in diverse problem settings across the design stack. Based on our work, we find that noise need not always be mitigated, but the robustness vs. cost trade-off in the presence of both hardware noise and data noise can be enhanced via *noise shaping*. For example, in Chapter 2, we show how hardware noise distribution in spintronic digital implementations can be shaped to achieve robust computation even in the presence of error-prone logic gates. As an another example, in Chapter 7, we demonstrate how *shaped noise* can be specifically introduced in the deep nets to achieve high adversarial robustness at significantly lower training cost.

1.4 Dissertation Organization

Our contributions in this dissertation are organized as follows:

Chapter 2 explores how the noise distribution in spin-based digital implementations can be shaped by exploiting inherent trade-off between error rate, energy, and delay for spin-based logic gates. Such noise shaping enabled efficient error compensation via a Shannon-inspired model of computation. Resulting classifier implementation is 1000× more tolerant to gate-level switching error rate compared to the conventional implementation. These robustness improvements are translated into energy-efficiency gains.

Chapter 3 demonstrates how the impact of spin device noise is fundamentally changed via novel circuit design and algorithmic techniques to

achieve energy-efficient inference implementations. In particular, this chapter presents *spin channel networks (SCN)* – a novel spin-based circuit design approach that exploit exponential decay of spin current to efficiently realize multi-bit dot product computation. Furthermore, Adaptive Boosting (AdaBoost) framework is employed to design multiple isolated tiny spin channel networks (t-SCNs) that work in unison to solve an arbitrary binary classification task. Such boosted t-SCNs achieve 112×-to-22.5× and 14×-to-2.5× higher energy-efficiency over conventional spin-based and 20 nm CMOS designs, respectively, when realizing 10-to-100-dimensional binary classifiers.

Chapter 4 proposes an MRAM-based deep in-memory architecture to achieve multi-bit matrix-vector multiplication (MVM) within a single read operation of MRAM bitcell array (MRAM-BCA). The MRAM-BCA peripheral circuits are modified to potentially achieve such multi-bit computation at 20× and 10× lower energy and delay, respectively, compared to digital MRAM implementation. The robustness challenges that emerge as a byproduct of achieved energy benefits are further characterized, and techniques to address them are discussed.

Chapter 5 presents signal-to-noise ratio (SNR) analysis for in-memory computing in resistive crossbar arrays. Specifically, we show how the circuit-level specifications of peripherals affect SNR as a function of array size, sensing circuit impedance, as well as energy per operation for PCM, MRAM, and RRAM. The noise sources considered are mismatch in the conductance and input digital-to-analog converters (DACs), as well as, clipping and quantization noise in the readout circuits. We conclude by proving some design guidelines for appropriate noise budgeting in order to achieve accurate in-memory matrix vector multiplication in the in-memory crossbar arrays.

Chapter 6 aims to develop deeper understanding of geometric orientations of adversarial perturbations of deep nets via subspace analysis for image classification tasks. Specifically, we propose and validate the hypotheses about orientations of dominant subspaces of adversarial perturbations. We demonstrate how changes in the curvature of decision boundary of the deep nets affects the orientations of the adversarial perturbations. The hypotheses and insights developed in this chapter lead to noise shaping and augmentation techniques proposed in Chapter 7.

Chapter 7 explores how shaped noise can be employed as to enhance adversarial robustness vs. cost trade-off in deep nets based on the insights

from Chapter 6. Specifically, we propose *shaped noise augmented processing* (SNAP), a method to efficiently train deep nets that are robust to multiple types of adversarial perturbations, simultaneously. SNAP prepends a deep net with a shaped noise augmentation layer whose distribution is learned along with the network parameters using any established robust training framework. Based on extensive comparisons with nine state-of-the-art (SOTA) robust training frameworks, we show that SNAP achieves the best robustness vs. training cost trade-off. Furthermore, thanks to the computational efficiency of our approach, we report *for the first time* ResNet-50 (ResNet-101) networks on ImageNet that achieve $> 30\%$ robust accuracy against the union of $(\ell_\infty, \ell_2, \ell_1)$ adversarial perturbations.

Chapter 8 outlines a few potential extensions and future research directions.

CHAPTER 2

ROBUST SPINTRONICS VIA SHANNON-INSPIRED APPROACH

2.1 Overview

As we discussed in Chapter 1, CMOS scaling is slowing down and none of the beyond-CMOS devices are yet able to significantly outperform CMOS [97, 98]. Spin-based computational devices built with nanomagnets and spin-polarized transport have emerged as a viable beyond CMOS option, due to their following favorable attributes: (i) non-volatility, (ii) higher logical efficiency, and (iii) high integration density and compatibility with the state-of-art back-end electronics manufacturing processes. These devices are a subset of the beyond-CMOS devices which include devices based on electron spin [109, 103] and magneto-electric [110, 111] phenomena. However, spin-based devices are not competitive to CMOS [98], in terms of switching energy and delay, due to their high energy-delay requirements to achieve deterministic switching [102, 112, 113, 114]. As switching energy or delay is reduced, their switching error probability increases, rendering them incompatible with the required determinism of the digital logic.

In this chapter, we demonstrate how spin-based digital implementations of ML classifiers can be made robust to a significant increase in the switching error probability of their component logic gates. We employ the Shannon-inspired model of computation [5] to enhance such gate-level switching error tolerance. In the Shannon-inspired framework, hardware errors are engineered and then efficiently compensated via the introduction of tailored redundancy, in the spirit of Shannon's theory for communications [115]. The detailed contributions of this chapter are as follows:

- We characterize the ϵ-energy-delay trade-off for ASL gates to enable non-uniform ϵ assignments across logic gates.

- We propose logic-level path delay reallocation techniques to assign appropriate error rates to individual gates such that the resulting output error distributions are shaped to facilitate error compensation.
- We propose a novel maximum likelihood (ML) error compensation scheme that exploits these shaped output error statistics to compensate the errors efficiently.
- We demonstrate a 1000× higher average error rate tolerance and a 3× lower energy-per-decision for an ASL-based digital support vector machine (SVM) implementation, while maintaining its system-level classification accuracy.

2.2 Background

2.2.1 All Spin Logic Device

Figure 2.1(a) shows a diagram of an ASL inverter. It consists of two nanomagnets separated by a conducting channel of length L. The input magnet (M_{in}) polarizes the supply current passing through it. This creates a spin concentration gradient and propagates a spin current in the channel of length L. This spin current, in turn, exerts a torque on the magnetization of the output magnet (M_{out}) forcing it to switch. Since the magnets are non-volatile, they retain the magnetization vector state when the supply current is switched off.

Electrical current in the order of $10\,\mu A$-to-$100\,\mu A$ is required to generate sufficient spin current to switch the output magnet. Since the nanomagnets and the spin channel are metallic, the equivalent electrical resistance across the nanomagnet-channel stack is small (few Ωs), enabling these devices to operate at ultra-low supply voltages. However, the electrical current through the input nanomagnet flows irrespective of output activity, causing high static energy consumption. The nanomagnets, being non-volatile, retain the magnetization vector state even when the supply current is switched off. Hence, researchers propose to clock these devices via a MOSFET [116, 117], operating in the linear region, which acts as a switch turning ON the ASL device only when it needs to compute as shown in Fig. 2.1(a). The ON

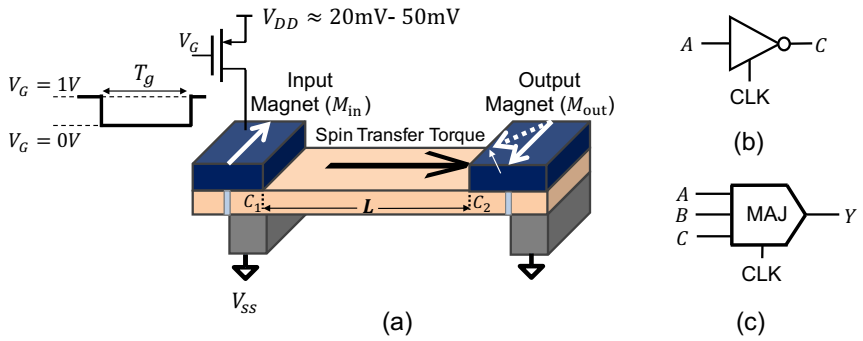


Figure 2.1: All Spin Logic (ASL) (a) diagram of clocked ASL inverter gate [118, 117], (b) clocked ASL inverter symbol, and (c) clocked ASL 3-majority gate symbol.

duration T_g of clock can be externally controlled for each gate. Thus, the energy consumption of the clocked ASL gates is completely determined by T_g and the ON current of the gating MOSFET. Figure 2.1(b) and 2.1(c) show the logical symbols for clocked ASL inverter and 3-majority gate, respectively. Pajouhi *et al.* [116] proposed to share a single MOSFET across multiple nanomagnets by electrically stacking their supply terminals in series to significantly amortize the clock pulse generation and MOSFET switching overheads. In this work, we assume such amortization described in [116] and focus on the impact of gate-level switching errors on the final output.

2.2.2 Support Vector Machine

Linear SVM [119] is a simple and popular machine learning algorithm for binary classification. The SVM learns a hyperplane to separate the training feature vectors into two regions, each corresponding to one class, as shown below:

$$\mathbf{w}^T\mathbf{x} + b \underset{\hat{z}=-1}{\overset{\hat{z}=1}{\gtrless}} 0$$

where $\mathbf{w}$ and b denote the trained weight vector and bias representing the separating hyperplane, respectively, $\mathbf{x}$ denotes the N-dimensional input feature vector, and $\hat{z}$ denotes the predicted label. If the true label is denoted by z, the accuracy of SVM is given by the probability of classification error $p_e = \Pr\{\hat{z} \neq z\}$, which can be empirically estimated for a given dataset.

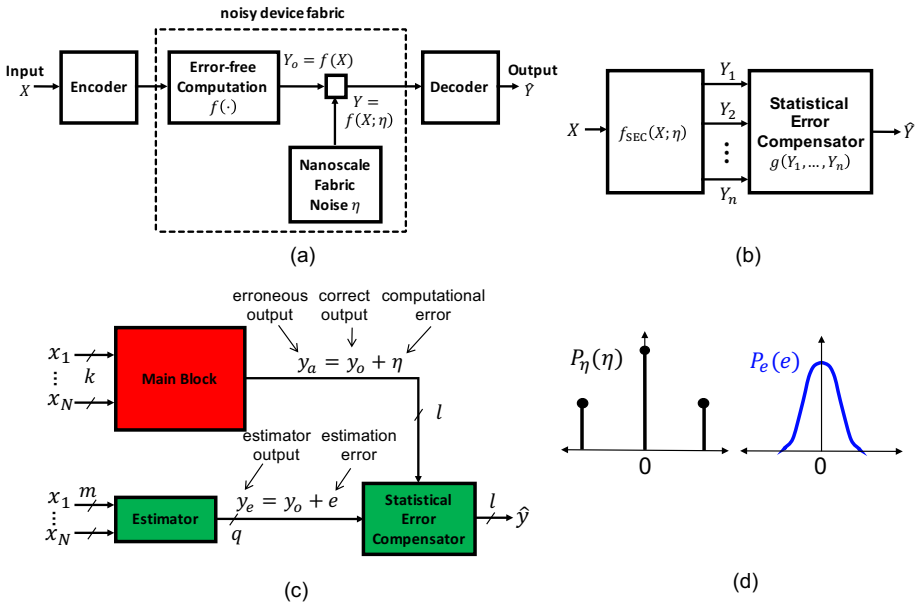


Figure 2.2: The Shannon-inspired model of computation: (a) model, (b) statistical error compensation (SEC), (c) algorithmic noise tolerance (ANT), a special case of SEC, where the error compensator combines two unreliable outputs y_a and y_e, (d) illustrative distributions of computational error η, estimation error e that lead to a low-complexity and accurate error compensator.

2.2.3 Shannon-inspired Model of Computation

The Shannon-inspired model of computation [5] (Fig. 2.2(a)) comprises an encoder, a noise-free computation of the desired correct output $Y_o = f(X)$ being corrupted by noise in nanoscale fabrics parametrized by variable η to generate the observed output $Y = f(X;\eta)$ of the error-prone device fabric (the channel), followed by the decoder that recovers the corrected output $\hat{Y}$. In Fig. 2.2(a), all variables $(X, Y_o, \eta, Y, \hat{Y})$ are random variables. In this chapter, we use capitals to denote random variables and small letters to denote their particular instance. For example, Y denotes a random variable, while y denotes a specific value of Y.

Statistical error compensation (SEC) (Fig. 2.2(b)), one class of the design techniques within the Shannon-inspired framework [5, 120], introduces a statistical error compensator block as a decoder, which combines multiple unreliable outputs $Y_1, \ldots, Y_n$ to compute corrected output $\hat{Y}$. Algorithmic noise tolerance (ANT) (Fig. 2.2(c)) is a special case of SEC where the error compensator combines two unreliable outputs y_a and y_e. ANT consists of a main block designed using unreliable/noisy device fabric accounts for 85%-

90% of total gate count complexity. It strives to compute correct output y_o, but ends up computing y_a due to the unreliability of the underlying device fabric. ANT augments the main block with a low complexity estimator that computes an estimate y_e of the correct output y_o. Under the assumption of additive noise model, the main block and estimator outputs are described as follows:

$$y_a = y_o + \eta \tag{2.1}$$

$$y_e = y_o + e \tag{2.2}$$

where η is a system-level hardware error observed at the main block output, and e is the estimation error incurred due to inherent lower complexity of the estimator.

The estimator and the error compensator are designed using reliable, and hence energy-inefficient, circuits, constituting the error compensation overhead in ANT. Hence, their combined complexity (in terms of gate count) needs to be significantly ($\approx$ 5-to-10$\times$) smaller than the main block. Previously, it has been shown that [7, 121, 122] the complexity of the error compensator can be reduced by shaping the distributions of η and e, $P_\eta(\eta)$ and $P_e(e)$, respectively, to be disparate from each other as shown in Fig. 2.2(b) and 2.2(c). In particular, a dense $P_e(e)$ is realized by introducing a reduced-precision estimator, while a sparse $P_\eta(\eta)$ is realized by permitting MSB errors in the LSB-first architectures [121, 123, 63, 122]. Various design techniques to reduce the overhead of the estimator and the error compensator have been proposed [63, 124, 125, 123].

2.2.4 Mutual Information (MI)

The mutual information (MI) $I(X;Y)$ between two random variables X and Y quantifies the amount of information conveyed about X by knowing the value of Y, and vice versa. The MI $I(X;Y)$ is defined as:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{2.3}$$

where $H(X)$ and $H(X|Y)$ denote the entropy of X and conditional entropy of X given Y, respectively. The entropy $H(X)$ of a random vari-

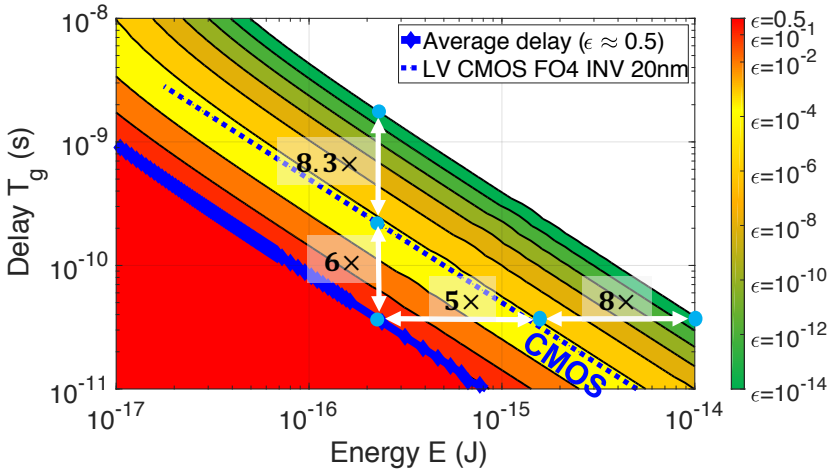


Figure 2.3: Trade-off between switching error rate ϵ, switching energy E_g, and switching delay T_g for a clocked ASL inverter gate.

able X quantifies the uncertainty about the value of X, and is a function of its probability distribution. In this chapter, we use MI metric to show that the Shannon-inspired model of computation (Fig. 2.2) enhances the MI $I(Y_o; Y_a, Y_e)$, thereby enabling an accurate recovery of y_o from y_a and y_e.

2.3 Modeling Stochasticity of ASL Devices

In this section, we develop a gate-level model to capture the inherent device-level stochasticity of ASL at the circuit and architecture level. Even after receiving supply current I_{on} ($> I_{\text{crit}}$) at the input nanomagnet, the output nanomagnet of the ASL gate may not switch due to the presence of Langevin thermal noise [102, 112, 113, 114], where I_{crit} denotes the minimum current required for nanomagnetic switching. In this chapter, we refer to this probabilistic event as *switching error*, and its probability ϵ as the *switching error rate*. In [102], an analytical expression for ϵ was derived by employing the Fokker-Planck equation for magnetization vector switching dynamics governed by the fundamental LLG equation and was validated against Landau-Lifshitz simulations of a macrospin including appropriate thermal field. This analysis indicates a gate-level trade-off between switching error rate ϵ, the switching energy E_g, and the switching delay T_g of ASL gates.

Figure 2.3 shows the iso-error rate delay vs. energy contours of an ASL inverter at various error rates. As expected, the error rate decreases with

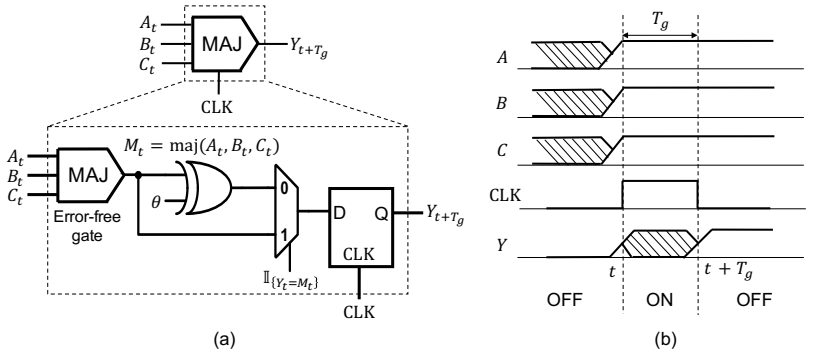


Figure 2.4: The modified ϵ-noisy gate model for clocked ASL: (a) a gate-level schematic emulating the stochastic behavior of a non-volatile, clocked ASL 3-majority gate, and (b) timing diagram illustrating the phase where the gate is ON and OFF.

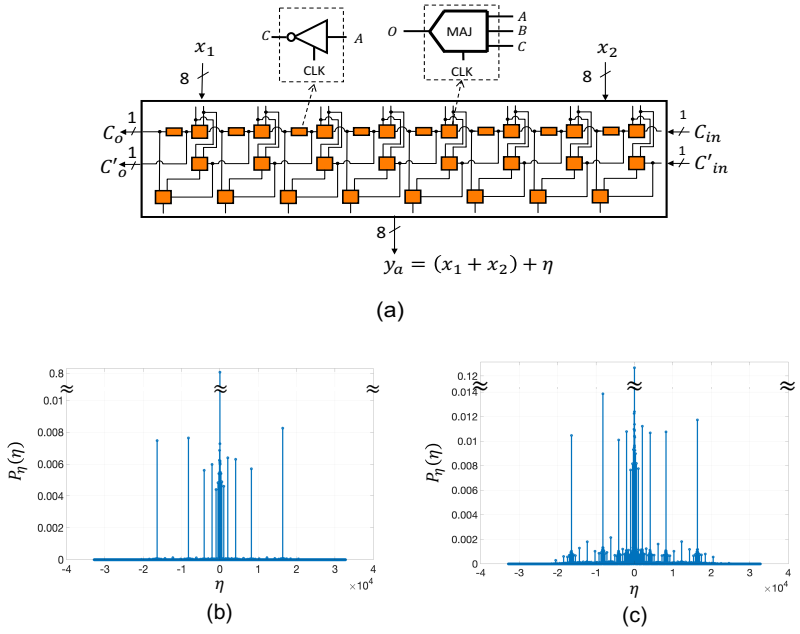


Figure 2.5: RCA with gate-level uniform error rate ϵ assignment operating at total delay of 1.24 ns: (a) schematic of 8 bit RCA showing all gates operating at $\epsilon = \epsilon_{\text{cp-avg}} = 10^{-2}$, and error distribution $P_\eta(\eta)$ for a 15 bit RCA (b) when $\epsilon = \epsilon_{\text{cp-avg}} = 10^{-2}$, $E_{\text{RCA15}} = 90\,\text{fJ}$, and (c) when $\epsilon_{\text{cp-avg}} = 10^{-1}$, $E_{\text{RCA15}} = 60\,\text{fJ}$, where E_{RCA15} denotes total switching energy of 15 bit RCA.

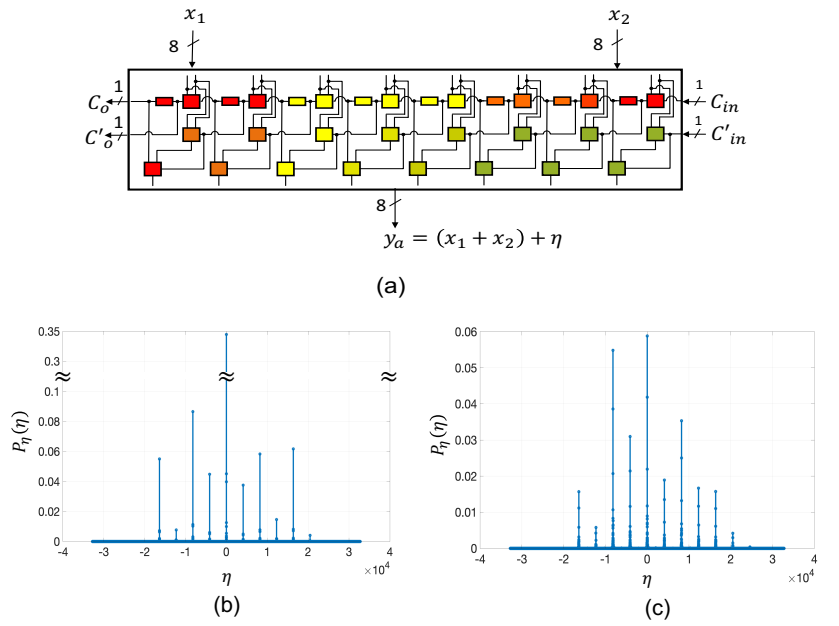


Figure 2.6: RCA with shaped error statistics operating at total delay of 1.24 ns: (a) schematic of an 8 bit RCA illustrating spatial distribution of gate-level ϵ after applying PDB and PDR, and error distribution $P_\eta(\eta)$ for a 15 bit RCA (b) when $\epsilon_{\text{cp-avg}} = 10^{-2}$, $E_{\text{RCA15}} = 90\,\text{fJ}$, and (c) when $\epsilon_{\text{cp-avg}} = 10^{-1}$, $E_{\text{RCA15}} = 60\,\text{fJ}$. The colors in (a) approximately convey the error rates of the gates as per the color code in Fig. 2.3.

increasing energy or delay. In fact, when $I_{\text{on}} \gg I_{\text{crit}}$, the expression for ϵ [102] can be simplified via Taylor series approximation to:

$$\epsilon(E_g, T_g) = \beta \exp(-\zeta\sqrt{E_g T_g}) \tag{2.4}$$

where β and ζ are device-dependent constants A 3-majority ASL gate operates with error rate of $\epsilon(E_g, T_g)$ if all its inputs are equal, and with higher error rate of $\epsilon(\frac{E_g}{3}, T_g)$ otherwise. In this work, we conservatively upper-bound the error rate of 3-majority gate to $\epsilon(\frac{E_g}{3}, T_g)$. Equation (2.4) explains the observed linearity of the contours at higher values of E_g or T_g in Fig. 2.3. We further note that ASL inverter consumes $8\times$ more energy compared to 20 nm CMOS FO4 inverter [103] at $\epsilon = 10^{-14}$ and at identical switching delays. Hence, ASL-based conventional digital architectures remain non-competitive with respect to present day CMOS. As ϵ is increased beyond 1%, the ASL inverter becomes more energy-efficient than CMOS, demonstrating the potential for achieving energy-efficiency, if one can tolerate such high gate-level error rates while maintaining final system-level accuracy.

We develop a modified ϵ-noisy gate model (Fig. 2.4(a)) to describe a clocked ASL gate, which comprehends its underlying stochastic behavior, while being sufficiently abstract to permit the design and analysis of complex ASL networks. The modified ϵ-noisy gate model captures: (a) the logic-level manifestation of device-level stochasticity, (b) the input dependence of ASL errors due to the non-volatility of the nanomagnets, i.e., the ASL gate makes an error only when the output nanomagnet fails to switch when it should, implying a dependence of the error event on the input data, and (c) the role of the CLK terminal in the gate operation.

Figure 2.4(b) shows the timing diagram for the modified ϵ-noisy model. The Boolean inputs A, B, and C are applied at time t. The ASL gate generates its output Y at time $t + T_g$ where T_g is the switching delay assigned to the ASL gate. The model comprises of an ideal noise-free Boolean gate whose output $M_t = \text{maj}\{A_t, B_t, C_t\}$ is EXORed with a Bernoulli random variable θ with parameter ϵ, i.e., $\Pr\{\theta = 1\} = \epsilon$. The output selector (implemented using a multiplexer in Fig. 2.4(b)) computes the final output Y_{t+T_g} by choosing either the output of the EXOR gate $M_t \oplus \theta$ or the error-free output M_t. The D flip-flop models the non-volatility, i.e., the ability to retain the output when $CLK = 0$. The EXOR gate output is chosen only if $Y_t \neq M_t$, capturing the fact that the switching error can occur only if the output nanomagnet is required to switch.

2.4 Shannon-inspired ASL Architecture

In this section, we describe how the Shannon-inspired approach can be applied to clocked ASL networks to increase their tolerance to switching errors. In Sec. 2.4.1, we propose path delay reallocation techniques that exploit the gate-level trade-off between ϵ, E_g, and T_g to shape the output error statistics and thereby ease error recovery. In Sec. 2.4.2, we propose a novel fusion block architecture to compensate for the switching errors.

2.4.1 Shaping Error Statistics

In clocked digital ASL networks, the random switching errors occur at the output of every logic gate as modeled in Sec. 2.3. The impact of such gate-

level errors accumulates as the input propagates to the final output. For example, consider a clocked ASL-based 8 bit ripple carry adder (RCA) consisting of all ASL gates operating at identical switching delay T_g, switching energy per nanomagnet E_g, and hence, identical $\epsilon(E_g, T_g)$ as shown in Fig. 2.5(a). The resulting distribution $P_\eta(\eta)$ of output error η for a 15 bit RCA is dense as shown in Fig. 2.5(b) and 2.5(c) for $\epsilon(E_g, T_g) = 10^{-2}$ and $\epsilon(E_g, T_g) = 10^{-1}$, respectively. Brute force compensation of the errors having such distributions can be computationally expensive as discussed in Sec. 2.4.2. We propose error statistics shaping techniques to impose a structure on $P_\eta(\eta)$ to reduce the complexity of error compensation.

We exploit the error rate, energy, and delay trade-off of the clocked ASL gates (shown in Fig. 2.3) to shape the distribution of error η. In particular, we control the gate-level switching delay via clock pulse width modulation as described in Sec. 2.2.1 [117, 116]. Exploiting this degree-of-freedom, we propose two logic-level delay assignment steps, namely path delay balancing (PDB) and path delay redistribution (PDR). We begin with a logic gate network with all gate delays equal to T_g. Thus, the critical paths are those with the maximum number of gates N_{cp} and therefore have the path delay $T_{\text{cp}} = T_g N_{\text{cp}}$. In PDB and PDR steps, the gate delays are reassigned at a constant switching energy (per nanomagnet) of E_g (moving vertically in Fig. 2.3) and at a constant throughput (identical critical path delay T_{cp}) as follows.

PDB

In PDB, delays of gates lying on the shorter paths are increased, at a constant energy E_g, making every gate to lie on one or more critical paths. Thus, PDB reduces error rate of gates on shorter paths, while leaving the original critical path unaltered, now containing gates with highest error rates.

PDR

In PDR, the gates delays along all critical paths are further redistributed to further enhance the sparsity of $P_\eta(\eta)$, while keeping their path delay constant. In particular, the delays of the few gates in the middle of the critical path are increased (lowering ϵ) at the expense of the reduction in the delays (increasing

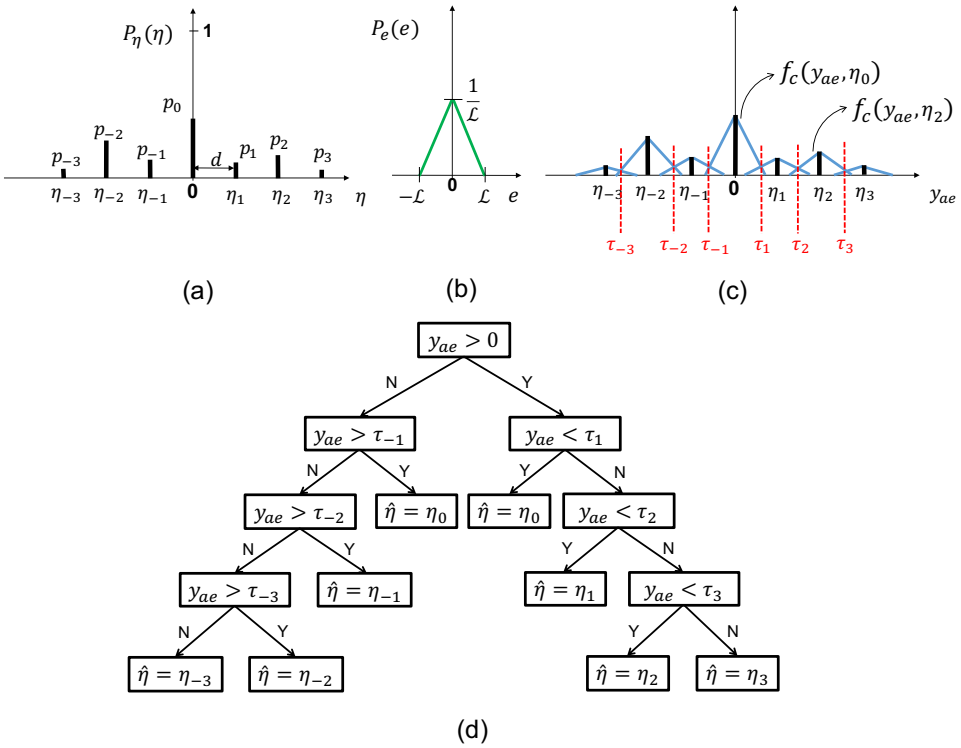


Figure 2.7: Maximum likelihood (ML) error compensation: (a) illustrative $P_\eta(\eta)$ consisting of seven distinct peaks, (b) illustrative $P_e(e)$, (c) corresponding $f_c(y_{ae}, \eta_i)$ defined in (2.8), and (d) TreeCompensator, the resulting ML error compensator having a decision-tree structure.

ϵ) of the gates lying at the beginning and at the end of the critical path. Such delay redistribution increases the error rates of the top few MSBs and bottom few LSBs, while reducing the error rates of the other bits in the middle. Doing so results in increased probability of errors having extreme magnitudes (both very high and very low), leading to a highly sparse $P_\eta(\eta)$.

We define the average device error rate of the clocked ASL network as $\epsilon_{\text{cp-avg}} = \epsilon(E_g, T_{\text{cp-avg}})$, where $T_{\text{cp-avg}} = \frac{T_{\text{cp}}}{N_{\text{cp}}}$. Note: $T_{\text{cp-avg}} = T_g$, when all gates on the critical path have equal delay. Figure 2.6(a) illustrates the spatial distribution in gate-level switching error rates (employing the color code from Fig. 2.3) for an 8 bit clocked ASL-based RCA after applying both PDB and PDR. The resulting $P_\eta(\eta)$ for a 15 bit RCA subject to PDB and PDR is shown in Fig. 2.6(b) and (c) for $\epsilon_{\text{cp-avg}} = 10^{-2}$ and $\epsilon_{\text{cp-avg}} = 10^{-1}$, respectively. Compared to the distributions in 2.5(b) and (c), the distributions in 2.6(b) and (c) are sparse, i.e., they have distinct well-separated peaks with relatively smaller spread around them.

Next, we show that error statistics shaping via PDB and PDR preserves the information in the erroneous output y_a about the correct output y_o, which can be quantified via the MI $I(Y_a; Y_o)$. We empirically estimate $I(Y_a; Y_o)$ for the 15 bit RCA example in Fig. 2.5 and Fig. 2.6. For an error-free RCA, $I(Y_a; Y_o) = 13.98$ bits, which drops to 6.18 bits, with all gates are operating at an identical error rate of $\epsilon_{\text{cp-avg}} = 10^{-1}$. The resulting $P_\eta(\eta)$ in Fig. 2.5(c) is dense. The shaped error statistics in Fig. 2.6(c) enhances MI $I(Y_a; Y_o)$ to 11.15 bits. Noted that there exist multiple methods of shaping $P_\eta(\eta)$ to increase the MI. Furthermore, a high value of $I(Y_a; Y_o)$ only guarantees the existence of an error compensation scheme to reliably recover y_o from y_a. However, such scheme need not be efficient. In Sec. 2.4.2, we derive a near-optimal low-complexity error compensation scheme that exploits the sparsity of $P_\eta(\eta)$.

2.4.2 Maximum Likelihood (ML) Error Compensator

The role of the fusion block in SEC is to compute the estimate $\hat{y}$ of the correct output y_o, as a function of two error-prone observations y_a and y_e (see Fig. 2.2(a)). One approach to make $\hat{y}$ a *good* estimate of y_o is to choose $\hat{y}$ such

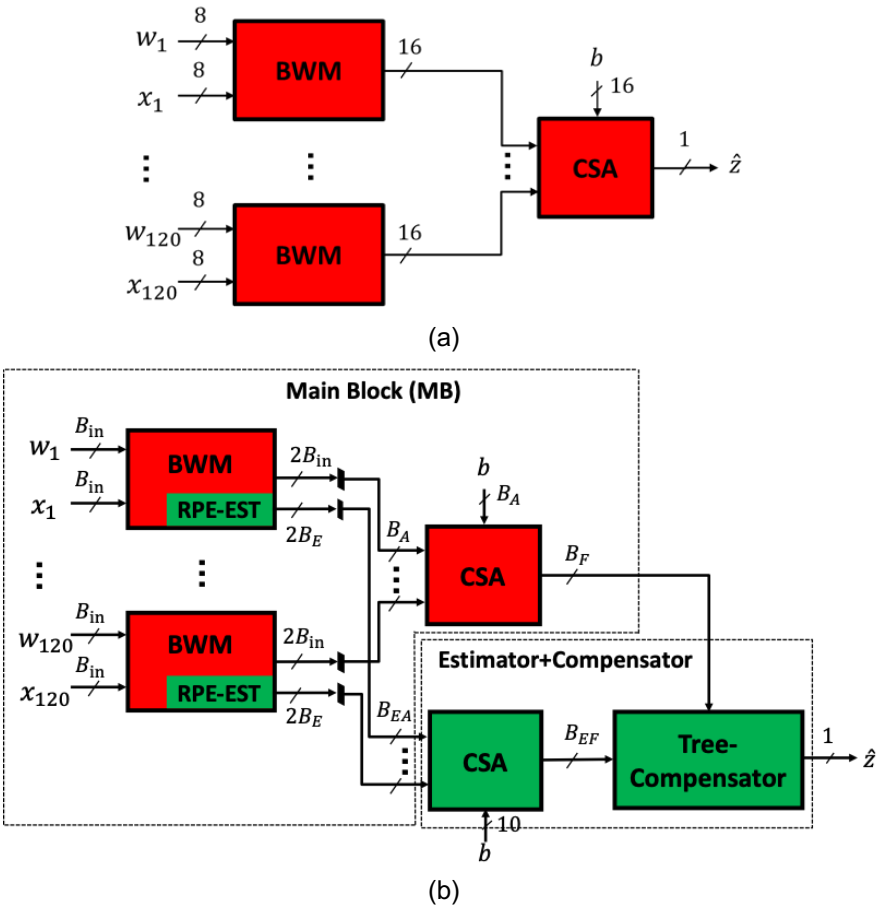


Figure 2.8: Digital clocked ASL-based 120-dimensional SVM classifiers: (a) conventional serial architecture with uniform delay assignments, (b) Shannon-inspired architecture.

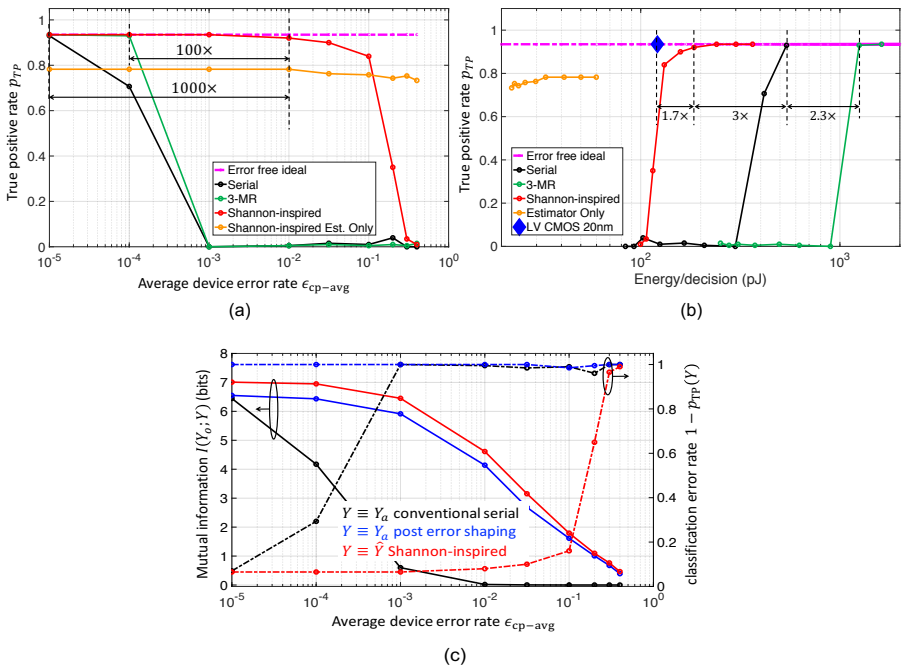


Figure 2.9: Accuracy vs. energy, error rate trade-off for different digital clocked ASL-based 120-dimensional SVM classifier implementations operating at a fixed decision delay and p_{FA} of 1%: (a) True positive rate p_{TP} vs. average device error rate $\epsilon_{\text{cp-avg}}$, (b) p_{TP} vs. total classifier energy per decision, (c) mutual information (MI) $I(Y_o;Y)$ and corresponding classification error rate $(1-p_{\text{TP}})$ vs. $\epsilon_{\text{cp-avg}}$ curves for serial architecture (black), and after shaping its error η statistics (blue), and Shannon-inspired architecture (red).

that it maximizes the likelihood of the observations y_a and y_e as follows:

$$\hat{y} = \arg\max_{y} P_{Y_a,Y_e|Y_o}\left\{Y_a = y_a, Y_e = y_e \middle| Y_o = y\right\} \tag{2.5}$$

where $P_{Y_a,Y_e|Y_o}$ denotes the likelihood of Y_a and Y_e given Y_o and y denotes a free variable in the maximization that is swept over the range of possible values of correct output y_o. Thus, $\hat{y}$ is a *maximum likelihood* (ML) estimate of y_o. In general, it can be computationally expensive to compute and maximize $P_{Y_a,Y_e|Y_o}$. However, the error statistics shaping described in Sec. 2.4.1 significantly reduces the computation of the ML estimate $\hat{y}$ as shown next.

Noting the independence of η and e conditioned on Y_o in (2.5), we get,

$$\hat{y} = \arg\max_{y} P_{\eta}(y_a - y)P_e(y_e - y) \tag{2.6}$$

We employ parametric models for $P_{\eta}(\eta)$ and $P_e(e)$ [123] as shown in Fig.

2.7(a) and 2.7(b), respectively, to simplify (2.6) to:

$$\hat{y} = y_a - \hat{\eta} \tag{2.7}$$

with $\hat{\eta}$ given as:

$$\hat{\eta} = \arg\max_{\eta_i} \left[\underbrace{p_i \mathbb{1}_{\{\eta_i - \mathcal{L} < y_{ae} < \eta_i + \mathcal{L}\}} f_e(-y_{ae} + \eta_i)}_{f_c(y_{ae}, \eta_i)} \right] \tag{2.8}$$

where $y_{ae} = y_a - y_e = \eta - e$, $\Pr\{\eta = \eta_i\} = p_i$, $\min_{i,j} |\eta_i - \eta_j| = d$, $\Pr\{|e| < \mathcal{L}\} = 1$, and f_e denotes a functional description of P_e when $|e| < \mathcal{L}$.

Given y_a, y_e, a brute-force computation of the ML estimate $\hat{y}$ requires evaluating (2.7) by calculating RHS of (2.8) for every η_i, and selecting $\eta_i = \hat{\eta}$ that maximizes it. Figure 2.7(c) illustrates plots of $f_c(y_{ae}, \eta)$ as a function of y_{ae} for all values of η. It can be observed that $\hat{\eta}$ can be approximately computed via comparisons of y_{ae} with thresholds τ_is. Thus, the ML error compensator has a decision tree structure as shown in Fig. 2.7(d), and is henceforth referred to as a TreeCompensator. The thresholds τ_is in the TreeCompensator are a function of error distributions P_η and P_e. For a given implementation, these distributions can be characterized once during simulations, or one-time calibration phase of the prototype chip. Once the thresholds are computed offline and stored, the TreeCompensator can be implemented efficiently using only a few subtracters.

2.4.3 Digital Clocked ASL-based Dot Product Implementations

Figure 2.8(a) shows the conventional serial architecture of a 120-dimensional SVM classifier. It employs 8 bit signed Baugh Wooley multipliers (BWM) and a carry save adder (CSA). All gates in this architecture operate at identical error rates. The Shannon-inspired architecture in Fig. 2.8(b) employs the conventional serial architecture as the main block (MB), and applies PDB and PDR to shape its output error distribution. Since PDB and PDR techniques make some gates operate at a lower error rate, few reliable intermediate signals in BWMs can be employed as the estimates of the BWM outputs indicated via green reduced-precision embedded estimator (RPE-EST)

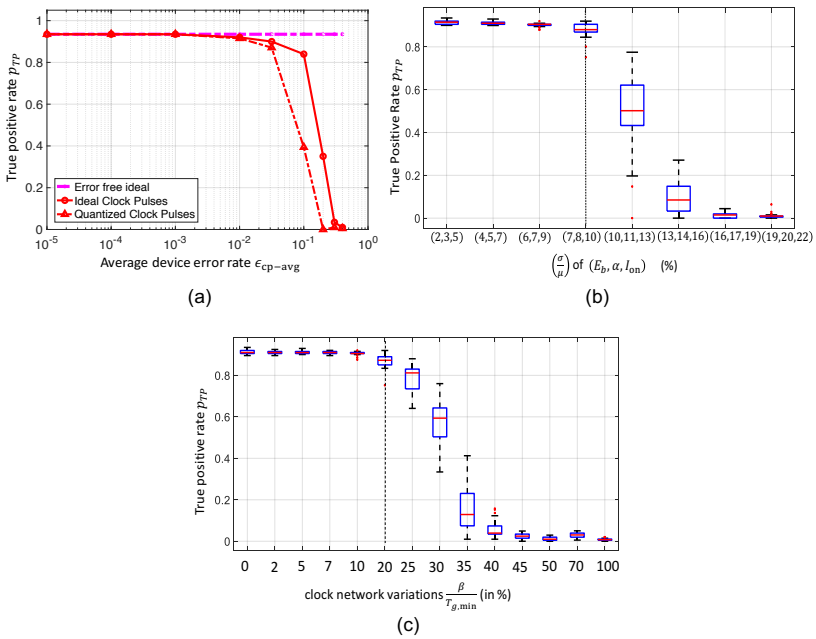


Figure 2.10: Impact of non-idealities and process variations on the Shannon-inspired implementation: (a) p_{TP} vs $\epsilon_{\text{cp-avg}}$ trade-off for Shannon-inspired implementation having 46 distinct clock pulse widths, (b) p_{TP} box plot for different levels of static within-die process variations measured in terms of $\frac{\sigma}{\mu}$ E_b, α, and I_{on} for Shannon-inspired implementation having 46 distinct clock pulse widths, and (c) p_{TP} box plot as a function of extent of dynamic clock network variations $\frac{\beta}{T_{\text{g,min}}}$ for the Shannon-inspired implementation having 46 distinct clock pulse widths, and $\frac{\sigma}{\mu}$ of E_b, α, and I_{on} set at 4%, 5%, and 7%, respectively.

blocks in BWMs, similar to techniques discussed in [63] to reduce estimator overhead. The additional overhead consists of a CSA and a digital clocked ASL implementation of the TreeCompensator derived in Sec. 2.4.2 to compute error compensated output $\hat{y}$. The bit precisions in the estimator and the compensator blocks are primarily dictated by the number of dominant peaks in the sparse shape of the η distribution of the main block. The CSA and the compensator overhead amounts to 11% of the gate complexity of the MB. We assume a low error rate $\epsilon = 10^{-4}\epsilon_{\text{cp-avg}}$ for all the gates in the CSA and TreeCompensator (marked green in Fig. 2.8(b)). We assume that the TreeCompensator computation can be pipelined since it operates only on the final outputs of the MB and the estimator. This allows the gates in the TreeCompensator to operate at a lower energy since its critical path is shorter than that of the MB.

2.5 Simulation Results

We demonstrate the benefits of the Shannon-inspired model of computation for a digital clocked ASL architecture of SVM classifier used for electroencephalogram (EEG) based seizure detection. The accuracy of the classifier is captured in terms of true positive (TP) rate p_{TP} and false alarm (FA) rate p_{FA}, where $p_{TP} = \Pr\{\hat{z} = 1|z = 1\}$ and $p_{FA} = \Pr\{\hat{z} = 1|z = 0\}$, and the probabilities are estimated empirically (via leave-one-out cross-validation) [126] for the MIT-CHB EEG dataset [127] by running extensive Monte Carlo simulations. We compare the Shannon-inspired architecture (Fig. 2.8(b)) with (i) clocked ASL-based conventional serial architecture (Fig. 2.8(a)) consisting of 54,332 gates, (ii) clocked ASL-based 3-MR architecture, which replicates the conventional serial architecture thrice and takes a bitwise majority vote on their outputs, and (iii) 20 nm LV CMOS architecture, which consists of the exact same full adder-level logic network as that of the serial architecture. We compare p_{TP} vs. energy per decision and $\epsilon_{\text{cp-avg}}$ trade-offs at a fixed decision delay of 9.7 ns and $p_{FA} = 1\%$.

2.5.1 Accuracy vs. $\epsilon_{\text{cp-avg}}$ and Energy Trade-off

We observe in Fig. 2.9(a) that Shannon-inspired architecture (Fig. 2.8(b)) can tolerate 1000× higher $\epsilon_{\text{cp-avg}}$ compared to the conventional serial architecture (Fig. 2.8(a)) while maintaining the p_{TP} close to that of the fixed-point ideal error-free architecture. In particular, the p_{TP} for Shannon-inspired architecture is close to 93% even though $\epsilon_{\text{cp-avg}}$ is as high as 1%. The 3-MR architecture tolerates an $\epsilon_{\text{cp-avg}}$ up to 0.01%. It is greater than that of the serial architecture but worse by 100× when compared to the Shannon-inspired architecture. Furthermore, we show that intermediate estimator-only output (y_e in Fig. 2.8(b)) achieves lower accuracy, emphasizing the requirement to combine the two erroneous outputs (y_a,y_e in Fig. 2.8(b)) to achieve close-to-ideal accuracy.

The Shannon-inspired architecture achieves a 3× lower energy compared to the conventional serial architecture (Fig. 2.9(b)) while maintaining $p_{TP} =$ 93%. The 3-MR architecture, however, consumes 2.3× *more* energy than the serial architecture even though it operates at a higher device error rate. This is because the energy overhead of replication offsets the energy reduction

achieved by operating at higher device error rate. However, despite its high error tolerance, the Shannon-inspired architecture still requires 1.7× more energy compared to the 20 nm LV CMOS architecture, indicating the need to explore devices with improved energy vs. error rate trade-offs and/or the use of increasingly powerful SEC techniques [128, 129, 61]. We also note in Fig. 2.9(b) that the estimator block (consisting only of the green CSA block in Fig. 2.8(b)) consumes 20% of the total energy ("Estimator Only" curve in Fig. 2.9(b)).

The reason for the effectiveness of the Shannon-inspired model in compensating for errors is the enhancement in MI $I(Y_o; Y_a)$ due to error statistics shaping via PDB and PDR as shown Fig. 2.9(c). Despite error statistics shaping, y_a remains a poor estimate of y_o, as evident from its high classification error rate $(1 - p_{TP})$. Since $I(Y_o; Y_a)$ is high, it implies that Y_o can be estimated accurately from Y_a. However, such an error compensator need not be efficient. Hence, in the Shannon-inspired model, we rely on two error-prone observations y_a and y_e to estimate y_o both efficiently and accurately. The MI $I(Y_o; \hat{Y})$ is even higher than $I(Y_o; Y_a)$ due to additional information about y_o contributed by y_e.

2.5.2 Impact of Non-idealities and Process Variations

Next, we evaluate the tolerance of the proposed Shannon-inspired architecture to various practical non-idealities, such as, finite number of distinct clock pulse widths, process variations, and clock pulse width variations. While PDB and PDR can potentially assign a unique delay to each gate, in practice, those delays need to be further quantized to take one value out of the finite set of available distinct clock pulse widths. Figure 2.10(a) shows the p_{TP} vs. $\epsilon_{\text{cp-avg}}$ curves for the Shannon-inspired architecture after quantizing the ideal clock pulse widths to 46 distinct pulse widths for the SVM implementation (Fig. 2.8(b)) consisting of 54,332 gates. The number of distinct clock pulse widths is of the same order as the number of gating domains explored in [116]. We observe negligible deterioration in the accuracy of Shannon-inspired architecture (in $\epsilon_{\text{cp-avg}} < 1\%$ regime). Such gate clock pulse width quantization enables amortization of the clock pulse generation circuitry, including the sharing of the clocking transistors across different nanomagnets [116]. The clock network design is further simplified since the

quantized clock pulse widths are integer multiples of the shortest reference clock, and multiple parallel dot products (in applications such as filter banks, neural networks) can share a single clock generation circuitry.

Process variations present an additional challenge in beyond-CMOS systems. We evaluate the tolerance of the Shannon-inspired approach to static within-die variations in three device parameters, namely, energy barrier E_b and damping coefficient α of the nanomagnets, and clocking transistor ON current I_{on}. We observe in Fig. 2.10(b) that the Shannon-inspired architecture with quantized clock pulse widths can tolerate a $3(\frac{\sigma}{\mu})$ variations of up to 24% in each of the three device parameters. When dynamic variations in the clock pulse widths are included in addition to their quantization and process variations, we find in Fig. 2.10(c) that the Shannon-inspired architecture can tolerate a maximum deviation (β) of 20% of the minimum clock pulse width ($T_{\text{g,min}}$).

2.6 Discussion

In this chapter, we demonstrated how the Shannon-inspired model of computation can be employed to make digital clocked ASL implementations robust to random gate-level switching errors. Such an approach can be extended to many other spintronic devices, such as MESO [110], CoMET [111], as long as they use nanomagnet switching for information processing. Shannon-inspired techniques have previously been applied to CMOS implementations to further reduce their energy consumption via voltage overscaling [7, 129]. In contrast, ASL/spintronics provides a new way to trade off stochasticity with energy by realizing this energy-accuracy trade-off at the device level.

This chapter also showed how the hardware noise distribution in digital clocked ASL-based implementations can be shaped to enhance the robustness vs. cost trade-off discussed in Chapter 1. Such trade-off enhancements enable the use of a highly error prone but scalable physical devices (*e.g.* ASL, MESO, CoMET, MRAM, RRAM, and others) in ML inference implementations by meeting the system-level accuracy requirements despite the device-level unreliability.

CHAPTER 3

EFFICIENT INFERENCE VIA SPIN CHANNEL NETWORKS

3.1 Overview

In Chapter 2, we demonstrated how the robustness enhancement can be exploited to improve energy-efficiency of spin-based digital implementations. However, despite 1000× improvement in the robustness to gate-level switching errors, the energy-efficiency improvements fell short of outperforming CMOS implementations. Recently, Ganguly *et al.* [130] took a physics-based approach for examining the power dissipation in spintronic switches. They identified that nanomagnetic switching consumes $10^3\times$-to-$10^4\times$ higher switching charge (Q_{sw}) compared to the CMOS inverter of comparable size. Such a large gap in the switching charge requirements underscores the fundamentally expensive nature of the nanomagnetic switching. ASL networks (Fig. 3.1(a)) use nanomagnetic switching at the output of every gate to implement digital logic. Hence, they require switching of a large number intermediate nanomagnets leading to a high energy consumption.

In this chapter, we propose *spin channel networks* (SCN) (Fig. 3.1(b)), where all intermediate nanomagnets are eliminated and all input nanomag-

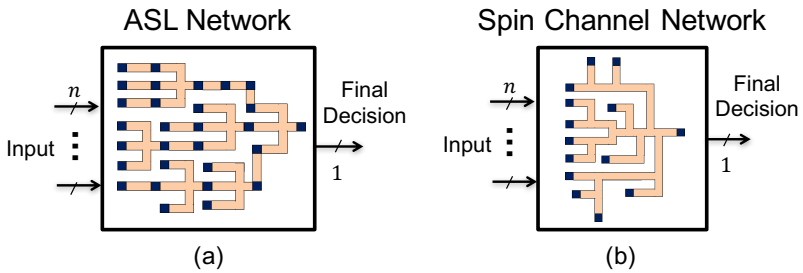


Figure 3.1: Illustration of (a) an all spin logic (ASL) network, and (b) a spin channel network (SCN) implementing an inference kernel by mapping a large n bit input vector to 1 bit decision.

nets contribute to the charge required to switch a single output nanomagnet to represent a final decision, thereby amortizing the energy consumed in switching it. It is particularly suited for inference implementations. While elimination of intermediate nanomagnetic switching is expected to enhance the energy-efficiency, it also presents following two key challenges:

(1) how does one realize arbitrary computation while accumulating analog spin currents from multiple nanomagnets?

(2) will this approach scale with the input vector dimensionality (complexity) of the inference kernel?

To address challenge (1), we show that the exponential decay property of spin current along the spin channel, a disadvantage in digital ASL networks, can be exploited to achieve energy-efficient analog dot product implementation. To circumvent the challenge (2) we employ Adaptive Boosting (AdaBoost) [131] framework to design multiple isolated tiny spin channel networks (t-SCNs) that work in unison to solve an arbitrary binary classification task. Such boosted t-SCNs achieve 112×-to-22.5× and 14×-to-2.5× higher energy-efficiency over conventional ASL-based and 20 nm CMOS designs, respectively, when realizing 10-to-100-dimensional binary classifiers.

3.2 Background

3.2.1 ASL Device Primitives

Figure 3.2 identifies two ASL primitives and their functionalities, that will be used to design SCNs in Sec. 3.3. The primitives are nanomagnet with a spin channel and a spin channel. In particular, nanomagnet takes input charge current I_c and injects proportional spin current $I_{\mathrm{s,o}}$ in the channel, where β_m is a proportionality constant that depends upon the device material and geometry, including the channel length L_c. The input spin current $I_{\mathrm{s,in}}$ into a spin channel is reduced by a factor of $e^{\frac{L}{\lambda}}$ to generate an output spin current $I_{\mathrm{s,o}}$, where L denotes channel length, and λ is the spin flip length [132, 133]. The layouts are obtained by following the λ-rules in [98].

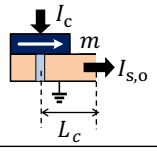
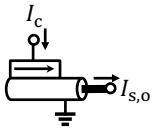
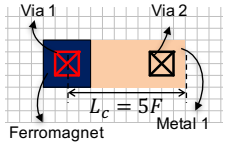
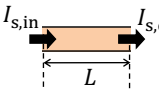
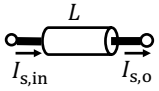
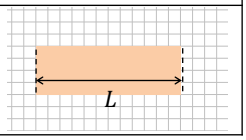
	Conceptual diagrams	Transfer function	Symbols for schematics	Layouts
Nano-magnet with a spin channel		$I_{s,o} = \beta_m I_c m$		
Spin channel		$I_{s,o} = I_{s,in} e^{-\frac{L}{\lambda}}$		

Figure 3.2: SCN primitives derived from ASL: symbol, transfer function, and layout. Each layout grid cell is of size $\frac{F}{2} \times \frac{F}{2} = 7.5\,\text{nm} \times 7.5\,\text{nm}$ [97].

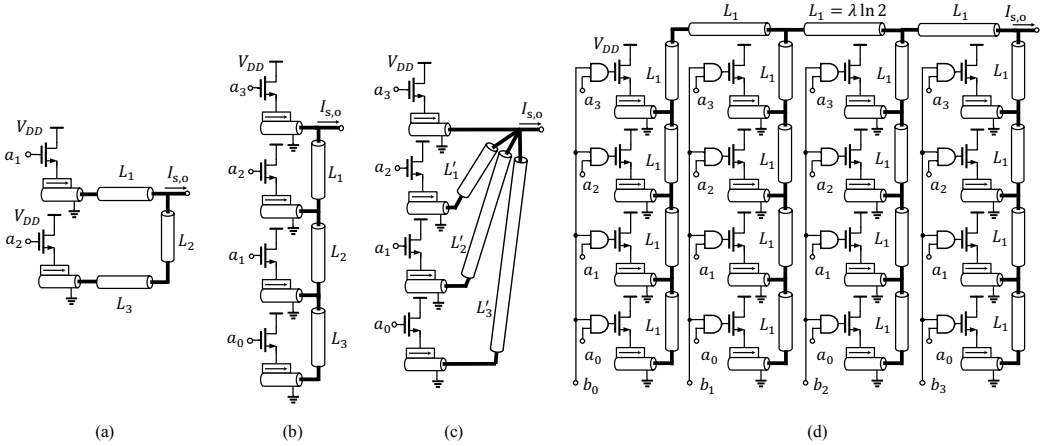


Figure 3.3: Conceptual spin channel network topologies: (a) basic, (b) *ladder*, (c) *star*, and (d) a *ladder-of-ladders* topology of a 4×4 bit spin channel network multiplier (SCNM).

3.2.2 Classifier Ensemble via Adaptive Boosting (AdaBoost)

A classifier ensemble consists of multiple weak classifiers. Each weak classifier is computationally simple but inaccurate, i.e., with p_e close to 0.5. However, decisions of the weak classifiers can be combined to obtain a highly accurate final decision. Adaptive boosting (AdaBoost) [134] is a technique to train these weak classifiers sequentially. Each weak classifier is specifically trained to correct errors made by the other weak classifiers trained earlier (see [134] for the training algorithm). Let the output label of ith weak classifier be denoted as $\hat{y}_i = f_{\mathbf{w}_i}(\mathbf{x})$, where $f_{\mathbf{w}_i}(.)$ denotes the ith weak classifier function parametrized by weight vector $\mathbf{w}_i$, which is computed during training. The final decision $\hat{y}_f$ is computed by linearly combining the weak classifier decisions $\hat{y}_i$, followed by thresholding as shown below:

$$\sum_{i=1}^{M} \alpha_i \hat{y}_i \underset{\hat{y}_f=-1}{\overset{\hat{y}_f=1}{\gtrless}} 0 \tag{3.1}$$

where output weights α_is of the linear combiner are also learned during the training phase.

3.3 Spin Channel Networks

3.3.1 Basic Concept

Spin channel networks exploit the exponential decay of spin current along spin channels for efficient computation. They compute via weighted analog accumulation of spin currents by careful choice of spin channel lengths. SCNs are composed of the two primitives defined in Fig. 3.2. The most basic SCN consists of two nanomagnets connected using spin channels having different lengths is shown in Fig. 3.3(a). The resulting output spin current $I_{\mathrm{s,o}}$, is approximately given by

$$I_{\mathrm{s,o}} = \beta_m I_c \big(a_1 m_1 e^{\frac{-L_1}{\lambda}} + a_2 m_2 e^{\frac{-(L_2+L_3)}{\lambda}} \big) \tag{3.2}$$

where $a_1, a_2 \in \{0, 1\}$ are the digital Boolean inputs, $m_1, m_2 \in \{-1, 1\}$ denote the directions of magnetization vectors of two nanomagnets, λ denotes spin

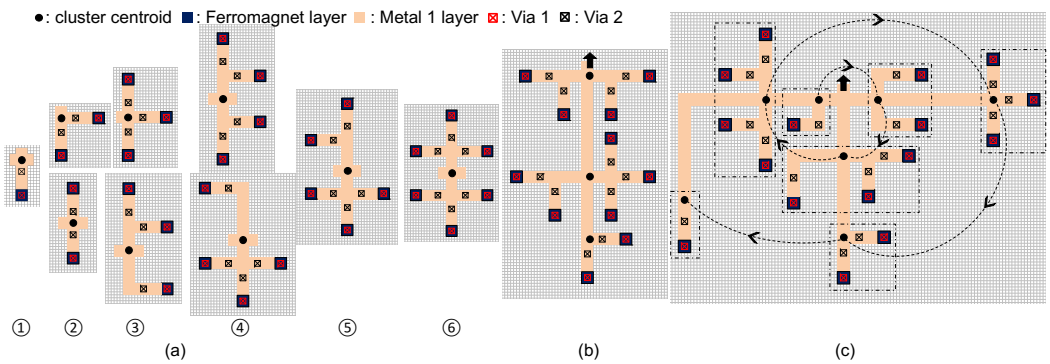


Figure 3.4: Layouts of spin channel networks: (a) a set of clusters along with their centroids, (b) a ladder topology of three clusters, (c) a star-of-ladders layout topology of a 4×4 bit spin channel network multiplier. The layout grid cell is of size $7.5\,\text{nm} \times 7.5\,\text{nm}$ [97].

flip length, I_c denotes the ON current of the NMOS transistors. Each bit a_i controls the charge current through one nanomagnet, and the corresponding spin current is weighed by a factor exponential in its channel length. More complex SCNs can be designed to achieve weighted accumulation of spin currents from M nanomagnets placed at lengths L_is, where $i \in \{0, \ldots, M-1\}$.

3.3.2 SCN Topologies

For $M > 2$, multiple circuit topologies can achieve the same input-to-output transfer function (up to a scaling constant) depending upon how the nanomagnets and spin channels are interconnected. For example, conceptual diagrams of two extreme topologies, namely the *ladder* and the *star* topology, are shown for $M = 4$ in Fig. 3.3(b) and 3.3(c), respectively. In the ladder topology, all nanomagnets share a single spin channel that connects them to the output node. The star topology, on the other hand, consists of a unique channel connecting each nanomagnet to the output node.

Weighted accumulation of spin currents in SCNs can be used to efficiently implement multiplication in analog. The $M \times N$ bit SCN multiplier (SCNM) in Fig. 3.3(d) takes two charge-domain digital operands A and B having bitwidths M and N, respectively, and generates an output spin current proportional to their product $A \times B$. The SCNM consists of $M \times N$ input nanomagnets, each contributing spin current corresponding to a partial product. Individual partial products are computed by the AND gates with bits a_is

and b_js of operands A and B, respectively. The AND gates drive the gate of the NMOS, thereby controlling the input charge current through the SCN nanomagnets. Figure 3.3(d) shows a *ladder-of-ladders* topology of a 4×4 bit SCNM, where four vertical ladders are connected horizontally in a ladder topology. Note that at all the spin channel lengths are multiples of $\lambda \ln 2$ in order to achieve appropriate weighing (in the powers of two) of spin currents corresponding to individual partial products. The output spin current of this 4×4 bit SCNM is given by:

$$I_{\mathrm{s,o}} = I_{\mathrm{s,lsb}} \sum_{i,j=0}^{3} a_i b_j 2^{(i+j)} \tag{3.3}$$

where the unit spin current $I_{\mathrm{s,lsb}}$ corresponds to the least significant partial product for a given NMOS ON current I_c. For the SCNM in Fig. 3.3(d), $I_{\mathrm{s,lsb}} = \frac{\beta_m I_c}{2^7}$. The signs of these operands can be accounted for by changing the magnetization vector directions of the corresponding magnets (for A), and by using a differential supply [135] (for B). The energy consumption of such an SCNM is given by:

$$E_{\mathrm{mult}}(A,B) = \left[I_c^2 T_g (R_{\mathrm{spin}} + R_{\mathrm{mos}}) + C_g V_g^2 + E_{\mathrm{and}} \right] \sum_{i,j=0}^{3} a_i b_j \tag{3.4}$$

where R_{spin} denotes the series resistance of the nanomagnet and channel, E_{and} denotes the energy consumed in switching of the AND gate, while R_{mos} and C_g denote the ON resistance and gate capacitance of the transistor, respectively. The gate voltage V_g is applied to switch ON the NMOS for T_g duration.

The other topologies such as *star-of-ladders*, *star-of-stars* and *ladder-of-stars* are also possible, and this topological degree of freedom will be explored while identifying the energy-efficient layout in the following Sec. 3.3.3.

3.3.3 Hierarchical Layout Construction

Topological schematic diagrams in Fig. 3.3 are idealized and convey the SCN functionality at a very high level. They neither account for spin current branching at the spin channel junction, nor the physical constraints of component placements and maintaining spin channel lengths. We address both

of these issues by developing precise layouts for SCN circuits, and obtain input-to-output transfer function from SCN layouts.

For layouts, we choose $F = 15\,\text{nm}$, where F denotes the DRAM half pitch [97, 98]. All SCN layouts need to satisfy λ-rules described (in terms of F) in [97]. For example, the layout pitch between any two contacts needs to be at least $4F$. The value of channel length L_c turns out to be $5F$ (in Fig. 3.2) as a direct consequence of λ-rule constraints. Similarly, λ-rules impose constraints on minimum distance between nanomagnet and a spin channel, and two parallel spin channels.

The layouts of more complex SCNs are particularly challenging, since there is a trade-off between satisfying λ-rule constraints and the magnitude of the output spin current, and hence the energy consumption. We propose a hierarchical construction of SCN layouts. We define nine primitive topologies referred to as *clusters* (see Fig. 3.4(a)). Each nanomagnet in a cluster contributes identical spin currents to the output node, referred to as the *cluster centroid*. The layouts of the clusters are fixed per λ-rules. These clusters can be connected in various topologies, such as a ladder, star or a ladder-of-stars, to generate layouts of more complex SCNs in a hierarchical manner. An illustrative ladder topology of three clusters is shown in Fig. 3.4(b). Once the clusters are connected, only the lengths between their centroids need to be adjusted to achieve appropriate weighing of the corresponding spin currents and simultaneously satisfy λ-rules.

Figure 3.4(c) shows a star-of-ladders layout topology of a 4×4 bit SCNM. Each cluster centroid J_k generates spin current corresponding to p_k, where p_k is defined as the sum of partial products having identical binary weight k as follows:

$$p_k = \left(\sum_{\substack{i,j=0 \\ i+j=k}}^{3} a_i b_j \right) \tag{3.5}$$

The final output spin current in Eq. (3.3) can be computed as:

$$I_{\text{s,o}} = I_{\text{s,lsb}} \sum_{k=0}^{6} 2^k p_k \tag{3.6}$$

where the binary weighing of 2^k among the spin current contributions is achieved by adjusting spin channel lengths between them. The clusters are sequentially placed in a spiral order along three ladders as shown in Fig.

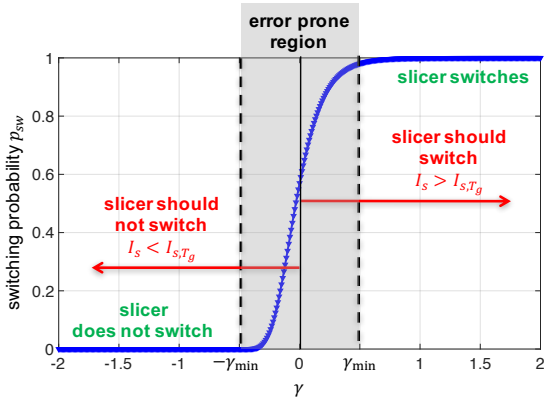


Figure 3.5: The switching probability p_{sw} of stochastic slicer as a function of γ (given approximately by Eq. (3.7)) with $E_b = 35kT$ and $T = 300\,\text{K}$. The dotted outlines of the gray region mark the minimum energy operating point (MEOP) at $\gamma_{\text{min}} = 0.5$ corresponding to slicer switching accuracy of 99%.

3.4(c). Thus, along a single ladder, the minimum channel length between any two consecutive clusters corresponds to the spin current weighing of 2^{-3}, thus allowing sufficient spacing to satisfy λ-rules. Hence, this layout topology effectively spreads out the nanomagnets radially. The actual channel lengths in the layout are chosen via extensive simulations using SPICE-based circuit models of spin current injection and propagation in spin devices [133] in order to account for spin current branching at the spin channel junctions.

3.3.4 Stochastic Slicer

The output of SCN circuits is an analog spin current. We use a nanomagnet as the final decision device. It acts as a sink for the spin current and thresholds it to produce the final decision represented by its magnetization vector. The nanomagnetic switching is stochastic due dominant thermal noise in the nanomagnet [102]. Hence, we refer to such decision generating nanomagnet as a *stochastic slicer*. The stochastic slicer switches when the magnetization direction of the corresponding nanomagnets flips due to the input spin current.

In this work, we operate the stochastic slicer for the fixed duration of T_g. If the slicer switches during this duration, it corresponds to final decision $\hat{y} = 1$, otherwise, $\hat{y} = -1$. The slicer is reset after every decision. For a given duration T_g, the probability that slicer switches p_{sw} can be approximated as

a function of its input spin current I_s as follows:

$$p_{\text{sw}}(I_s) \approx \left(\frac{1}{2}\right)^{\left[\left(\frac{\beta_1}{\ln 2}\right)^{-\gamma}\right]} \tag{3.7}$$

where $\gamma = \left(\frac{I_s - I_{s,T_g}}{I_{s,T_g}}\right)$, I_{s,T_g} denotes the spin current for which $p_{\text{sw}}(I_{s,T_g}) \approx 0.5$, and $\beta_1 = \frac{\pi^2 E_b}{4kT}$. In particular, E_b denotes the energy barrier of the nanomagnet, while k and T denote Boltzmann constant and absolute temperature, respectively. The spin current value I_{s,T_g} is a device-dependent constant for a given switching duration T_g. Equation (3.7) provides a good approximation of the p_{sw} expression in [102], when $|I_s|, I_{s,T_g} \gg I_{\text{crit}}$, where I_{crit} denotes the minimum spin current required for nanomagnet to switch with probability 1 as $T_g \to \infty$. The stochastic slicer strives realize the thresholding operation:

$$I_s \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} I_{s,T_g} \tag{3.8}$$

However, due its stochastic nature, stochastic slicer probabilistically makes switching errors, i.e., it switches with certain non-zero probability even when $I_s < I_{s,T_g}$, and vice versa as shown in Fig. 3.5. Thus, there exists a trade-off between input spin current magnitude I_s (proportional to energy consumption) and switching probability p_{sw} (see Eq. (3.7)). There exists a minimum energy operating point (MEOP) for a target switching probability. For example, as shown in Fig. 3.5, $|I_s| > 1.5 I_{s,T_g}$ to achieve slicer switching accuracy of 99%. This MEOP of slicer dictates the minimum charge current $I_{c,\min}$ through each nanomagnet required for SCN-based binary classifier to achieve certain classification error probability p_e. For a fixed decision delay and error probability, MEOP for SCN-based binary classifiers is uniquely defined by the value of $I_{c,\min}$ as shown in Sec. 3.5.

3.3.5 CMOS Driver

Figure 3.6(a) and (b) show the abstract model and transistor-level schematic of the CMOS input driver in a 14 nm technology, respectively. The nanomagnet is controlled by an NMOS, which should switch ON only when $a_i = 1$ and $b_j = 1$. This is achieved via a CMOS NOR gate driving the gate of

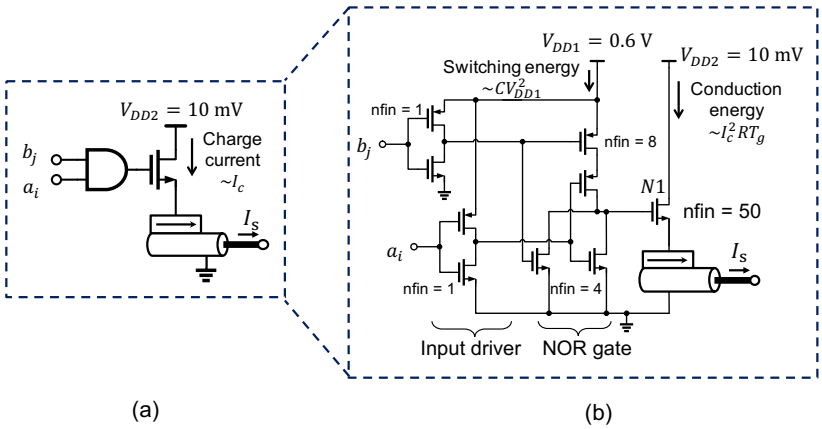


Figure 3.6: Detailed schematic of the CMOS input driver designed and simulated using 14 nm HP FinFET ASU predictive technology models [136], where nfin denotes number of fins in the FinFET [137].

the NMOS $N1$ as shown in Fig. 3.6(b). The inputs to the NOR gate are driven by identical inverters, who receive ideal step inputs. The NMOS $N1$ is sized to provide a charge current of $I_{c,\min}$, while satisfying $V_{DS} < 10\,\text{mV}$. The gate voltage of $N1$ gets raised to 600 mV, turning it ON in the linear region with overdrive voltage $\geq 350\,\text{mV}$. The NOR gate is sized so that the CMOS driver switches within 50 ps while driving $N1$ and the inverters are minimum sized. We simulate this schematic using 14 nm HP FinFET ASU predictive technology models [136] to estimate its switching delay and energy consumption.

3.4 Design of SCN-based Classifiers

3.4.1 Linear Support Vector Machine (SVM) Classifier

A linear SVM classifier can be realized using the proposed SCNs by connecting multiple SCNMs in parallel (Fig. 3.7(c)) and one stochastic slicer to generate the final classification decision. The SCNM and slicer symbols are defined in Fig. 3.7(a) and (b), respectively. Each multiplier generates spin current corresponding to the product $x_i w_i$, where x_i and w_i denote ith dimension of input feature vector $\mathbf{x}$ and the weight vector $\mathbf{w}$, respectively. Both x_i and w_i are fixed-point binary numbers. The spin currents at the output of SCNMs accumulate in a common channel (generating $I_{\text{s,o}}$) and feed

into the stochastic slicer.

The inputs are kept ON for the duration T_g since the stochastic slicer requires that much time to produce its decision. Noting that the stochastic slicer thresholding involves a comparison of its input spin current with I_{s,T_g}, Eq. (2.1) of SVM can be realized as follows:

$$I_{\mathrm{s,o}} = (\mathbf{w}^T\mathbf{x} + b)I_{\mathrm{s,lsb}} + I_{\mathrm{s,bias}} \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} I_{s,T_g} \tag{3.9}$$

where $I_{\mathrm{s,bias}}$ is additional bias current.

When $I_{\mathrm{s,o}} = I_{s,T_g}$, the slicer switches with probability 0.5. In SVM, this operating point would occur when the input feature vector $\mathbf{x}$ lies on the classifier hyperplane, i.e. when $\mathbf{w}^T\mathbf{x} + b = 0$, resulting in

$$(I_{\mathrm{s,o}})|_{\mathbf{w}^T\mathbf{x}+b=0} = I_{s,T_g} = I_{\mathrm{s,bias}} \tag{3.10}$$

To avoid large bias currents, we modify the feature vector $\mathbf{x}$ to $(\mathbf{x} + d\mathbf{1})$, where $\mathbf{1}$ denotes the all-one vector and d is a constant, thereby transforming Eq. (3.9) into

$$\left[\mathbf{w}^T(\mathbf{x} + d\mathbf{1})\right] I_{\mathrm{s,lsb}} + I_{\mathrm{s,bias}} \underset{\hat{y}=-1}{\overset{\hat{y}=1}{\gtrless}} I_{s,T_g} \tag{3.11}$$

where $I_{\mathrm{s,bias}}$ is now given by:

$$I_{\mathrm{s,bias}} = I_{s,T_g} + bI_{\mathrm{s,lsb}} - \left[d\Big(\sum_i w_i\Big)\right] I_{\mathrm{s,lsb}} \tag{3.12}$$

The bias current $I_{\mathrm{s,bias}}$ is generated by having an additional magnet with a supply current $I_{\mathrm{c,bias}}$ as shown in Fig. 3.7. If the signed precision of x_i is M bits, we choose d to be 2^{M-1}. This makes $(x_i + d)$ an unsigned number, removing the need for differential supply. The sign of $\mathbf{w}$ is accounted for by changing the magnetization vector directions of the corresponding spin-current injecting nanomagnets appropriately.

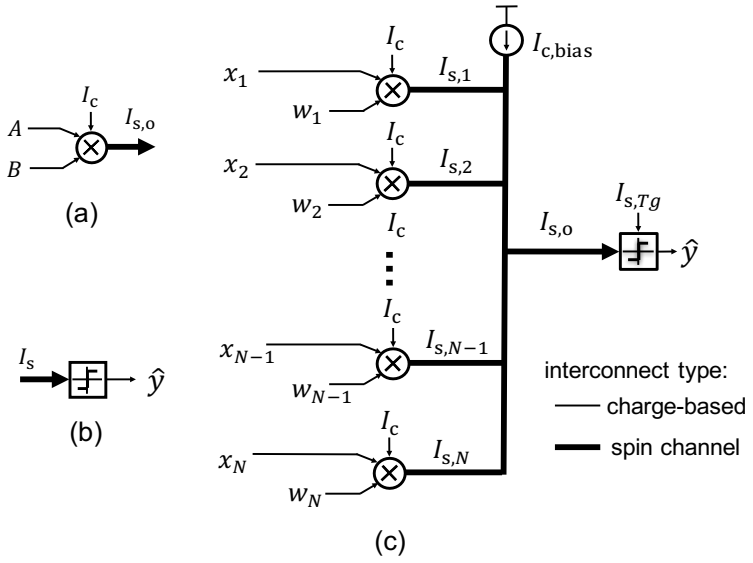


Figure 3.7: SCN-based linear SVM classifier: (a) SCNM symbol, (b) stochastic slicer symbol, and (c) the SCN-based N-dimensional linear SVM classifier architecture.

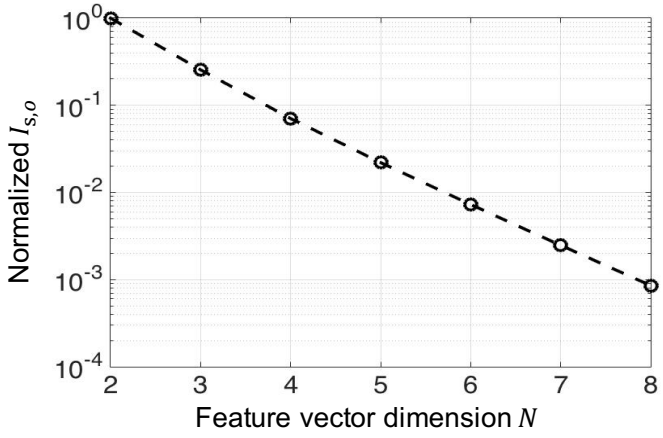


Figure 3.8: Normalized magnitude of $I_{s,o}$ in Fig. 3.7(c) as a function of classifier dimensionality N, when $x_1 = c_1 \neq 0$, $w_1 = c_2 \neq 0$, and $x_i = w_i = 0$ for all $i \in \{2, \ldots, N\}$, where c_1 and c_2 are some constants.

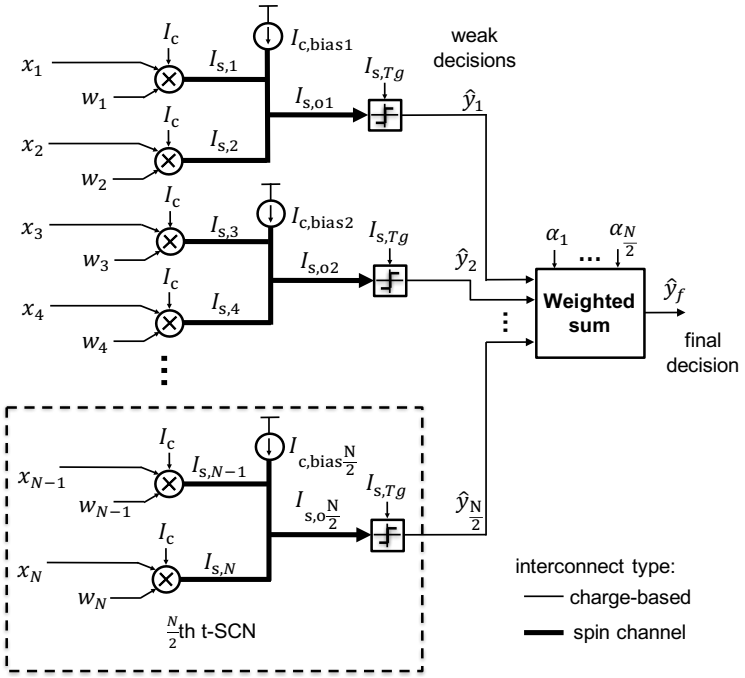


Figure 3.9: Boosted tiny SCNs architecture: adaptively boosted ensemble of $\frac{N}{2}$ tiny SCNs (t-SCNs), where each t-SCN consists of 2 SCNMs in parallel and one stochastic slicer to implement a two-dimensional linear SVM classifier.

3.4.2 Classifier Dimensionality Scaling via Boosted Tiny SCNs (t-SCNs)

The exponential decay of spin current exploited in SCNM makes it very hard to route the output spin current to another block as doing so inevitably incurs a significant loss in the spin current magnitude. This severely limits the ability to scale the classifier dimensionality (Fig. 3.7(c)), which requires the N multiplier output spin currents to be routed to a single stochastic slicer. Assuming a circular layout of N SCNMs with the slicer at its center, we estimate the loss in the spin current $I_{\mathrm{s,o}}$ magnitude as a function of classifier dimensionality N as shown in Fig. 3.8, when $x_1 = c_1 \neq 0$, $w_1 = c_2 \neq 0$, and $x_i = w_i = 0$ for all $i \in \{2, \ldots, N\}$, where c_1 and c_2 are some constants. Recall from Fig. 3.5 that the stochastic slicer requires minimum magnitude of the input spin current in order to operate accurately for a given switching delay. Thus, the exponential loss in the output spin current magnitude results in exponentially increasing classifier energy consumption in order to maintain classifier accuracy.

In order to address the problem, we limit the dimensionality of the SCN-based linear SVM classifier to only two dimensions, and refer to the resulting

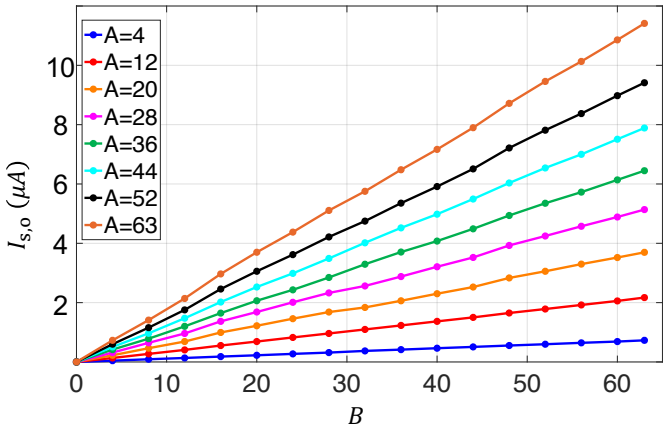


Figure 3.10: Simulated transfer function of a 6×6 bit SCNM realizing $A \times B$.

design as *tiny SCN* (t-SCN). We then employ AdaBoost to design an ensemble of multiple such t-SCNs to implement an arbitrary N-dimensional binary classification task. Figure 3.9 shows the boosted t-SCNs architecture. In particular, given an N-dimensional input feature vector, each t-SCN observes only two unique feature dimensions and computes its local decision $\hat{y}_i$. These local decisions could be inaccurate with higher probability, and hence are referred to as *weak decisions*. The final weighted sum block combines these weak decisions to obtain the final decision $\hat{y}_f$ as per Eq. (3.1). We restrict the number of weak classifiers to $\frac{N}{2}$ so that computational complexity of the boosted t-SCNs architecture is similar to the standard N-dimensional linear SVM implementation (Fig. 3.7(c)).

It is important to note that, in the boosted architecture, the output spin current of the channel network gets processed locally, and only the binary weak decisions are routed to the final weighted sum block, thus requiring much shorter spin interconnect routing within each t-SCN. It is straightforward to convert the binary slicer decisions $\hat{y}_i$ $i \in \{1, \ldots, \frac{N}{2}\}$ to equivalent voltage [138] and then route it using charge interconnects. We designed the linear combiner in Eq. (3.1) in conventional digital 14 nm CMOS. Its complexity in terms of the full adder count is less than 5% of the total complexity of the $\frac{N}{2}$ t-SCNs. One can also employ other schemes, such as Boolean logic, Winner-Take-All, to efficiently combine binary t-SCN outputs, achieving similar energy benefits.

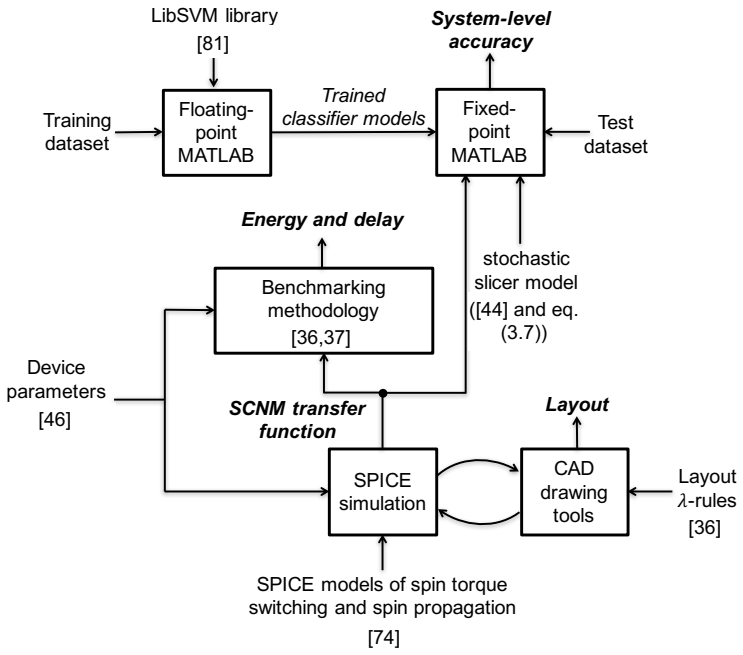


Figure 3.11: Simulation methodology.

3.5 Simulation Results

3.5.1 Simulation Methodology

Our simulation methodology is shown in Fig. 3.11. We first develop the SCNM layout and obtain its transfer function via SPICE-based simulations. We employ SPICE-based spin device models [133] for SCNM schematic simulations and use CAD drawing tools for corresponding layouts and λ-rule checks. We assume the material parameters provided in Table 3.1. We design and characterize a 6×6 bit SCNM using the SPICE-based spin device models [133]. The channel lengths between the clusters in the SCNM layouts are repeatedly adjusted until the appropriate spin current weighting is achieved and all λ-rules are satisfied. Figure 3.10 shows the interpolated SCNM transfer function after carrying out detailed spice simulations for 289 different A and B values. The observed $(\frac{\sigma}{\mu})$ of the deviations from linearity in output spin current is 2%. Such good linearity was achieved, in part, because process variations (such as line edge roughness etc.) and length quantization between two clusters are not yet accounted for. However, this does indicate that there is still room to budget such impact of more variations within in-

Table 3.1: The SPICE-based models developed in [133] were employed to simulate all spin channel network designs in this chapter. Hence, most of the device and material parameters of nanomangets and the copper channel are chosen as mentioned in [133]. Few device parameters that were specifically chosen for the designs in this chapter are given in this table.

Variable	**Value**
Saturation magnetization [103]	250×10^3 A/m
Effective internal anisotropic field [103]	16×10^4 A/m
Damping coefficient[103]	0.007
Nanomagnet dimensions	30 nm×30 nm×10 nm
Slicer dimensions	30 nm×80 nm×2 nm
Spin channel width	30 nm
Spin channel thickness	100 nm

herent tolerance of machine learning classifiers. Also, since all such variations within the layouts are static, one can employ retraining [64] to alleviate their impact. We leave this exploration for future work.

We use the SCNM transfer function and stochastic slicer model (based on Eq. (3.7) in the main text and analysis in [102]) for system-level classifier accuracy predictions. We estimate the classification error probability p_e of boosted t-SCN implementation in MATLAB. We use LIBSVM [139] to train all SVM classifiers. The classifier training always happens in floating-point precision in MATLAB. We then quantize the trained classifier model and test data to have 6 bit precision, which we found out to be sufficient. We carry out 6-fold cross validation over the available data to estimate the average accuracy of all classifier implementations.

We employ benchmarking methodology [97, 98] to estimate energy and delay for all classifier implementations compared in Sec. 3.5.2. For ASL implementations, we assume ASL device consisting of nanomagnets with improved anisotropy reported in [103]. We also assume ASL gates to be clocked using a MOSFET to avoid static power consumptions. For digital CMOS implementations, we use the delay vs. energy curve of 20 nm LV CMOS FO4 inverter reported in [103], and assume activity factor of 33%.

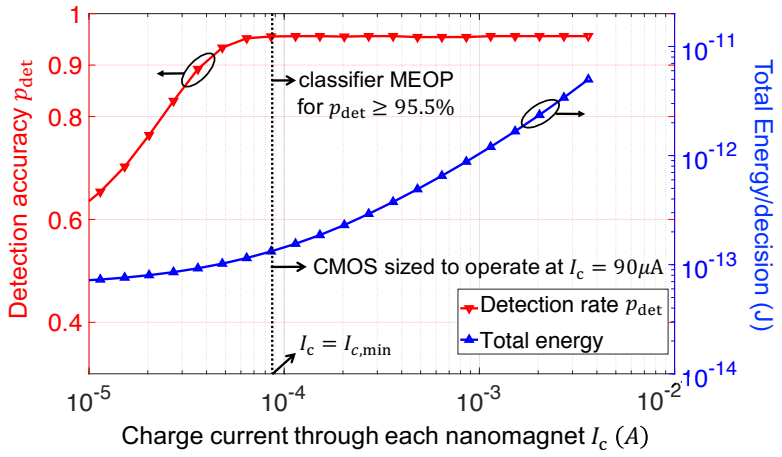


Figure 3.12: Accuracy p_{det} and total energy vs. I_c trade-off for 10-dimensional boosted t-SCN classifier operating at a final decision delay of 3 ns. The minimum energy operating point (MEOP) is achieved at $I_{c,\text{min}} = 90\,\mu\text{A}$.

3.5.2 Results

We demonstrate the effectiveness of proposed approach for two classification tasks: (1) 10-dimensional (10D) breast cancer detection (UCI repository dataset[140], [141]), and (2) 100-dimensional (100D) face detection (MIT CBCL dataset [142]). We quantify classification accuracy in terms of detection rate $p_{\text{det}} = 1 - p_e$, where p_e is classification error probability.

For each t-SCN, there exists a trade-off between NMOS current I_c and weak decision delay T_g for fixed p_{sw}. We choose $T_g = 2.5$ ns throughout this chapter to make sure that CMOS driver switching energy $\leq \approx 33\%$ of the total energy. We compare the energy consumption and accuracy of 10D and 100D classifier implementations at a fixed final decision delay of 3 ns and 4 ns, respectively. The remaining duration accounts for the delay of CMOS driver switching, slicer reset, and weighted logic block operation. In particular, the CMOS driver can be switched within 50 ps. For 10D classifier, weighted logic block operation can be approximated as a majority operation. We choose identical I_c for all weak classifiers. Given I_c and T_g, $I_{\text{c,bias}}$ is chosen according to Eq. (3.12) for each weak classifier.

For a fixed final decision delay (of 3 ns), the trade-off between the accuracy and total energy consumption of the 10D boosted t-SCN classifier is shown in Fig. 3.12 as a function of I_c. As expected, both accuracy and total energy decrease with I_c. The accuracy degradation occurs due to reductions in p_{sw}s

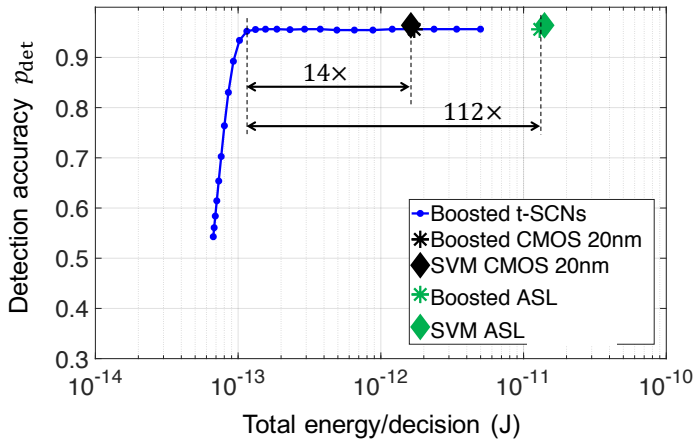


Figure 3.13: Classification accuracy p_{det} vs. energy trade-off for different 10-dimensional classifier implementations operating at a final decision delay of 3 ns.

of the stochastic slicers. For accuracy of 95.5%, the classifier MEOP (defined in Sec. 3.3.4) is achieved at $I_{c,\min} = 90\,\mu$A. Hence, we size the NMOS $N1$ to provide I_c of 90 μA at $V_{DD2} = 10$ mV (see Fig. 3.6).

Figure 3.13 shows the accuracy vs. energy trade-off for different 10D classifier implementations. Boosted t-SCN classifier achieves at least 112× lower energy per decision compared to that of the conventional boosted ASL implementation, while maintaining accuracy. Such large energy savings can be attributed to the elimination of all intermediate switching nanomagnets in the spin channel network implementation. It also achieves 14× lower energy compared to boosted 20 nm LV CMOS digital implementation, while operating at the identical final decision delay. We also observe that both boosted CMOS and boosted ASL implementations achieve energy consumption similar to the corresponding N-dimensional linear SVM implementations. For 100D classifier (Fig. 3.14), the energy benefits of boosted t-SCN implementation reduce to 2.5× and 22.5× over CMOS and ASL SVM implementations, respectively. This is primarily because of higher I_c requirements for its weak SVM classifiers, resulting from lower class separability of the dataset. Thus, the energy benefits are a function of the input data statistics as well. We only compare dynamic energy here, but leakage energy will be the least for SCN implementations due to having fewer transistors compared to both CMOS and ASL implementations. For all ASL implementations, we assume that

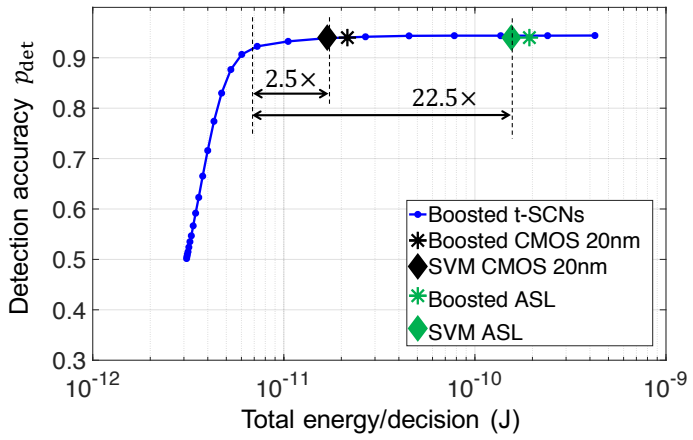


Figure 3.14: Classification accuracy p_{det} vs. energy trade-off for different 100-dimensional classifier implementations operating at a final decision delay of 4 ns.

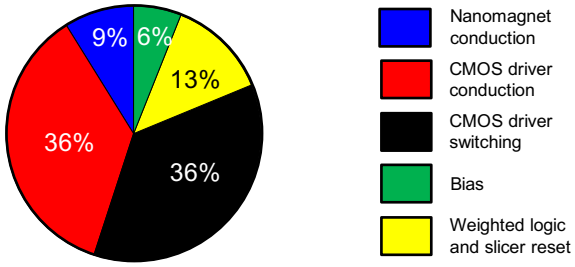


Figure 3.15: Category-wise energy breakdown for 10-dimensional boosted t-SCN implementation operating at a decision delay of 3 ns and with $I_{c,\min} = 90\,\mu$A.

the clocking transistors are shared across multiple nanomagnets [116], significantly amortizing their energy consumption.

In Fig. 3.15, we observe that the CMOS driver conduction energy and switching energy are comparable, and together dominate the energy consumption of the 10D boosted SCN classifier. The CMOS driver is expensive to switch due to large size of NMOS $N1$, which is necessary due to large charge current requirements ($I_{c,\min} \approx 100\,\mu$A) of SCN classifiers. Conduction energies of CMOS driver and nanomagnet add up to a constant $V_{DD2} I_{c,\min} T_g$. These trends are similar to ASL implementations.

3.6 Related Works

There have been few works that exploit the stochastic nature of nanomagnetic switching to achieve energy-efficiency. In [138], the switching behavior of magnets in super-paramagnetic regime was shown to resemble the dynamics of a Boltzmann machine, and thus a nanomagnetic network was trained to implement certain inference tasks. Stochastic magnets were also employed for energy-efficient random number generation [143] in stochastic computing, as well as for spike generation [144] in spiking neural network implementation.

Several research efforts have explored the neuromorphic design space using spin-devices in order to achieve energy-efficiency. While it is clear that the switching of nanomagnet naturally implements thresholding function of a neuron, these approaches use additional devices (resistive memory [145], domain wall magnets [146, 147, 148]) to achieve synaptic weighing of spin currents feeding into the nanomagnet. In [135], multiple binary weighted CMOS drivers along with a clever nanomagnet configuration were employed to obtain spin-current weighing in cellular neural network implementation.

There exists work at the architectural-level to fully exploit the advantages of emerging spin-device configurations such as racetrack memory [149, 150]. For example, high area-efficiency and serial access of racetrack memory was exploited to achieve reconfigurable precision [151] and efficient logic operations [152]. In [153], a novel data converter design was proposed by exploiting the serial structure of racetrack memory devices.

3.7 Discussion

In this chapter, we proposed spin channel networks where multiple input nanomagnets contribute to the spin current required to switch a single decision nanomagnet. These networks exploit diffusive spin current for efficient local computation to achieve very high energy-efficiency and ensemble of such isolated networks can solve any given classification task. Note that the exponential decay of the spin current and error-prone switching of the nanomagnets have been viewed as disadvantages in the digital all spin logic (ASL) implementations. This chapter demonstrates how the synergistic application of algorithmic techniques can turn apparent disadvantages of the emerging

devices into their strengths, and improve the system-level robustness vs. cost trade-off significantly in the process. For example, the proposed spin channel networks with their diffusive spin currents and unreliable nanomagnets achieved comparable accuracy at 10× lower energy per decision compared to conventional digital 20 nm CMOS implementations.

CHAPTER 4

EFFICIENT INFERENCE VIA MRAM-BASED DEEP IN-MEMORY ARCHITECTURE

4.1 Overview

As discussed in Sec. 1.2, the energy to access data from memory is significantly more than doing computations on that data [18]. This challenge is particularly exacerbated for deep net implementations, since they require millions of stored parameters to be accessed and computed upon. To address this concern, recently, *in-memory computing* (IMC) approach has emerged as one of the promising directions. In this approach, one strives to embed processing elements (PEs) in the periphery of memory bit-cell array (BCA) [108, 65], in order to directly readout a computed function of the stored data as a part of the memory access. For example, given a matrix $\mathbf{W}$ stored in BCA and input activation vector $\mathbf{x}$, IMC strives to compute output vector $\mathbf{a} = \mathbf{W}\mathbf{x}$ within a single BCA read access. This is particularly useful for efficient deep net inference implementations, since almost all memory intensive operations can be viewed as matrix vector multiplications (MVMs). However, they often need to be implemented in analog, due to severe area/pitch-matching constraints arising due to the high density nature of BCAs. Such embedded in-memory computation, while fast and energy-efficient, remains vulnerable to process variations [65]. Thus, in this approach, one needs to balance the energy/latency cost vs. robustness in order to maximize the benefits.

Much of the recent work in IMC focuses on SRAMs [108, 65, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 156]. Most notably, [108, 65] proposed an SRAM-based deep in-memory architecture (DIMA) to achieve a multi-bit vector dot product without requiring any modification in the standard 6T SRAM BCA. However, for state-of-the-art deep nets, on-chip SRAM may not be sufficient to store all the parameters, requiring highly energy and latency expensive off-chip DRAM accesses. Hence, it is necessary to ex-

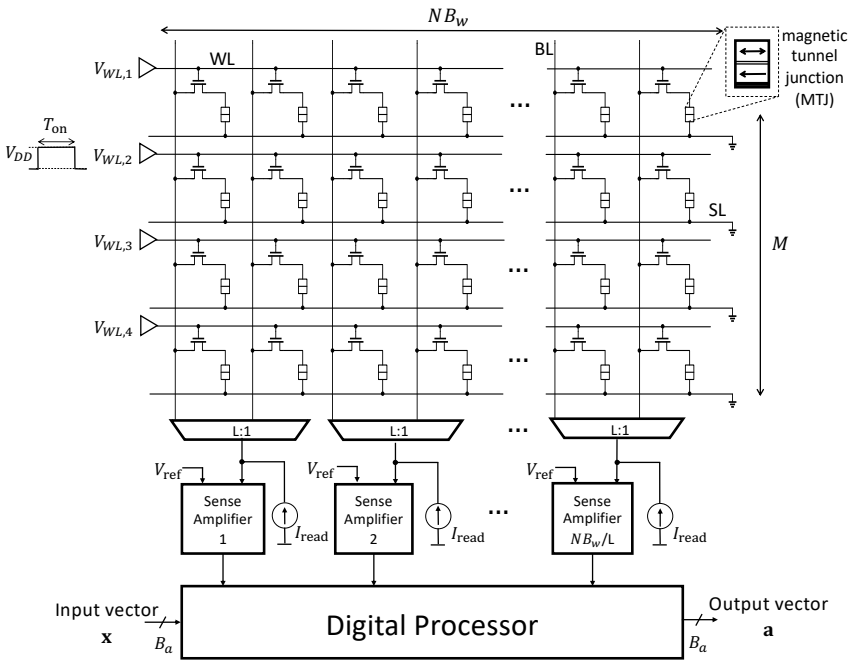


Figure 4.1: Digital implementation of matrix-vector multiplication (MVM) with weights stored in 1T-1MTJ MRAM bitcell array (BCA).

tend such an approach to emerging memories such as RRAM/PCM/MRAM, which are attractive alternatives due to their higher density and non-volatility. They seem particularly promising for hardware implementations of deep nets due to the possibility of having larger on-chip memory and potentially fewer memory write operations.

In the area of emerging memories, most of the recent works have focused on RRAM/PCM-based in-memory computing [166, 167, 168, 169, 170, 171, 8, 172] by exploiting multi-bit storage and/or large ON/OFF resistance ratio in RRAM/PCM bitcells. However, MRAM-based in-memory computing remains unexplored today even though MRAM has started to become available as a standard digital memory as a part of commercial process development kits [173]. This is, in part, because low ON/OFF resistance ratio and the absence of multi-state bitcells in MRAM makes it challenging to achieve multi-bit dot product computation within MRAM BCA.

In this chapter, we explore MRAM-based DIMA (MRAM-DIMA) to achieve multi-bit matrix-vector multiplication (MVM) within a single read operation. We employ a standard MRAM-BCA, without any modifications, to preserve its density. We propose modified peripheral circuits to achieve such multi-

bit computation, even though, unlike RRAM/PCM, each bitcell stores only 1 bit. The proposed MRAM-DIMA is shown to potentially achieve 10× and 20× lower energy and delay, respectively, compared to digital MRAM implementations with the matrix stored in an identical MRAM array.

4.2 Preliminaries

4.2.1 Notation

In this chapter, we assume the following MVM computation needs to be executed:

$$\mathbf{a} = \mathbf{W}\mathbf{x} \tag{4.1}$$

where $\mathbf{W}$ denotes a $M \times N$ weight matrix, and $\mathbf{x}$ and $\mathbf{a}$ denote the input and output vectors, respectively. Each weight is denoted as w_{ij} $\forall$ $i \in \{1, \dots M\}, j \in \{1, \dots N\}$ and is quantized to B_w bits. Each element of the vectors $\mathbf{x}$ and $\mathbf{a}$, denoted as x_i and a_i, respectively, is quantized to B_a bits.

4.2.2 Digital MRAM Architecture

Figure 4.1 shows a diagram of a conventional digital MRAM architecture with a BCA of size $N_{\text{row}} \times N_{\text{col}} = M \times NB_w$ which stores the weight matrix $\mathbf{W}$. This BCA has sourcelines (SLs) and wordlines (WLs) perpendicular to the bitlines (BLs). The MRAM bitcell consists of an NMOS access transistor and a magnetic tunnel junction (MTJ) (1T-1MTJ). The stored bit is read by sensing the resistance across the MTJ, which is low when the two magnetization vectors within the MTJ are parallel to each other (P state), and is high when they are antiparallel (AP state). In this chapter, bit value 1 (0) corresponds to the P (AP) state. We denote total bitcell resistance (conductance) by R_{P} (G_{P}) when MTJ is the P state, and by R_{AP} (G_{AP}) when it is in the AP state.

During an MRAM read operation, a constant current of I_{read} is passed through the bitcells and the voltage developed on the BLs is sensed to determine the MTJ state. Access transistors of a single row are activated for

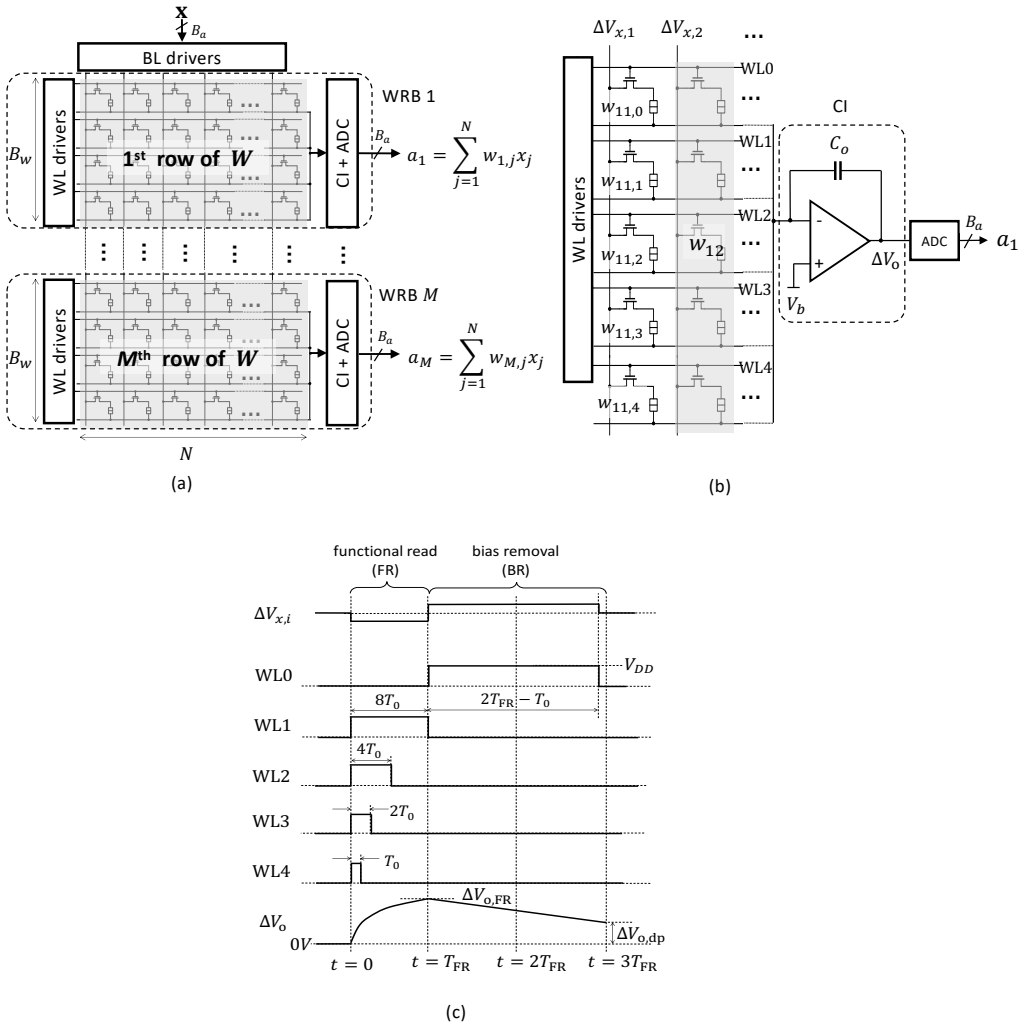


Figure 4.2: The proposed MRAM-based deep in-memory architecture (MRAM-DIMA): (a) the overall architecture consisting of M parallel word-row blocks (WRBs) each sharing N BL drivers but possessing separate WL drivers and current integrators (CIs), (b) schematic of a WRB, and (c) the read access timing diagram.

T_{on} duration by driving its WL to V_{DD}. A sense amplifier (SA) converts the voltage on the BL to a bit decision. Since the SA is typically wider than a bitcell, it is shared across multiple columns via $L:1$ multiplexers (typical value of L is from 8 to 32). Thus, $\frac{N}{L}$ weights are read per cycle, which are then input to a digital processor consisting of $\frac{N}{L}$ parallel multipliers followed by an adder tree for computing a dot product.

The average energy required by the MRAM-digital architecture to implement the MVM computation in Eq. (4.1) is given by:

$$E_{\text{digital}} = MNB_w \left[(I_{\text{read}} V_{DD} T_{\text{on}} + E_{SA}) + LC_{\text{wl-cell}} V_{DD}^2 \right] + E_{\text{proc}} \tag{4.2}$$

where E_{SA} and E_{proc} denote the sense amplifier energy and the total digital processor energy, respectively, T_{on} is the ON time of a single row during BCA read operation, and $C_{\text{wl-cell}}$ denotes the WL capacitance per bitcell. Similarly, the delay to complete the MVM operation is given by:

$$T_{\text{digital}} = MLT_{\text{on}} + T_{\text{proc}} \tag{4.3}$$

where T_{proc} denotes the delay of the digital processor in Fig. 4.1.

4.3 Voltage Driven MRAM-based Deep In-memory Architecture

4.3.1 Overall Architecture

Figure 4.2(a) shows the MRAM-DIMA consists of a conventional MRAM BCA and peripheral blocks including WL drivers, BL drivers, and current integrators (CIs) on SLs. The weights are stored in the BCA in a column major format. A set of B_w rows of BCA storing one row of $\mathbf{W}$ constitutes a word-row block (WRB). Each WRB implements a vector dot product to compute one element of the output vector $\mathbf{a}$ in Eq. (4.1). All WRBs operate in parallel, sharing the BLs, but possess separate WL drivers and SL CIs, as shown in Fig. 4.2(a).

4.3.2 MRAM-DIMA Operation

In this subsection, we describe the operation of a single WRB in detail. Figure 4.2(b) shows a schematic of WRB 1 with $B_w = 5$ and $B_a = 4$. Analog voltages $\Delta V_{x,i} = -x_i V_{\text{lsb}}$ are applied to the BLs via per column DACs (V_{lsb} denotes the DAC output resolution). Next, a functional read (FR) [108] step is initiated, in which the bottom $B_w - 1$ WLs are activated simultaneously by applying pulse-width modulated (PWM) access pulses with the bth WL turned ON for a duration of $2^{B_w-b-1}T_0$. Total FR phase duration is $T_{FR} = 2^{B_w-2}T_0$. All SL currents are summed in the CI, followed by an ADC to generate the digital outputs a_is. At the end of the FR phase, the resulting CI output voltage $\Delta V_{\text{o,FR}}$ is given by:

$$\Delta V_{\text{o,FR}} = \underbrace{\frac{T_{\text{o}} V_{\text{lsb}}}{C_o}\left[\Delta G \sum_{j=1}^{N} w_{ij}x_j\right]}_{=\Delta V_{\text{o,dp}}\ \text{(dot product)}} + \underbrace{\frac{T_{\text{o}} V_{\text{lsb}}}{C_o}\left[(2^{B_w-1}-1)\sum_{j=1}^{N} G_j x_j\right]}_{=\Delta V_{\text{o,bias}}\ \text{(bias)}} \tag{4.4}$$

where $\Delta G = G_P - G_{AP}$, $G_P = \frac{1}{R_P + R_{\text{mos}}}$, $G_{AP} = \frac{1}{R_{AP} + R_{\text{mos}}}$, C_o denotes the capacitor in the CI, and $G_j = G_{AP}$ $(= G_P)$, when $w_{ij} \geq 0$ $(w_{ij} < 0)$.

The bias term $\Delta V_{\text{o,bias}}$ in Eq. (4.4) is generated by the non-zero bitcell current when storing a zero-valued bit. This bias term is removed in bias removal (BR) step which discharges C_o appropriately by enabling only WL0 as shown in Fig. 4.2(c).

4.3.3 Scaled-up MRAM-DIMA

The current integrator in Fig. 4.2(b) has a finite output swing, i.e., $\Delta V_{\text{o}} \leq \Delta V_{\text{o,max}}$. This output swing limitation is exploited to naturally implement clipped ReLU activation function in DNNs. However, in order to achieve correct computation, it is also necessary that $\Delta V_{\text{o,FR}} < \Delta V_{\text{o,max}}$. This condition is challenging to meet as vector dimension N increases (see Eq. (4.4)), since $\Delta V_{\text{o,bias}}$ gets canceled only during the BR step.

To design MRAM-DIMA for large N, we propose swing budgeting via *phase multiplexed computation* within each WRB. We partition the $MB_w \times N$ BCA vertically into three sub-arrays, each consisting of $\frac{N}{3}$ columns as shown in Fig. 4.3(a). All three sub-arrays share their SLs and CIs. However, we

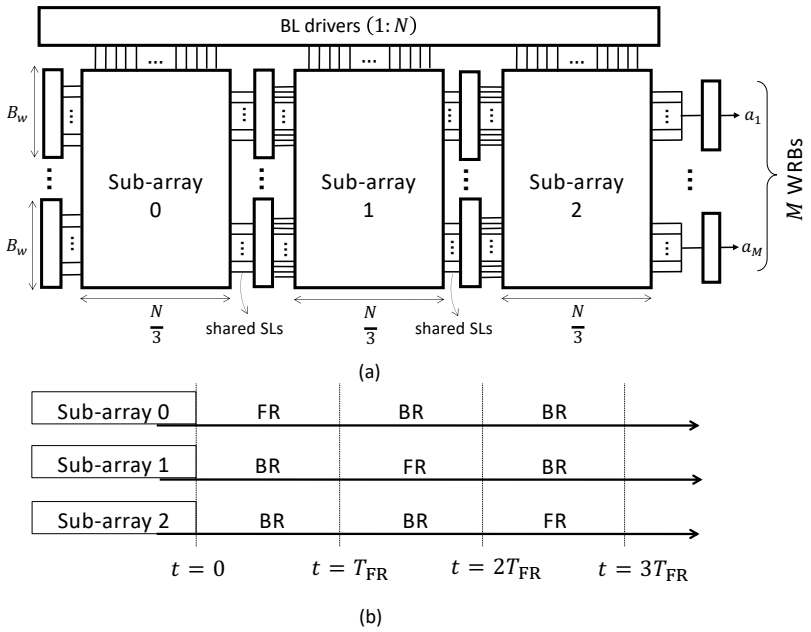


Figure 4.3: Realizing scaled-up MRAM-DIMA via phase multiplexed computations: (a) architecture, and (b) timing diagram.

introduce distinct WL drivers for individual sub-arrays to operate two in the BR phase, while the third in FR step as shown in Fig. 4.3(b). This approach drastically reduces the resulting $\Delta V_{\mathrm{o,FR}}$ at $t = T_{\mathrm{FR}}$ and $t = 2T_{\mathrm{FR}}$. For LeNet-300-100 DNN on MNIST dataset, it is estimated that $\Pr\{\Delta V_{\mathrm{o,FR}} > \Delta V_{\mathrm{o,max}}\} < 0.01\%$ at any given time instant using the behavioral models developed in Sec. 4.3.4, resulting in negligible system-level accuracy drop.

4.3.4 Modeling Energy, Delay, and MTJ Process Variations

The average energy consumption of MRAM-DIMA for implementing $M \times N$ MVM is:

$$E_{\mathrm{dima}} = MNB_w\left[\left(\frac{2^{B_w}-2}{B_w}\right)\bar{x}V_{\mathrm{lsb}}G_{\mathrm{cell}}V_{DD}T_0 + C_{\text{wl-cell}}V_{DD}^2\right] + ME_{\mathrm{adc}} + ME_{\mathrm{CI}} + NE_{\mathrm{dac}} \tag{4.5}$$

where $G_{\mathrm{cell}} = \frac{G_P+G_{AP}}{2}$, $\bar{x}$ denotes the average value of $\mathbf{x}$ across its N components, E_{adc}, E_{CI}, and E_{dac} denote the energy consumed in the ADC, CI, and DAC, respectively. The DAC is assumed to be equipped with an opamp-based voltage follower to drive the required current in BL [174]. Hence, E_{dac}

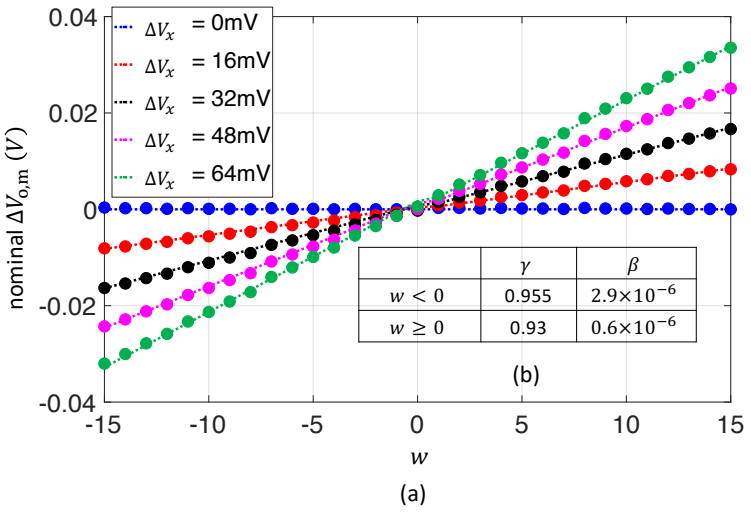


Figure 4.4: Transfer function of a scalar multiplier in WRB operating on two operands w and $x \propto \Delta V_x$ for $B_w = 5$ in a commercial 22 nm CMOS-MRAM process: (a) simulated (circles) and modeled (lines) and (b) estimated fitting parameters.

and E_{CI} are dominated by the respective opamp bias current energies. The value of $\bar{x}$ is estimated from the distribution of activations in DNNs. Compared to the digital MRAM energy consumption (in Eq. (4.2)), the BCA energy is reduced due to two factors: (1) the small value of $\bar{x}$, thanks to high activation sparsity in DNNs, caused by ReLU, and (2) the reduced read current per bitcell, since multiple bitcell currents are aggregated before ADC operation. The MRAM-DIMA delay is:

$$T_{\mathrm{dima}} = 3 \times 2^{B_w - 2} T_0 + T_{\mathrm{adc}} + T_{\mathrm{dac}} \tag{4.6}$$

where T_{adc}, T_{dac} denotes the delay of ADC and the DAC, respectively. The delay of MRAM-DIMA is nearly constant, resulting in the speed-up increasing with the number of rows M.

The expression in Eq. (4.4) for an analog vector dot product output $\Delta V_{\mathrm{o,dp}}$ ignores the impact of circuit non-idealities such as the body-effect of access transistors, virtual ground voltage bounce in the current integrator, and process variations in the MTJ. To account for these non-idealities, we model the analog output of a multiplier with a B_w bit operand w and an analog input

x in the WRB as follows:

$$\Delta V_{\mathrm{o,m}} = \frac{\gamma T_0 V_{\mathrm{lsb}}}{C_o}\big[\Delta G w + (2^{B_w-1}-1)\beta\big]x + \eta \tag{4.7}$$

where η denotes the spatial noise arising due to process variations in the MTJ. The fitting parameters β and γ are obtained via circuit simulations in a commercial 22 nm CMOS-MRAM process (see in Fig. 4.4).

Process variations at the multiplier output arises from R_{MTJ} variations across different bitcells. Hence, η in Eq. (4.7) follows a zero-mean Gaussian distribution with the variance σ_η^2 given by:

$$\sigma_\eta^2 = \left(\frac{T_0 V_{\mathrm{lsb}}}{C_0}x\right)^2 \left((2^{B_w-1}-1)^2 G_0^2 + \sum_{b=1}^{B_w-1} 4^{b-1} {G_b}^2\right)\left(\frac{\sigma}{\mu}\right)^2_{\text{G-bc}} \tag{4.8}$$

where $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ denotes σ-to-μ ratio of G_{cell}, and $G_b \in \{G_P, G_{AP}\}$ is the conductance of the b-th cell in a column, where $b \in \{0, \ldots, B_w - 1\}$. It is to be noted that both nominal $\Delta V_{\mathrm{o,m}}$ and σ_η scale linearly with T_0 and V_{lsb} keeping the signal-to-noise ratio unchanged. We estimate $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}} \leq 6\%$ via Monte Carlo circuit simulations for both P and AP states.

4.4 Simulation Results for Voltage Driven MRAM-DIMA

4.4.1 Design Choices

In this chapter, we choose $B_w = 5$ and $B_a = 4$, since these are typical DNN precision requirements for inference [175, 176]. For MRAM-DIMA, we assume a single-slope ADC architecture with a shared ramp generation circuit across the WRBs, typically employed in column-parallel ADCs in CMOS image sensors [177]. From the measured 8 bit, 7.1 MS/s 65 nm single-slope ADC results in [108], we conservatively estimate $E_{\mathrm{adc}} = 0.84$ pJ/sample and $T_{\mathrm{adc}} + T_{\mathrm{dac}} = 25$ ns for 4 bit precision, with T_{adc} dominating the delay. Furthermore, assuming $\Delta V_{\mathrm{o,max}} = \frac{V_{DD}}{3} = 300$ mV and $V_{\mathrm{lsb}} = 4$ mV, the system-level output swing requirements for mapping DNN computations described in Sec. 4.4.3 dictate $T_0 = 1$ ns, which is previously shown to be achievable

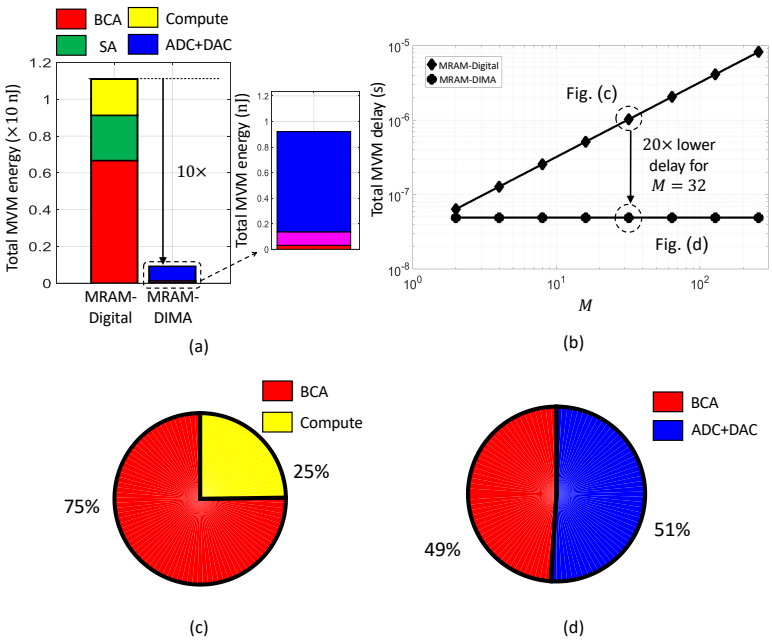


Figure 4.5: Simulation results for MVM operation: (a) energy breakdown, (b) delay as a function of M, (c) component-wise delay contributions in digital MRAM, (d) component-wise delay contributions in MRAM-DIMA for $M \times N = 32 \times 192$, $B_w = 5$, and $B_a = 4$.

[178, 179, 180] without significant energy overhead.

In the digital MRAM architecture, we use a BCA of identical capacity as in DIMA to store the weight matrix. Using the measured offset voltage variation in SA [181] and observed $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ in the PDK, we derive $I_{\text{read}} = 40\,\mu\text{A}$. Similarly, we assume $T_{\text{on}} = 3\,\text{ns}$, $L = 8$, and $E_{SA} = 40\,\text{fJ}$ from the results reported in [182, 181, 183, 184]. We obtain energy and delay of a full-adder via SPICE simulations and use it to estimate E_{proc} and T_{proc} in Eq. (4.2) and Eq. (4.3), respectively.

4.4.2 Energy and Throughput Benefits

In Fig. 4.5, we plot energy and delay models in Sec. 4.3.4 for $M \times N = 32 \times 192$ MVM computation corresponding to a 3×3 convolutional layer with 64 input and output channels. The MRAM-DIMA achieves a $10\times$ lower energy as shown in Fig. 4.5(a), primarily due to elimination of SA, and BCA energy reduction due to activation sparsity and reduced read current per bitcell. In MRAM-DIMA, E_{dac} and E_{CI} dominate due to their large opmap bias

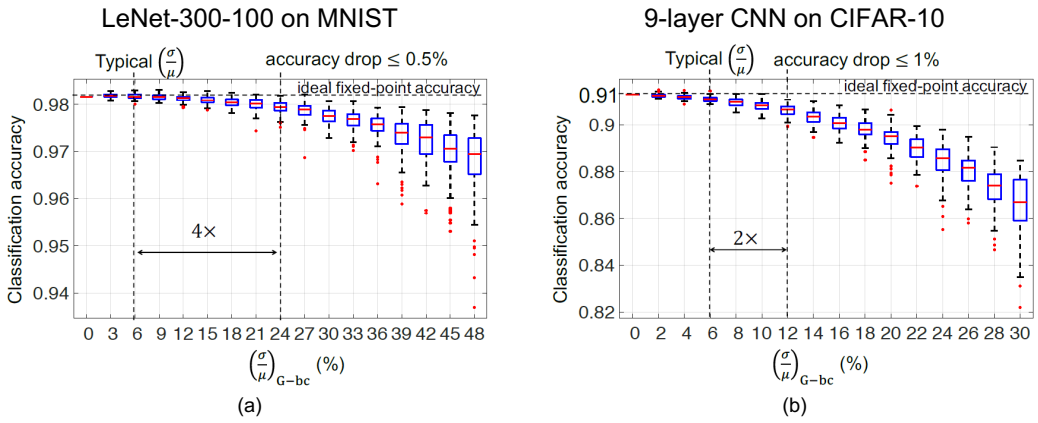


Figure 4.6: Classification accuracy vs $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ obtained using the models in Sec. 4.3.4 for: (a) LeNet-300-100 on MNIST dataset, and (b) nine-layer CNN on CIFAR-10 dataset. For each value of $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$, 100 instances of the BCA were generated to obtain the statistics shown in the error bars.

currents. The aggregated maximum current on SLs in a WRB is smaller than maximum current on single BL due to activation sparsity and phase multiplexed bias removal. This leads to smaller bias current energy in E_{CI} than that of the voltage follower in E_{dac}. The delay benefits of MRAM-DIMA over digital MRAM scale linearly with M due to the inherent parallelism in MRAM-DIMA (see Fig. 4.5(b)). A delay reduction of up to 20× is achieved for $M = 32$. While memory access delay dominates in digital MRAM, (see Fig. 4.5(c)), T_{adc} constitutes $\approx 50\%$ of T_{dima} (Fig. 4.5(d)).

4.4.3 Impact on System-level Accuracy

We estimate the impact of MRAM-DIMA's analog computation and process variations on the system-level accuracy of: (a) LeNet-300-100 MLP on MNIST dataset, and (b) a nine-layer CNN on CIFAR-10 dataset. These networks were trained using DoReFa-net [185] training methodology. The activations pass through a clipped ReLU activation function and are quantized between $[0, 1]$. The weights lie in the range between $[-\alpha_{ij}, +\alpha_{ij}]$, where α_{ij} is the scaling parameter for the i-th filter in the j-th layer, and they all lie in the range $[0.03, 0.15]$. Limitation on the voltage swing at the CI output ($\Delta V_{\text{o,max}}$) naturally implements clipped ReLU, while the α_{ij}s are implemented by appropriately scaling T_0. We emulate MRAM-DIMA's computation in PyTorch by using nominal behavioral models described in Sec. 4.3.4 for deterministic

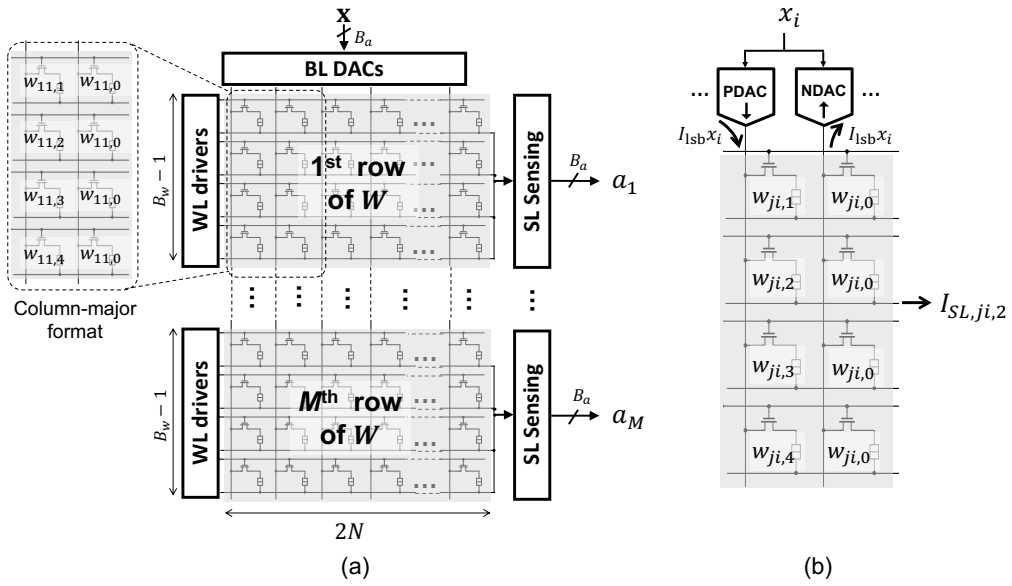


Figure 4.7: Current driven MRAM-DIMA design details: (a) column-major storage pattern of matrix **W** in BCA, and (b) current DAC driven BLs.

non-idealities and add bitwise random noise in bitcell conductance values to model MTJ variations. The MRAM-DIMA is able to tolerate a 4× and 2× higher $\left(\frac{\sigma}{\mu}\right)_{\text{G-bc}}$ than its typical value while maintaining $< 1\%$ drop in accuracy compared to ideal fixed-point computation for LeNet-300-100 and nine-layer CNN as shown in Fig. 4.6(a) and (b), respectively.

4.5 Current Driven MRAM-DIMA

In this section, we present a modified version of, where the BLs are driven by current DACs instead of the voltage DACs as discussed in Sec. 4.3. We discuss corresponding changes in the BCA storage and BL/SL peripherals.

4.5.1 BCA

As shown in Fig. 4.7(a), the weight matrix **W** is stored in the BCA in a column-major format, where $w_{ij,0} \ldots w_{ij,4}$ denote individual bits of 5 bit scalar elements w_{ij} represented in 1's complement format. Here $w_{ij,0}$ indicates the sign, while $w_{ij,1}$ is treated as the most significant bit (MSB). Thus, 5 bit scalar element w_{ij} occupies eight bitcells, with the sign bit $w_{ij,0}$ replicated

four times in the adjacent column. Again, a WRB is formed by combining four rows of BCA storing one row of $\mathbf{W}$ (similar to voltage driven design in Fig. 4.2(a). Furthermore, the pulse-width modulation for the WLs in each WRB is also same as that of the voltage driven design. However, the BR phase in voltage driven design is eliminated here by employing both P-type and N-type current DACs as discussed in the next subsection.

4.5.2 Current Driven BLs

BLs are driven by current DACs as shown in Fig. 4.7(b). This enables independent control of the current through each BL. Assuming ideal SL sensing, *i.e.* with zero input impedance looking into SL sensing circuit, the current through bitcell storing $w_{ji,2}$ can be written as:

$$I_{ji,2} = \frac{1}{G_{\mathrm{BL},i}}\Big[G_{\mathrm{AP}} + \Delta G w_{ji,2}\Big] x_i I_{\mathrm{lsb}} \tag{4.9}$$

where $G_{\mathrm{BL},i}$ denotes total conductance looking into the BL in ith column storing $w_{ji,2}$, $\Delta G = G_{\mathrm{P}} - G_{\mathrm{AP}}$, and I_{lsb} denotes the LSB current of BL DAC. Eq. (4.9) indicates that for achieving accurate dot product in single WRB, each BL needs to have identical conductance looking into it, *i.e.*, $G_{\mathrm{BL},i} = G_{\mathrm{BL}}$ $\forall i$. Note that this does not occur naturally, since each $G_{\mathrm{BL},i}$ is a function of data stored in that column. For now, we assume that BL conductances for all columns are equal to G_{BL}, and address BL conductance equalization in Sec. 4.5.4.

It is also to be noted that only the second term in the above Eq. (4.9) is desired for the dot product computation, while the first term is an additional bias that needs to be cancelled. It arises due to the fact that $G_{\mathrm{AP}} \neq 0$, *i.e.*, the bitcell contributes a non-zero current despite storing a 0 bit. This bias can be removed via an N-type current DAC connected to BL in the adjoining column storing replicated sign bit $w_{ji,0}$, as shown in Fig. 4.7(b). The resulting *unbiased* SL current contribution from bitcell pair $w_{ji,2}$ and $w_{ji,0}$ is given as:

$$I_{\mathrm{SL},ji,2} = \frac{\Delta G}{G_{\mathrm{BL}}}(w_{ji,2} - w_{ji,0}) x_i I_{\mathrm{lsb}} \tag{4.10}$$

It is to be noted that such bias removal via N-type current DAC also naturally enables signed computation, where resulting SL is negative for

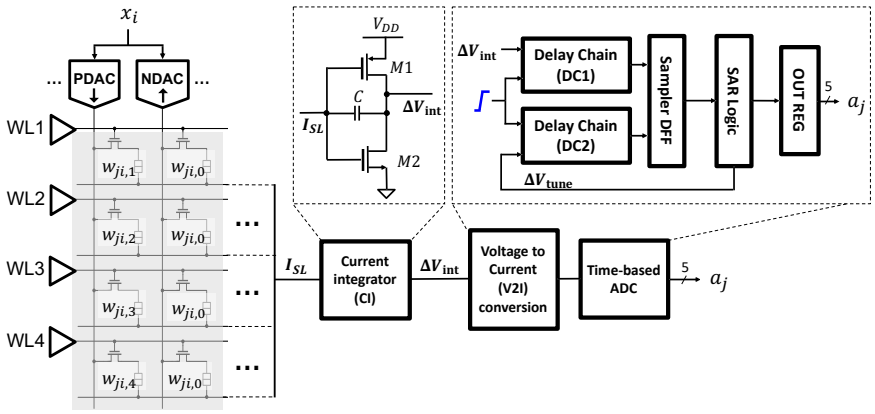


Figure 4.8: Block diagram of SL sensing circuits in MRAM-DIMA.

negative valued weight w_{ji}.

4.5.3 Time-based Sensing Circuits

Figure 4.8 shows a block diagram of sensing circuits at the SL. They consist of three stages, namely, current integrator (CI), voltage-to-time conversion, and time-based ADC. In CI, the incoming SL currents are integrated to develop voltage change ΔV_{int} at its output, which is proportional to the dot product computed in that WRB. The CI is implemented as an inverter with a capacitive feedback. This integrator output ΔV_{int} is converted to a digital code via a time-based ADC to produce 5 bit output a_j. In particular, it is applied to a delay chain DC1 (see Fig. 4.8) such that it proportionately changes the time it takes for the rising edge to pass through the delay chain. Thus, it is compared with a tunable reference voltage ΔV_{tune} by comparing the delays of DC1 and DC2 via sampling D-flipflop. Then, ΔV_{tune} is adjusted via standard successive approximation register and a current DAC, until ΔV_{tune} is within one LSB of ΔV_{int}. The resulting digital code corresponding to ΔV_{int} is read out via the OUT register.

The simplicity and component-efficiency of the proposed sensing circuit offers one crucial advantage – it enables pitch-matched layouts of this block, *i.e.*, the layout of each sensing circuit is pitch-matched to the height of each WRB, which equals to four times individual bitcell pitch. We discuss these layout constraints further in Sec. 4.6.

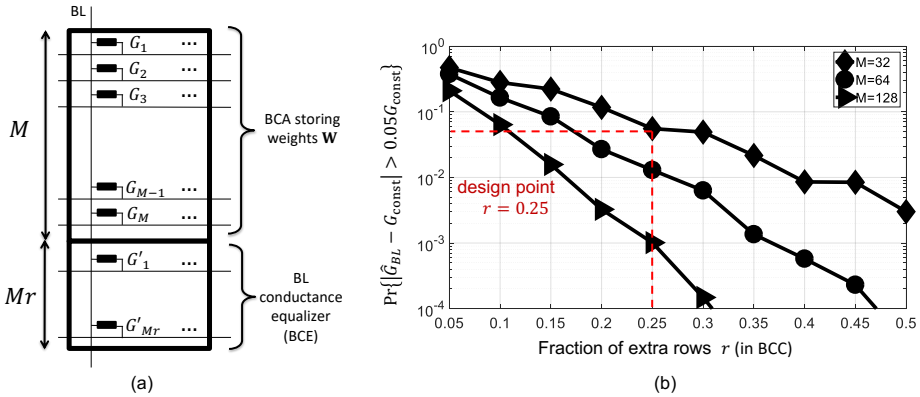


Figure 4.9: BL conductance equalization in MRAM-DIMA: (a) pictorial representation of BL conductance equalizer (BCE) with respect to BCA, and (b) probability of encountering a column whose conductance can not be compensated within $\pm 5\%$ of G_{const} as a function of the fraction of extra rows r constituting BCE.

4.5.4 BL Conductance Variations across Columns

When BLs are driven by current DACs, each bitcell current contribution gets scaled by total BL conductance in that column $G_{\text{BL},i}$ Eq. (4.9), deteriorating the accuracy of the dot product in each WRB. Hence, we propose to explicitly equalize the BL conductance across the by adding extra rows in the BCA, as shown in Fig. 4.9. We refer to these additional rows as *BL conductance equalizer (BCE)*, and denote the equalized conductance by $\hat{G}_{\text{BL}}$. Note that the BCE array is an identical BCA receiving the same WL pulses. The data the BCE is written such that total BL conductance (include BCA and BCE) is equal for each column. For example, in Fig. 4.9, the values of bitcell conductances $G'_1, \ldots, G'_{Mr}$ in BCE depend upon the bitcell conductances $G_1, \ldots, G_M$ in BCA column above them.

Let r denote a ratio of the numbers of rows in BCE to the number of rows in BCA. It is clear that the BL conductances are equalized perfectly across the columns, if $r = 1$. However, it also involves 100% overhead for such equalization. Hence, a key question is, can one achieve *good enough* equalization across the columns at much smaller values of r? This question can be answered rigorously as follows.

Let G_{const} denote the desired conductance after equalization for each column. One way to quantify the quality of equalization is by determining the probability (p_{bad}) of encountering a column whose conductance cannot be

Technology	GF 22-FDSOI
Die size	1.5mm X 1.5mm
Pins	72
V_{DD}	0.8 V
MRAM Capacity	5 kB (2 of 160×128)

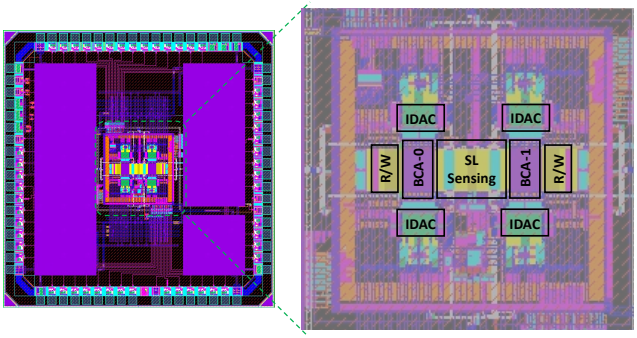


Figure 4.10: Layout and details of the chip prototype of current driven MRAM-DIMA.

compensated within 10% of G_{const}, despite having r rows in BCE, *i.e.*,

$$p_{\text{bad}} = \Pr\left\{|\hat{G}_{\text{BL}} - G_{\text{const}}| > 0.05 G_{\text{const}}\right\} \tag{4.11}$$

Under the assumption of Bernoulli data distribution, the relationship p_{bad} *vs*. r can be derived analytically for different values of BCA rows M, as shown in Fig. 4.9(b). Observe that, for a given p_{bad}, the required overhead r decreases, as expected, with increasing M. In this design, we have $M = 32$, and choose $r = 0.25$ to achieve $p_{\text{bad}} \approx 5\%$ as a sweet-spot between the accuracy *vs*. overhead of the conductance equalization.

4.6 Prototype Chip Tape-out

Figure 4.10 shows the details and the layout of our chip prototype of current driven MRAM-DIMA designed using GF-22 nm-FDSOI process development kit. The prototype was designed to implement 32×128 MVM computation. It consists of 160×256 MRAM crossbar which is columnwise split into two sub-arrays, namely BCA-0 and BCA-1. The SL sensing is located in the middle, reducing the impact of SL parasitic resistance. The standard digital MRAM read & write circuits (R/W blocks in Fig. 4.10) are located on the other sides of the two subarrays. Note that the SL sensing circuits are pitch-matched to the BCAs. This was challenging due to the high density of MRAMs. In particular, MRAM bitcells are so small that the pitch width of a *single* D flip-flop in the OUT Register (see Fig. 4.8) is comparable to the pitch of *eight* MRAM bitcells. Such tight pitch-matching constraints were met by

component-efficient time-based sensing circuits (see Sec. 4.5.3) and extensive manual layout efforts. At the end, SL sensing circuits corresponding to the consecutive WRBs are stacked sideways, and are pitch-matched to eight BCA rows.

During the prototype chip measurements, we found it challenging to achieve desired dot product accuracy due to significantly low signal-to-noise ratio (SNR) at the SL currents. In particular, we observed that the SL sensing circuits are overwhelmed by the sum of mismatch currents in all current DAC pairs, resulting in a large bias current at the SL that is not proportional to the dot product. This observation, while perplexing at first, motivated the detailed SNR analysis for IMC employing resistive crossbars presented in Chapter 5, where we analytically provide the reason behind this observation.

4.7 Discussion

In this chapter, we proposed an MRAM-based deep in-memory architecture to achieve matrix-vector multiplication within a single BCA read operation. It employed standard MRAM-BCA, without any modifications and modified peripheral circuits to achieve multi-bit computation even though each MRAM bitcell stores only 1 bit. We discussed both voltage-driven and current driven MRAM-DIMA. We showed that MRAM-DIMA can potentially achieve 10× and 20× lower energy and delay, respectively, compared to digital MRAM implementation with the matrix stored in an identical MRAM array.

While we designed a prototype chip implementing current-driven MRAM-DIMA, we observed prohibitively low SNR during measurements. This observation motivated a detailed SNR analysis of resistive crossbar IMCs presented in Chapter 5. Based on our analysis, we concluded that such amplification of the impact of input DAC mismatches does occur in current driven MRAM crossbars due to lower resistance states in MRAM and the larger input impedance ($\approx 2\,\mathrm{k}\Omega$) of sensing circuits. This being a fundamental device-level limitation, we recommend on-chip learning based error compensation techniques to improve the output SNR.

CHAPTER 5

SNR ANALYSIS OF IN-MEMORY COMPUTING EMPLOYING RESISTIVE CROSSBAR ARRAYS

5.1 Overview

In this chapter, we focus on understanding the compute SNR of IMCs employing emerging 1T-1R crossbar memories, such as RRAM, PCM, or MRAM. This SNR is significantly affected by the presence of process variations in the bitcell resistance as well as in the input digital-to-analog converters (DACs). While the recent works [166, 167, 168, 169, 170, 171, 8, 172] present individual design points, the understanding of how SNR is affected by the non-zero impedance of peripheral sensing circuits, the dimensionality of the dot product, and inherent ON/OFF resistance ratio for the bitcells remains incomplete. It is important to complete this understanding in order to identify the true capabilities of resistive IMC crossbars and develop design guidelines. This indeed is the goal of this chapter.

We conduct an SNR analysis of voltage driven resistive crossbars employed for in-memory MVM computations. The analysis is device-agnostic and is applicable to any resistive crossbar. Specifically, we use our analysis to comprehend the SNR trade-offs for PCM, MRAM, as well as RRAM. We consider the impact of device and circuit non-idealites such as the finite impedance R_s of the sensing circuits, and the mismatch in bitcells and input DACs. We also capture the impact of output analog-to-digital converters (ADCs) via accounting for clipping and quantization noise. In the case of voltage DAC (VDAC) driven crossbars, we show how SNR trades-off with R_s, the number of columns N in the crossbar, as well as the normalized energy per operation. Finally, we also consider current DAC driven crossbars and identify their challenges compared to voltage driven crossbars.

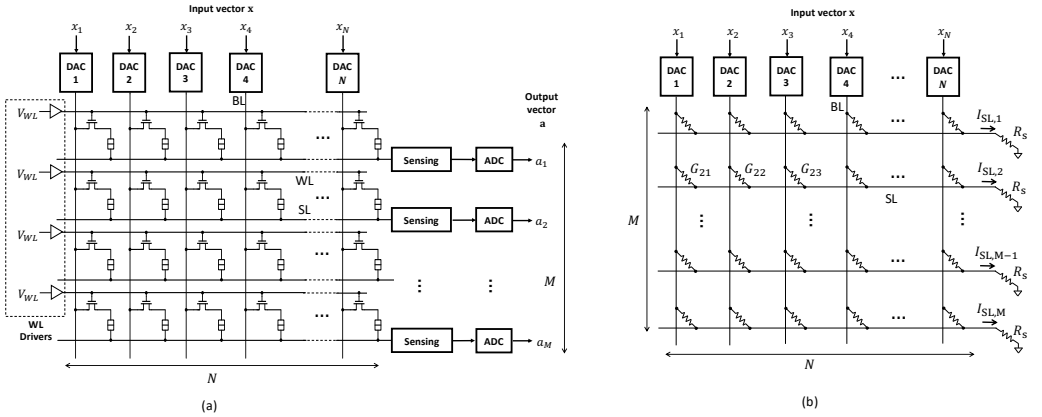


Figure 5.1: IMC with resistive non-volatile memories: (a) block diagram of a typical architecture, and (b) its abstraction for steady-state SNR analysis presented in this chapter.

5.2 Analysis Setup

Figure 5.1(a) shows a block diagram of a canonical crossbar architecture employing an $M \times N$ resistive BCA and associated peripherals required for MVM computation. Each bitcell in the BCA consists of an access transistor controlled via horizontal wordlines (WL) and a resistive memory device whose conductance is proportional to the value of data stored in the bitcell. Note that MRAM devices can have only two distinct conductance states, storing single bit per bitcell. However, RRAM/PCM devices can potentially store multiple bits within single bitcell via multiple distinct conductance states.

The input vector $\mathbf{x}$ is applied to the bitlines (BLs) via current/voltage DACs (see Fig. 5.1(a)), while the BCA stores the matrix $\mathbf{W}$ with every element stored in the corresponding bitcell. The elements of the output vector $\mathbf{a} = \mathbf{W}\mathbf{x}$ needs to be estimated from the row-wise horizontal sourceline (SL) currents via SL sensing circuits followed by analog-to-digital converters (ADCs). Notice that during the IMC operation all WLs are turned ON. Hence, for the SNR analysis in this chapter, we make the following two abstractions (see Fig. 5.1(b)): (i) the BCA is represented as a purely resistive crossbar array, where each resistance value includes both the ON resistance of the access transistor and the resistance of the memory device, and (ii) we represent the SL sensing circuits as a resistance R_s, which represents the Thevenin equivalent resistance looking into the input of the SL sensing circuit. We account for the impact of the ADC, such as quantization noise

and clipping noise, by referring those to the input of the SL sensing circuits.

We denote the resistance of a bitcell in ith row and jth column as $R_{ij} \in \{R_{\text{on}}, R_{\text{off}}\}$ and corresponding conductance $G_{ij} = \frac{1}{R_{ij}} \in \{G_{\text{on}}, G_{\text{off}}\}$, where we assume each bitcell stores a single bit with two distinct resistance states: low resistance state R_{on} and high resistance state R_{off}, and G_{on} and G_{off} denote the corresponding conductance values. We consider following three memory device technologies since they cover a wide range of $R_{\text{off}}/R_{\text{on}}$ values:

- MRAM [186]: $R_{\text{on}} \approx 3\,\text{k}\Omega$ and $R_{\text{off}} \approx 6\,\text{k}\Omega \implies \frac{R_{\text{off}}}{R_{\text{on}}} \approx 2$
- RRAM [172]: $R_{\text{on}} \approx 25\,\text{k}\Omega$ and $R_{\text{off}} \approx 300\,\text{k}\Omega \implies \frac{R_{\text{off}}}{R_{\text{on}}} \approx 12$
- PCM [8]: $R_{\text{on}} \approx 40\,\text{k}\Omega$ and $R_{\text{off}} \approx 3\,\text{M}\Omega \implies \frac{R_{\text{off}}}{R_{\text{on}}} \approx 75$

where we define the ratio $R_{\text{off}}/R_{\text{on}}$ as the *resistive contrast* of the bitcell.

In a resistive crossbar, the signal of interest is the SL current $I_{\text{SL,i}}$ in ith row whose *ideal* value is given by $\left(\sum_{k=1}^{N} V_k G_{i,k}\right)$ where V_k denotes the voltage applied at the kth BL. We first derive an exact closed-form expression of the nominal value of $I_{\text{SL,i}}$ as a function of R_s and N. Then, we consider the impact of variations and noise on $I_{\text{SL,i}}$. Specifically, the noise SL current I_{SL} for each row is written as (dropping index i for simplicity):

$$I_{\text{SL}} = I_{\text{sig}} + I_{\text{bm}} + I_{\text{dm}} + I_{\text{cn}} + I_{\text{qn}} \tag{5.1}$$

where I_{sig} denotes nominal SL current in the absence of any variations, and the other terms are defined below:

- I_{bm}: impact of mismatch variations in the bitcell conductances. It is well known that R_{on} and R_{off} values in the crossbars vary by 4%-to-6%.
- I_{dm}: impact of mismatch variations in the input DACs. Due to the threshold voltage variations, the ON current of each DAC finger varies by $\sim 5\%$, resulting in an input-dependent variation in the total current of each current DAC. This phenomenon is also present in voltage DACs if they are constructed by cascading a current DAC and a trans-impedance amplifier (TIA). Such voltage DAC design is preferred due to the strict area constraints in the IMC crossbars.
- I_{cn}: noise originated due to the clipping of I_{SL} beyond range $-I_{\text{clip}}$ to I_{clip}. When SL sensing circuits convert I_{SL} to voltage before ADC,

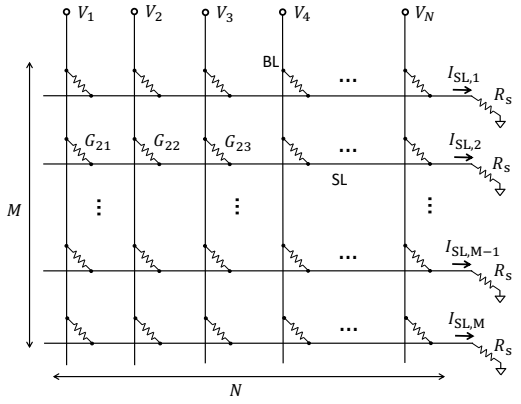


Figure 5.2: Voltage driven crossbar.

I_{clip} is dictated by the input voltage swing of the ADC. For advanced process technologies such as CMOS 22 nm, I_{clip} is typically in the range of a few μA for time-based successive approximation ADCs discussed in Sec. 4.5.3.

- I_{qn}: noise due to quantization via ADC. $I_{\text{qn}} \sim U\big(-\frac{I_{\text{clip}}}{2^{B_{\text{adc}}}}, \frac{I_{\text{clip}}}{2^{B_{\text{adc}}}}\big)$, where $U(L, H)$ denotes the uniform noise distribution between L and H, and B_{adc} denotes the ADC bit precision. For most resistive IMC crossbars with on-chip ADCs, B is typically chosen as 3-to-6 bits.

We define the SNR at the SL as:

$$\text{SNR} = \frac{\mathbb{E}\big[I_{\text{sig}}^2\big]}{\mathbb{E}\big[I_{\text{bm}}^2\big] + \mathbb{E}\big[I_{\text{dm}}^2\big] + \mathbb{E}\big[I_{\text{cn}}^2\big] + \mathbb{E}\big[I_{\text{qn}}^2\big]} \tag{5.2}$$

where the expectation $\mathbb{E}[\cdot]$ is taken over the joint distribution of data, variations, and noise.

5.3 Voltage Driven Crossbars

In a voltage driven crossbar (see Fig. 5.2), the jth BL is driven by voltage $V_{\text{BL},j} = V_{\text{DC}} + V_j$, where V_{DC} denotes the DC bias voltage, $V_j = x_j V_{\text{lsb}}$ input-dependent small-signal voltage, x_j denotes jth element of vector $\mathbf{x}$, and V_{lsb} denotes minimum DAC voltage step corresponding to least significant bit (LSB). For $I_{\text{SL,i}}$ to be proportional to the V_js, the DC voltage of SLs needs to be set to V_{DC}. This can be achieved by using the same TIA topology as that

of the voltage DACs in the SL sensing to convert resulting $I_{\text{SL,i}}$ into voltage. For the analysis in this chapter, we assume this to be the case. Furthermore, in practice, V_{DC} of each DAC and each SL also varies due to the threshold voltage variations in the constituent transistors of the corresponding TIAs. In this chapter, however, we do not include those variations in our analysis since they are input-independent and can be compensated by a one-time tuning of the TIAs per chip.

Notice that even if ith crossbar row stores a vector of all zeros, *i.e.*, when $G_{ij} = G_{\text{off}}, \quad j \in 1, \ldots, N$, the resulting current $I_{\text{SL,i}} \neq 0$ since $G_{\text{off}} \neq 0$. In order to remove any non-zero DC current when $G_{ij} = G_{\text{off}}$, it is a common practice to [171, 8]: (i) apply differential inputs on the BLs, *i.e.*, $V_{2k-1} = -V_{2k}, \quad k \in \{1, \ldots, \frac{N}{2}\}$, and (ii) employ two bitcells $(G_{i,2k-1}, G_{i,2k})$ to store a single signed bit as follows:

$$b_{i,2k} = \begin{cases} 1 & \text{if } G_{i,2k-1} > G_{i,2k} \\ 0 & \text{if } G_{i,2k-1} = G_{i,2k} \\ -1 & \text{if } G_{i,2k-1} < G_{i,2k} \end{cases} \tag{5.3}$$

where $b_{i,2k}$ denotes the value stored in the bitcell pair $(G_{i,2k-1}, G_{i,2k})$. Note that with this data storage scheme, the dimension of the computed dot product in a single row is $\frac{N}{2}$. Also, this scheme can be extended to bitcells storing multi-bit information in a straightforward manner.

In the absence of any variations and noise, one can derive an expression for the nominal SL current in ith row by applying Kirchhoff's current law at the SL to obtain (see Appendix A for a detailed derivation):

$$I_{\text{sig},i} = \left[\frac{R_{\text{arr},i}}{R_{\text{arr},i} + R_s}\right]\left(\sum_{k=1}^{N/2} V_{2k}\Delta G_{i,2k}\right) = S_I I_{\text{ideal},i} \tag{5.4}$$

where $0 < S_I < 1$ (first term) is a *scale factor*, $I_{\text{ideal},i}$ (second term) denotes ideal SL current when $R_s = 0$, $\Delta G_{i,2k} = G_{i,2k-1} - G_{i,2k}$, and $R_{\text{arr},i}$ (*array resistance*) is the Thevenin resistance looking into the ith SL given by:

$$R_{\text{arr},i} = \frac{1}{\sum_{j=1}^{N} G_{i,j}} \tag{5.5}$$

In the typical data storage scheme (see Eq. (5.3)), almost half bitcells store

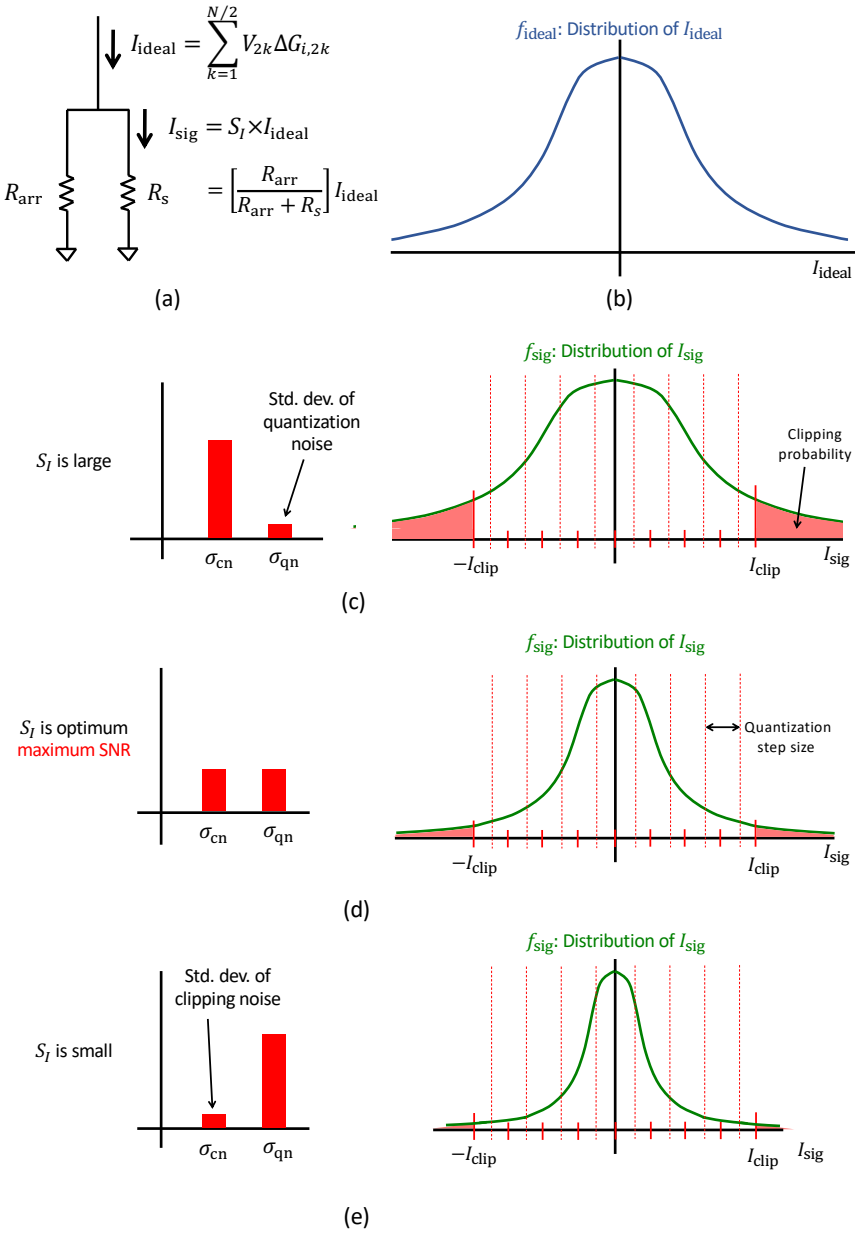


Figure 5.3: Interpreting Eq. (5.4): (a) equivalent circuit representation, (b) probability distribution of I_{ideal} and the probability distribution of I_{sig} when: (c) S_I is large ($\sigma_{\text{cn}} \gg \sigma_{\text{qn}}$), (d) S_I is optimum ($\sigma_{\text{cn}} \approx \sigma_{\text{qn}}$), and (e) S_I is small ($\sigma_{\text{cn}} \ll \sigma_{\text{qn}}$), where σ_{cn} and σ_{qn} denote the standard deviations of clipping and quantization noise. The vertical dotted lines in (c)-(e) denote ADC quantization bins reflected to the input of the SL sensing circuit.

G_{on} and remaining store G_{off} in every row. Hence, the array resistance can be approximated as:

$$R_{\text{arr},i} \approx \frac{1}{N\left(\frac{G_{\text{on}}+G_{\text{off}}}{2}\right)} = \frac{1}{NG_{\text{avg}}} = \frac{R_{\text{avg}}}{N} \tag{5.6}$$

where $G_{\text{avg}} = \frac{G_{\text{on}}+G_{\text{off}}}{2} = \frac{R_{\text{avg}}}{N}$ denotes the average of two bitcell conductance states, and its inverse R_{avg} is referred to as *average bitcell resistance*.

Substituting the value of $R_{\text{arr},i}$ from Eq. (5.6) in Eq. (5.4) and writing $\Delta G_{i,2k} = b_{i,2k}\frac{1}{R_{\text{off}}}\left(\frac{R_{\text{off}}}{R_{\text{on}}} - 1\right)$ and $V_{2k} = x_{2k}V_{\text{lsb}}$, we get,

$$I_{\text{sig},i} \approx \left[\frac{R_{\text{avg}}}{R_{\text{avg}} + NR_s}\right]\left(\frac{R_{\text{off}}}{R_{\text{on}}} - 1\right)\left(\sum_{k=1}^{N/2} b_{i,2k}x_{2k}\right)\left(\frac{V_{\text{lsb}}}{R_{\text{off}}}\right) \tag{5.7}$$

where the term $\frac{V_{\text{lsb}}}{R_{\text{off}}}$ is the minimum current through a bitcell. Notice that the third and fourth terms can be viewed as a numerical dot product scaled by the minimum bitcell current, while the first two terms are dimensionless gain factors. Naturally, in order to achieve high SNR, *both* of these factors need to be large. This shows that *ideally* bitcell resistance states should be such that their average $R_{\text{avg}} \gg NR_s$ and their contrast $\frac{R_{\text{off}}}{R_{\text{on}}} \gg 1$ in order to achieve high SNR.

Equation (5.4) can be interpreted (see Fig. 5.3(a)) as describing the process of current division of I_{ideal} across two resistors in parallel, viz. R_s and R_{arr}, where I_{sig} flows through R_s and $I_{\text{ideal}} - I_{\text{sig}}$ flows through R_{arr}. Since S_I linearly scales I_{sig}, its value controls the clipping vs. quantization noise trade-off. For example, a larger value of S_I results in higher clipping probability, and consequently a higher standard deviation of clipping noise σ_{nc} (see Fig. 5.3(b)). On the other hand, if S_I is too small, it squeezes the I_{sig} distribution close to zero, resulting in a higher standard deviation of quantization noise σ_{nq} (see Fig. 5.3(d)). Thus, there exists an optimum value of S_I when the clipping and quantization noise are balanced, *i.e.* when $\sigma_{\text{nc}} \approx \sigma_{\text{nq}}$ (see Fig. 5.3(c)), which maximizes the SNR.

Notice that the value of S_I in Eq. (5.4) is dictated by both R_s and R_{arr} as follows:

- A smaller value of $\frac{R_s}{R_{\text{arr},i}}$ results in a higher S_I. For example, when $R_s \to 0$, then $S_I \to 1$ and $I_{\text{sig},i} \approx I_{\text{ideal},i}$. However, achieving such small

Table 5.1: Typical simulation parameter values

Parameter	Value
VDAC bit precision (B_x)	5 bit
ADC bit precision (B_{adc})	6 bit
V_{lsb}	3 mV
I_{clip}	2 μA
$\left(\frac{\sigma}{\mu}\right)_{\text{bc}}$	4%
$\left(\frac{\sigma}{\mu}\right)_{V_{\text{lsb}}}$	4%

value of R_s ($< 500\Omega$) can be very challenging due to significant area and energy overhead of the SL sensing circuits with low input resistance.

- Increasing the number of columns N decreases S_I since $R_{\text{arr},i} \propto \frac{1}{N}$. This limits the dot product dimension $\frac{N}{2}$ that can be implemented.
- Smaller resistive contrast in the bitcell requires a smaller value of R_s for maintaining S_I. Thus, we can expect MRAM crossbar requiring a smaller R_s to achieve maximum SNR compared to RRAM and PCM.

Now we account for the impact of mismatch variations in the input voltage DACs by substituting $V_{2k} \leftarrow V_{2k} + \delta V_{2k}$ in Eq. (5.4) and separating out the mismatch term to obtain

$$I_{\text{dm},i} = \left[\frac{R_{\text{arr},i}}{R_{\text{arr},i} + R_s}\right]\left(\sum_{k=1}^{N/2} \delta V_{2k} \Delta G_{i,2k}\right) = S_I\left(\sum_{k=1}^{N/2} \delta V_{2k} \Delta G_{i,2k}\right) \tag{5.8}$$

where δV_{2k} denotes variation in the $2k$-th DAC pair. Recall that we assume VDAC is designed using a current DAC followed by a TIA. Hence, its variations are dominated by the variations in the current DAC, *i.e.* $\delta V_{2k} \sim \mathcal{N}\left(0, 2V_i V_{\text{lsb}} \left(\frac{\sigma}{\mu}\right)^2_{V_{\text{lsb}}}\right)$, where $\left(\frac{\sigma}{\mu}\right)_{V_{\text{lsb}}}$ corresponds to the deviation-to-mean ratio of the mismatch variations in a single current DAC finger. Similarly, we account for the impact of variations in the bitcell conductances by substituting $\Delta G_{i,2k} \leftarrow \Delta G_{i,2k} + \delta G_{i,2k}$ in Eq. (5.4) and separating out the mismatch term to obtain

$$I_{\text{bm},i} = \left[\frac{R_{\text{arr},i}}{R_{\text{arr},i} + R_s}\right]\left(\sum_{k=1}^{N/2} V_{2k} \delta G_{i,2k}\right) = S_I\left(\sum_{k=1}^{N/2} V_{2k} \delta G_{i,2k}\right) \tag{5.9}$$

where $\delta G_{i,2k} \sim \mathcal{N}\left(0, \left(\frac{\sigma}{\mu}\right)^2_{\text{bc}} (G^2_{\text{off}} + G^2_{\text{on}})\right)$ denotes the variation in the dif-

ference in conductances of $2k$-th pair of bitcells, $\left(\frac{\sigma}{\mu}\right)_{\text{bc}}$ denotes standard deviation-to-mean ratio for bitcell conductance mismatch. Also, notice that these conductance variations cause variations in $R_{\text{arr},i}$. However, it only changes the range of $I_{\text{sig},i}$ independent of the input vector $\mathbf{x}$. Hence, we do not include it in our analysis in this chapter since it can be accounted for by a corresponding small change in the ADC dynamic range. Note in Eq. (5.8) and Eq. (5.9) that both $I_{\text{bm},i}$ and $I_{\text{dm},i}$ scale with S_I in the same manner as I_{sig}. Hence, the ratios $\frac{I_{\text{sig}}}{I_{\text{bm},i}}$ and $\frac{I_{\text{sig}}}{I_{\text{dm},i}}$ do not depend on S_I, and hence on R_s. The implications will become clear in Sec. 5.3.1 where we will see that two of the SNR regimes are primarily dictated by the other two noise, *i.e.* quantization and clipping noise.

Next, we employ the expressions in Eq. (5.8) and Eq. (5.9) to study the impact of R_s, N as well as the BCA energy per operation for all three memory types on the SNR. Note that the expectations in SNR are estimated empirically via Monte Carlo simulations. The typical values of parameters chosen for these simulations are given in Table 5.1. Also, we constrain $b_{i,2k} \in \{-1, 1\}$ in our simulations for simplicity.

5.3.1 SNR vs. R_s Trade-off

We begin with a study of SNR vs. R_s trade-off. Given the tight area constraints on the SL sensing circuits, achieving very low R_s could be prohibitively expensive in resistive IMC crossbars. Hence it is important to understand how SNR changes with R_s and be able to estimate the value of R_s which maximizes the SNR. Figures 5.4(a), (b), and (c) show SNR vs. R_s trade-off for VDAC driven crossbars consisting of PCM, RRAM, and MRAM memory devices, respectively. We make following observations from Fig. 5.4:

- For each value of N, SNR is maximized at a particular value of R_s (defined as R_s^*) for all three memory device technologies. Furthermore, we observe that R_s^* decreases with N. This is expected since R_{arr} decreases with N.

- Recall that I_{sig} decreases when R_s increases. Thus, when $R_s < R_s^*$, clipping noise dominates, while quantization noise dominates when $R_s > R_s^*$.

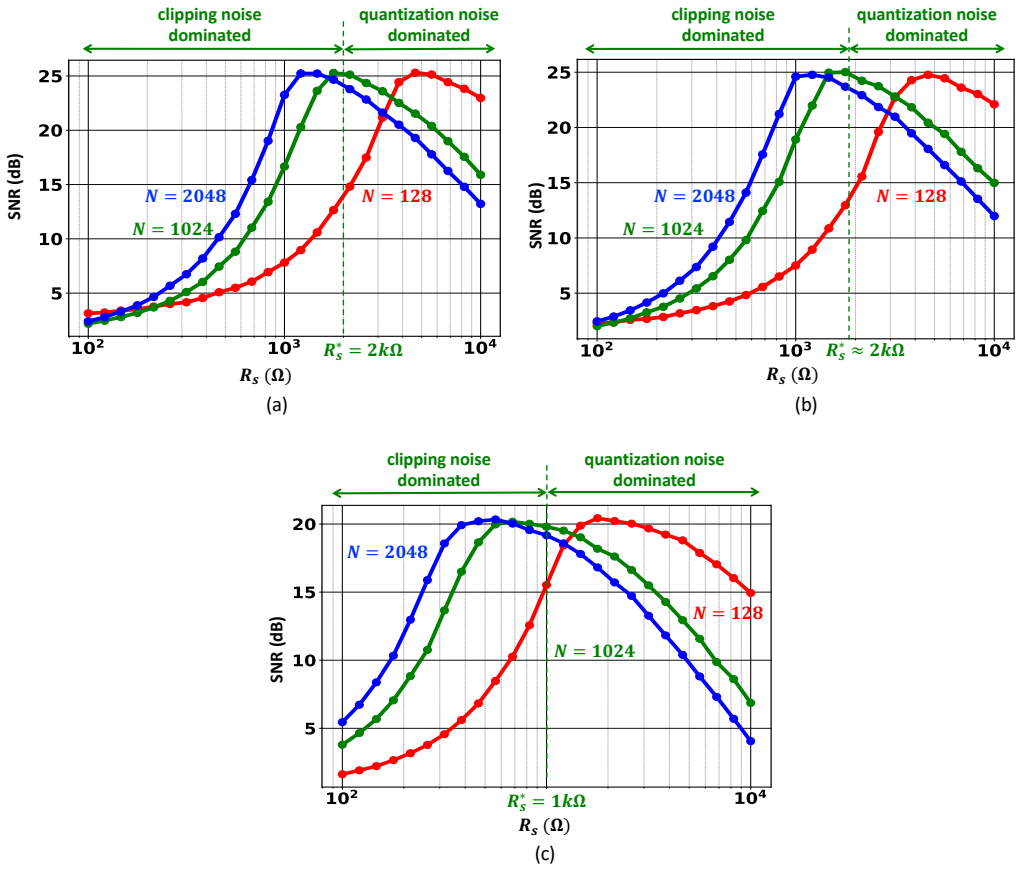


Figure 5.4: SNR vs. R_s trade-off in IMC with VDAC driven crossbars consisting of (a) PCM, (b) RRAM, and (c) MRAM memory devices.

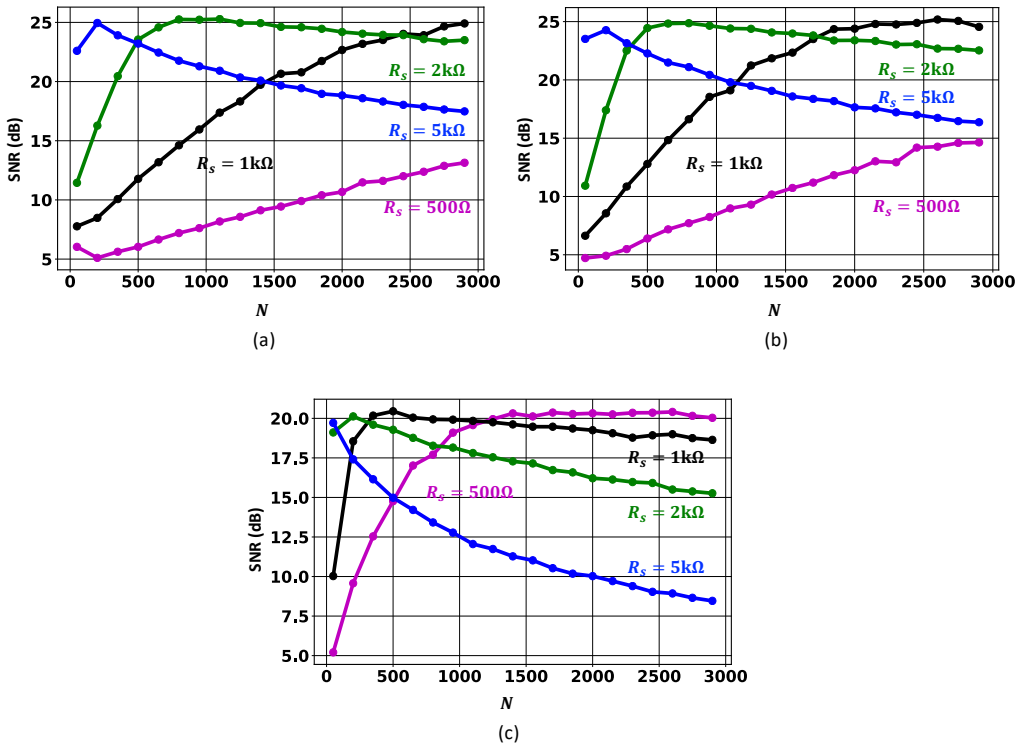


Figure 5.5: SNR vs. N trade-off in IMC with VDAC driven crossbars consisting of (a) PCM, (b) RRAM, and (c) MRAM memory devices.

- For a given value of N, R_s^* also depends weakly upon the resistive contrast of the memory device. For example, with $N = 1024$, R_s^* for MRAM is smaller than that of PCM and RRAM due to smaller resistive contrast of MRAM.

Notice that $R_s^* \approx 1\,\text{k}\Omega$-to-$2\,\text{k}\Omega$ for all three memory technologies, which is feasible even with the pitch-matching contraints of IMCs.

5.3.2 SNR vs. N Trade-off

Next, we now study the SNR vs. N trade-off for distinct values of R_s. Figures 5.5(a), (b), and (c) show the SNR vs. N trade-off for distinct values of R_s for VDAC driven crossbars consisting of PCM, RRAM, and MRAM memory devices, respectively. For RRAM and PCM, when R_s is too small (*e.g.* $R_s \leq 1\,\text{k}\Omega$), SNR increases with N since it remains dominated by clipping noise until $N = 3000$. For larger R_s values (*e.g.* $2\,\text{k}\Omega \leq R_s \leq 5\,\text{k}\Omega$), SNR first increases with N until $N = N^*$ and then decreases when $N > N^*$,

where it remains dominated by quantization noise. For example, with $R_s = R_s^* = 2\,\text{k}\Omega$, $N^* \approx 1000$ and $N^* \approx 750$ for PCM and RRAM, respectively. Also, notice that the decrease in SNR as a function of N in this case is very gradual for both PCM and RRAM.This is due to the opposing influence of two effects (see Eq. (5.4)): (i) $\mathcal{O}(\frac{1}{N})$ decrease in I_{sig} via R_{arr}, and (ii) $\mathcal{O}(\sqrt{N})$ increase in I_{sig} on average via $\sum_{k=1}^{N/2} V_{2k}\Delta G_{i,2k}$. Thus, SNR remains high for a broad range of N values.

Similar observations can be made for MRAM in Fig. 5.5(c). For instance, high SNR is achieved at a broad range of $500 \leq N \leq 1000$ with $R_s = R_s^* = 1\,\text{k}\Omega$. Also note that the maximum value of SNR for MRAM is only 20 dB, which is 5 dB smaller than that for PCM/RRAM. This is a direct consequence of MRAM's smaller resistive contrast.

5.3.3 Bound on M

Increasing M in IMC crossbars linearly improves throughput via parallelism. Naturally one would want to choose as high M as possible. Hence it is important to address following question: "How large can the number of rows (M) be in VDAC driven crossbars?". We address it in this subsection.

The value of M is limited by the constraint that $I_{\text{BL,max}} < I_{\text{VDAC,max}}$, where $I_{\text{VDAC,max}}$ denotes the maximum current that can be provided by VDAC while preserving its voltage. Based on Eq. (5.4), we first derive an expression for the maximum BL current $I_{\text{BL,m}}$ in a single column in a given crossbar with the sensing circuit resistance R_s by summing the maximum individual bitcell currents along the column to obtain:

$$I_{\text{BL,max}} = \sum_{i=1}^{M} \left[\frac{R_{\text{arr},i}}{R_{\text{arr},i} + R_s}\right] V_{\text{max}} G_{\text{on}} \tag{5.10}$$

where V_{max} maximum output voltage of VDAC. Note that a VDAC needs to provide the maximum current when all bitcell in its column are ON and its input corresponds to the V_{max}. For a B_x bit VDAC, $V_{\text{max}} = 2^{B_x} V_{\text{lsb}}$.

As observed in Eq. (5.10), $I_{\text{BL,max}}$ increases linearly with M. We plot $I_{\text{BL,max}}$ vs. M for $N = 200$ and $N = 1000$ in Fig. 5.6 for all three memory device technologies. For RRAM/PCM, we choose $R_s^* = 5\,\text{k}\Omega$ ($R_s^* = 2\,\text{k}\Omega$) corresponding to $N = 200$ ($N = 1000$) (recall from Fig. 5.5). Similarly, for

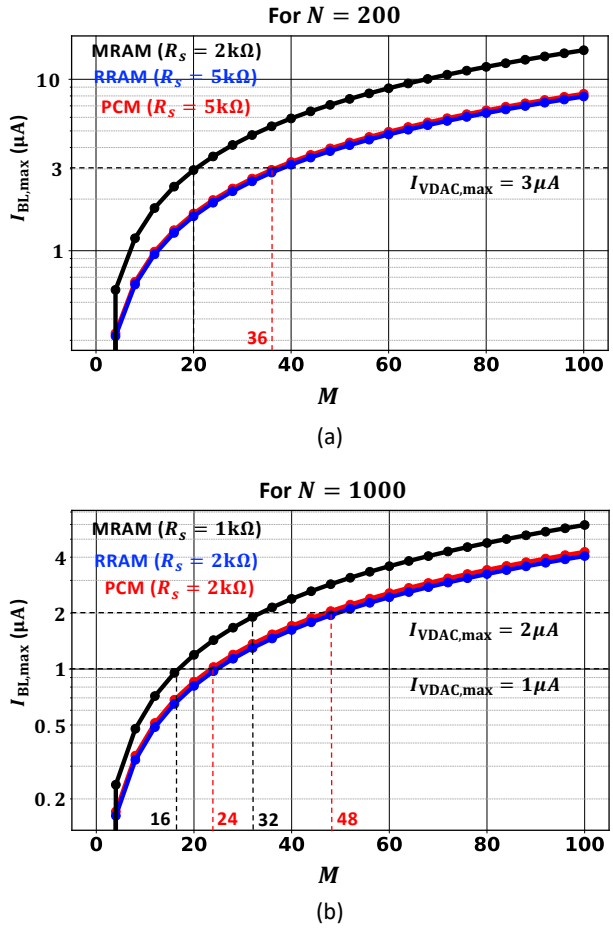


Figure 5.6: $I_{\mathrm{BL,max}}$ vs. M trade-off in IMC with voltage driven crossbars.

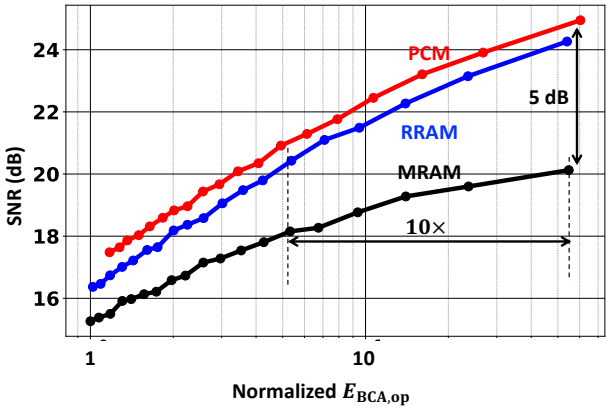


Figure 5.7: SNR vs. normalized BCA energy trade-off in IMC with VDAC driven crossbars consisting of PCM, RRAM, and MRAM memory devices.

MRAM, we choose $R_s^* = 2\,\mathrm{k}\Omega$ and $R_s^* = 1\,\mathrm{k}\Omega$ corresponding to $N = 200$ and $N = 1000$, respectively.

Note that given the stringent area constraints in the IMC crossbars, $I_{\mathrm{VDAC,max}}$ of a couple of μA is feasible to achieve. With $I_{\mathrm{VDAC,max}} = 2\,\mu\mathrm{A}$ ($I_{\mathrm{VDAC,max}} = 1\,\mu\mathrm{A}$), we find that $M = 48$ ($M = 24$) satisfies the constraint for RRAM/PCM with $N = 1000$ (see Fig. 5.6(b)). For MRAM, corresponding M values are reduced to 32 and 16, respectively. For smaller value of $N = 200$, we find that $I_{\mathrm{BL,max}}$ is higher for the similar range of M values for all three memory device technologies. Specifically, with larger $I_{\mathrm{VDAC,max}} = 3\,\mu\mathrm{A}$, we get $M = 36$ and $M = 20$ for RRAM/PCM and MRAM, respectively (see Fig. 5.6(a)).

5.3.4 SNR vs. Energy Trade-off

IMCs have a fundamental trade-off between the SNR and energy consumption, which determines their applicability in different tasks [65]. Figure 5.7 plots SNR vs. $E_{\mathrm{BCA,op}}$ trade-off for PCM, RRAM, and MRAM, where $E_{\mathrm{BCA,op}}$ denotes the BCA energy per $1\,\mathrm{bit}\times 1\,\mathrm{bit}$ MAC operation consumed during steady-state IMC operation. We estimate $E_{\mathrm{BCA,op}}$ as

$$E_{\mathrm{BCA,op}} = \frac{1}{N}\frac{1}{B_x} V_{\mathrm{DD}} T_{\mathrm{BCA}} \mathbb{E}\big[|I_{\mathrm{sig}}|\big] \tag{5.11}$$

where V_{DD}, T_{BCA} denote supply voltage and the ON duration of BL DACs, respectively, and $|\cdot|$ denotes the absolute value operation. We assume V_{DD}, T_{BCA} to be same across the three memory types: RRAM, PCM, and MRAM.

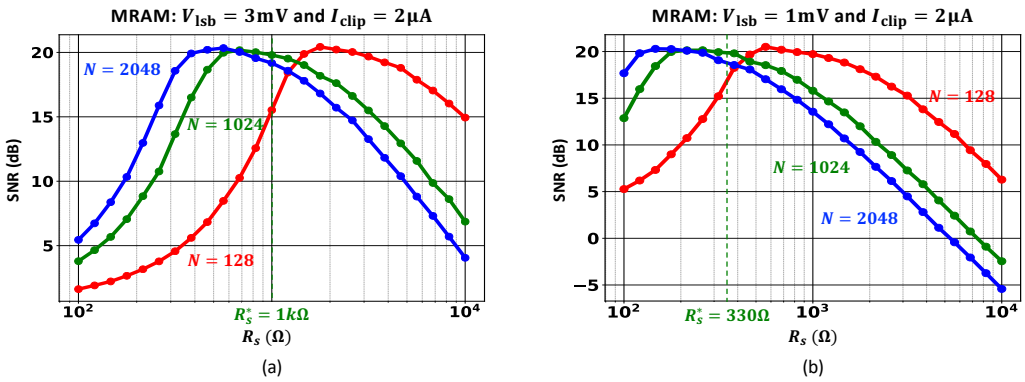


Figure 5.8: SNR vs. R_s trade-off in IMC MRAM crossbars with $I_{\text{clip}} = 2\,\mu\text{A}$ and: (a) $V_{\text{lsb}} = 3\,\text{mV}$, and (b) $V_{\text{lsb}} = 1\,\text{mV}$.

In Fig. 5.7, N is swept to obtain different points in the plot and $E_{\text{BCA,op}}$ values are normalized with respect to the smallest $E_{\text{BCA,op}}$ for MRAM (corresponding to $N = 1950$). For comparable $E_{\text{BCA,op}}$, SNR for MRAM crossbar is 5 dB smaller than that for RRAM/PCM crossbars. As we also discussed in Sec. 5.3.2, this is a direct consequence of smaller resistive contrast in MRAM. Figure 5.7 also quantifies the energy penalty of small resistive contrast. Specifically, RRAM and PCM achieve $\approx 10\times$ smaller $E_{\text{BCA,op}}$ compared to MRAM for comparable SNR of 20 dB. The larger separation between R_{on} and R_{off} values in RRAM/PCM enables their operation at smaller bitcell currents compared to MRAM while achieving similar computation accuracy.

5.3.5 Impact of V_{lsb} and I_{clip}

In earlier sections (from Sec. 5.3.1 to Sec. 5.3.4), we assumed $V_{\text{lsb}} = 3\,\text{mV}$ and $I_{\text{clip}} = 2\,\mu\text{A}$. While they can be reasonably realized in advanced process technologies, in this section, we discuss how the SNR trade-off will change w. r. t. both V_{lsb} and I_{clip}. We focus particularly on MRAM in this section since it turns out to be the most challenging due to its lower resistive contrast. However, similar conclusions can also be drawn for RRAM and PCM.

Figure 5.8 shows the impact of reducing V_{lsb} on SNR vs. R_s trade-off. Specifically, we find that a reduction in V_{lsb} results in a proportionate reduction in R_s^*. For example, in Fig. 5.8, changing V_{lsb} from 3 mV to 1 mV results in a proportionate $3\times$ reduction in R_s^*. This is expected since V_{lsb}

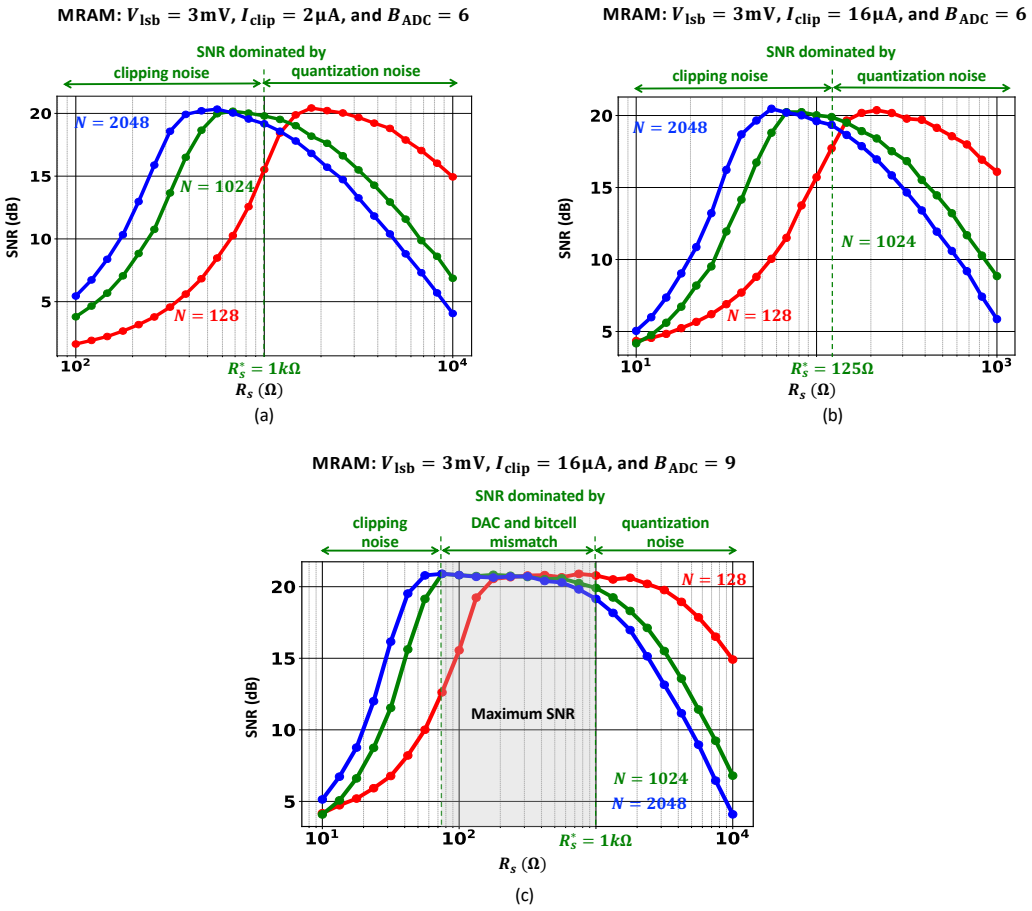


Figure 5.9: SNR vs. R_s trade-off in IMC MRAM crossbars with: (a) $I_{\text{clip}} = 2\,\mu\text{A}$ and $B_{\text{adc}} = 6\,\text{bit}$, (b) $I_{\text{clip}} = 16\,\mu\text{A}$ and $B_{\text{adc}} = 6\,\text{bit}$, and (c) $I_{\text{clip}} = 16\,\mu\text{A}$ and $B_{\text{adc}} = 9\,\text{bit}$.

linearly changes I_{ideal} in Eq. (5.4). Hence, the S_I needs to be correspondingly adjusted via a proportionate change in the R_s for maintaining the same SNR.

Similarly, we explore the impact of I_{clip} on SNR vs. R_s trade-off in Fig. 5.9. Note that the choice of I_{clip} is closely linked with the ADC bit precision B_{adc} since the quantization noise $I_{\text{qn}} \sim U\big(-\frac{I_{\text{clip}}}{2^{B_{\text{adc}}}}, \frac{I_{\text{clip}}}{2^{B_{\text{adc}}}}\big)$ (recall from Sec. 5.2). Hence, we consider following two cases in Fig. 5.9: I_{clip} is increased 8× from $2\,\mu\text{A}$ to $16\,\mu\text{A}$ (1) while preserving $B_{\text{adc}} = 6\,\text{bit}$ (Fig. 5.9(b)), and (2) while proportionately increasing B_{adc} from 6 bit to 9 bit in order to accommodate the larger current range (Fig. 5.9(c)). In Fig. 5.9(a) and Fig. 5.9(b), we find that when I_{clip} is increased while maintaining the B_{adc}, the same SNR vs. R_s trade-off is achieved at lower R_s values. Higher I_{clip} increases quantization noise, which needs to be compensated by appropriately increasing S_I

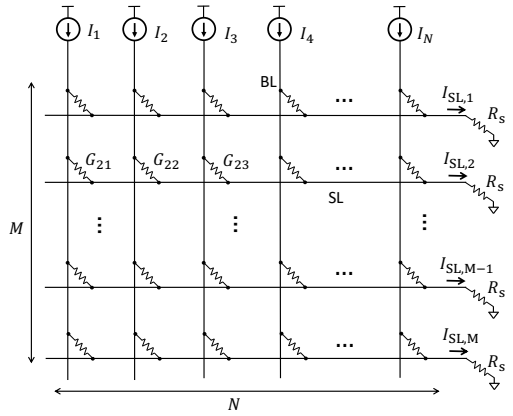


Figure 5.10: Current driven crossbar.

via proportionate reduction in R_s. Thus, the optimal value R_s^* is proportionately lower at the higher I_{clip}. The scenario gets interesting when B_{adc} is increased along with I_{clip}. In Fig. 5.9(c), there is a broad range of R_s (from $80\,\Omega$ to $1\,\text{k}\Omega$) where the SNR is maximized. In this regime, both quantization and clipping noise are small and the SNR is dominated by DAC and bitcell conductance mismatch variations. Note that the impact of these mismatch variations in Eq. (5.9) and Eq. (5.8) also gets scaled by the S_I making the SNR independent of R_s when these variations are dominant. Thus, higher I_{clip} and B_{adc} enable a broader design space by reducing clipping noise without any increase in the quantization noise.

5.4 Current Driven Crossbars

Figure 5.10 shows current driven crossbar, where each BL current $I_i = x_i I_{\text{lsb}}$ and I_{lsb} corresponds to the change in DAC current corresponding to the LSB. Unlike voltage driven crossbars, deriving a closed-form analytical expression for $I_{\text{SL},i}$ is mathematically intractable for an arbitrary value of M. Hence, we analyze a $2 \times N$ crossbar array. Doing so provides very good insights regarding the challenges involved in current DAC driven crossbars.

In the absence of any variations or noise, we apply the Kirchhoff's current law at the ith SL along with the charge conservation principle to obtain the SL current for $2 \times N$ crossbar array ($i \in \{1, 2\}$) as follows (see Appendix B

for a detailed derivation):

$$I_{\text{sig},i} = \left[\frac{R_s}{R_{\text{arr}} + 2R_s}\right]\left(\sum_{j=1}^{N} I_j\right) + \left[\frac{R_{\text{arr}}}{R_{\text{arr}} + 2R_s}\right]\left(\sum_{j=1}^{N} \frac{G_{1j} I_j}{G_{1j} + G_{2j}}\right) \tag{5.12}$$

$$= \beta_{\text{cm}} I_{\text{tot}} + \beta_{\text{dp}} I_{\text{dp}} \tag{5.13}$$

$$= \frac{1}{2}(1 - \beta_{\text{dp}}) I_{\text{tot}} + \beta_{\text{dp}} I_{\text{dp}} \tag{5.14}$$

where β_{dp} corresponds to the scaling of $I_{\text{sig},i}$ component that is proportional to the dot product (signal component), β_{cm} corresponds the scaling of the constant bias current component (bias component), and R_{arr} denotes the equivalent resistance corresponding to columnwise parallel combination of all bitcells as follows:

$$\frac{1}{R_{\text{arr}}} = \sum_{j=1}^{N} \frac{1}{R_{1,j} + R_{2,j}} \tag{5.15}$$

Notice that the SL currents in current DAC driven crossbars differ from those in VDAC driven crossbars in the following two ways:

1. **Need for constant sum of bitcell conductances along BL**: Each elementwise multiplication in I_{dp} is scaled by the sum of bitcell conductances along that particular BL. Hence, this sum needs to be constant across columns in order to achieve an accurate dot product. As discussed in Sec. 4.5.4, this sum can be equalized along columns by incorporating redundant BCA rows.

2. **Large input-dependent bias current**: Notice that the first term ($\frac{1}{2}(1 - \beta_{\text{dp}}) I_{\text{tot}}$) in Eq. (5.12) corresponds to an input vector $\mathbf{x}$ dependent bias current that is identical across all rows. Notice that this term appears due to the non-zero input impedance of sensing circuits, *i.e.* $R_s \neq 0$. Since R_{arr} and hence β_{dp} decreases with N, the relative proportion of this bias current increases with N. For example, with $N = 200$ and $R_s^* = 5\,\text{k}\Omega$ ($2\,\text{k}\Omega$) for PCM/RRAM (MRAM), we have:

   - PCM: $\beta_{\text{dp}} = 0.603$, $\frac{\beta_{\text{dp}}}{\beta_{\text{cm}}} \approx 304\%$
   - RRAM: $\beta_{\text{dp}} = 0.140$, $\frac{\beta_{\text{dp}}}{\beta_{\text{cm}}} \approx 32.5\%$
   - MRAM: $\beta_{\text{dp}} = 0.004$, $\frac{\beta_{\text{dp}}}{\beta_{\text{cm}}} \approx 0.9\%$

In PCM, thanks to its large R_{off} values, β_{dp} remains $3\times$ higher than β_{cm}. Even in RRAM, β_{dp} and β_{cm} remain comparable. However, in MRAM, β_{dp} turns out to be only 0.9% of β_{cm} due to its smaller $R_{\text{on}}/R_{\text{off}}$ values. Thus, a significant portion of $I_{\text{sig},i}$ corresponds to the constant bias current. Also, a larger value of β_{cm} corresponds to an amplified impact of the mismatch variations in the current DACs on $I_{\text{sig},i}$. Specifically, the impact of DAC mismatch variations, $I_{\text{dm},i}$ (recall Sec. 5.2), in current driven crossbars is given as:

$$I_{\text{dm},i} = \beta_{\text{cm}}\left(\sum_{j=1}^{N}\delta I_j\right) + \beta_{\text{dp}}\left(\sum_{j=1}^{N}\frac{G_{1j}\delta I_j}{G_{1j}+G_{2j}}\right) \tag{5.16}$$

where δI_j denotes the variation in jth current DAC. Notice that β_{cm} also scales the sum of total variations in the current DACs (first term in Eq. (5.16)). Thus, when $\beta_{\text{cm}} \gg \beta_{\text{dp}}$ (as is in the case of MRAM), the impact of mismatch variations in DACs will be larger than the signal current, *i.e.* $I_{\text{dm},i} \gg \beta_{\text{dp}} I_{\text{dp}}$.

Challenge 2 discussed above turned out to be the key difficulty in our chip prototype of current driven MRAM-DIMA in Sec. 4.6. In our prototype, it did turn out that $\beta_{\text{cm}} \gg \beta_{\text{dp}}$, resulting in SL currents getting overwhelmed by the signal-independent bias current $\beta_{\text{cm}}\left(\sum_{j=1}^{N}\delta I_j\right)$, resulting in a very low SNR during the chip measurements of current driven MRAM-DIMA prototype. This being the fundamental device-level limitation in the current drive setting, we concluded that one needs to employ on-chip learning based compensation to accurately recover $\beta_{\text{dp}} I_{\text{dp}}$ in the presence of DAC mismatch variations in the current driven MRAM crossbars.

5.5 Discussion

In this chapter, we analyzed the SNR of resistive crossbar arrays employed for in-memory computing. The analysis is device-agnostic and we study three memory device types, namely PCM, RRAM, and MRAM, having distinct resistive contrast. We find that there are optimal values sensing impedance and the number of columns that achieve maximum SNR in the case of voltage driven crossbars. We also considered current driven crossbars and identified

their challenges compared to VDAC driven crossbars. Based on our analysis, we draw the following four key conclusions:

- Voltage drive works well for IMC with resistive crossbars for all three memory device technologies (PCM, RRAM, and MRAM), despite large differences in their resistive contrast. The highest SNR is achieved with $R_s \approx 2\,\mathrm{k}\Omega$ for $N = 1000$, which is certainly feasible even after stringent area constraints. Furthermore, with $M \approx 20$ and $N \approx 1000$, one can reap the benefits of IMC while maintaining maximum DAC current reasonably low ($< 1\,\mu\mathrm{A}$).

- Current drive has a key disadvantage of large impact of DAC mismatches compared to voltage drive. It is particularly exacerbated in the case of MRAM due to its low resistive contrast and relatively large sensing impedance R_s. While reducing R_s by orders of magnitude will alleviate this challenge, it is often prohibitively expensive in terms of area and energy. Since the first term in Eq. (5.16) depends only on the inputs and is same for all SLs, one can compensate it via a shared circuit.

- MRAM achieves significantly poorer SNR vs. energy trade-off compared to RRAM and PCM (see Fig. 5.7), primarily due to its lower resistive contrast. Since this is a fundamental device limitation, it underscores the need to employ on-chip learning [65] based statistical error compensation (SEC) for improving SNR in MRAM based IMC.

- Equation (5.7) shows that ideally the bitcell resistance states should have their average $R_{\mathrm{avg}} \gg NR_s$ and their contrast $\frac{R_{\mathrm{off}}}{R_{\mathrm{on}}} \gg 1$ in order to achieve high SNR. Furthermore, given $E_{\mathrm{BCA,op}} \propto I_{\mathrm{sig}}$, one can trade-off the SNR improvements obtained via high average bitcell resistance and high resistive contrast to appropriately minimize the energy consumption. For example, higher R_{avg} and $\frac{R_{\mathrm{off}}}{R_{\mathrm{on}}}$ can enable smaller sensing circuit impedance R_s and higher ADC quantization step size, simplifying the sensing circuit design for the same target SNR.

Moving forward, an important next step is to characterize the impact of BL/SL parasitic impedance in crossbars on the SNR of in-memory computation. Also, in this chapter, we focus on crossbar arrays, where BLs are

perpendicular to SL. However, our approach can be followed to carry out similar analysis for parallel-bar arrays where BLs are parallel to SL.

CHAPTER 6

SUBSPACE ANALYSIS OF ADVERSARIAL PERTURBATIONS

6.1 Overview

Deep nets are vulnerable to *adversarial perturbations* [9], *i.e.*, changes imperceptible to humans result in incorrect decisions. Multiple approaches to construct adversarial examples (attack methods) have been proposed [28, 11, 29, 12, 30]. While the basic idea is to move the input in the direction of increasing loss, stronger attacks [13, 12] operate iteratively by perturbing the input in small increments. On the defense side, early attempts to provide adversarial defenses obfuscate gradient computation by introducing non-differentiable and/or randomized operations in the forward pass [35, 36, 37, 38]. However, such defenses were shown to be ineffective [42, 14], especially when the attacker employs adaptive attacks, *i.e.*, the attacks that modified specifically to evade a given defense. Today adversarial training (AT) [13, 22, 43] is the only empirical defense that has remained robust to a wide variety of the strongest possible known attacks.

Thus, defending deep nets against adversarial perturbations remains a formidable challenge. This is in part due to a lack of in-depth understanding of the underlying cause of deep nets' vulnerability to adversarial perturbations. Indeed, efforts to improve such understanding have led to intriguing observations/insights such as the role of the geometry of the decision boundary [187, 188], and the existence of an inherent trade-off between generalization and robustness [22, 55], the existence of universal perturbations [189], and the notion of robust vs. non-robust features [190]. Such insights, however, do not yet completely explain the cause of a DNN's adversarial vulnerability. Hence, further efforts are required to deepen our understanding of this phenomenon so that effective defenses against adversarial attacks can be developed.

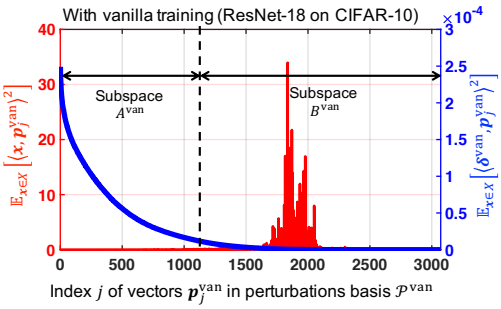
In this chapter, we employ subspace analysis to understand the geometric orientations of adversarial perturbations, and how they differ after vanilla vs. adversarial training. Specifically, we present and validate two hypotheses about geometric orientations of adversarial perturbations. In Sec. 6.2, we observe how adversarial perturbations are oriented relative to dominant image subspace and vice versa. In Sec. 6.3, we explore how orientations perturbations of different types, namely ℓ_∞, ℓ_2, and ℓ_1 perturbations, differ from each other in terms of their dominant subspaces. Note that the eventual goal of analysis presented in this chapter is to develop insights for improving robustness vs. cost trade-off in deep nets. We discuss the implications of our analysis in Sec. 6.4.

Without loss of generality, we employ ResNet-18 network on CIFAR-10 for the subspace analysis in this chapter. The analysis can be repeated for any other network or dataset. We refer to deep net parameterized by θ as $f_\theta(\boldsymbol{x})$, and the training set as $X = \{\boldsymbol{x}_i\}_{i=1}^R$ with cardinality $|X| = R$ and $\boldsymbol{x}_i \in \mathbb{R}^D$. For the CIFAR-10 dataset $R = 50,000$ and $D = 3072$. We employ vanilla training to obtain a non-robust network f_θ^{van}, while use TRADES [22] training (one of the SOTA defense) to obtain a robust network f_θ^{rob}.

6.2 Orthogonality between Images and Adversarial Perturbations

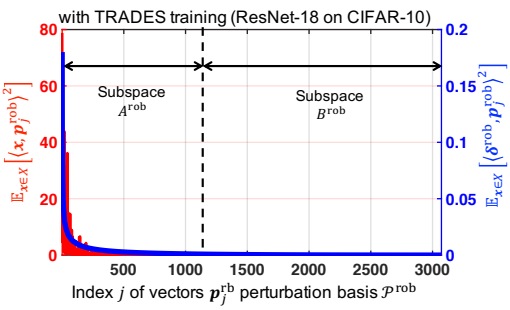
We begin by a following hypothesis (briefly alluded to in [188]): *adversarial perturbations of vanilla trained networks lie predominantly in a subspace orthogonal to that occupied by the input. Conversely, an adversarially trained network will find its adversarial perturbations to lie in the same subspace as the input.* We refer to this hypothesis as *Subspace Orthogonality of Adversarial Perturbations* (SOAP).

We employ a subspace analysis to empirically validate the SOAP hypothesis of adversarial perturbations. For both robust and non-robust deep nets we generate adversarial perturbations via DeepFool [11], *i.e.*, we iteratively find a perturbation vector $\boldsymbol{\delta}^\alpha \in \mathbb{R}^D \ \forall \alpha \in \{\text{van}, \text{rob}\}$ as an approximate solution



	$S = A^{\text{van}}$	$S = B^{\text{van}}$	$S = A^{\text{van}} \cup B^{\text{van}}$
$\frac{\mathbb{E}_{x\in X}\left[\left\|S^T\boldsymbol{x}\right\|_2^2\right]}{\mathbb{E}_{x\in X}\left[\left\|\boldsymbol{x}\right\|_2^2\right]}$	2.42%	**97.58%**	100%
$\frac{\mathbb{E}_{x\in X}\left[\left\|S^T\boldsymbol{\delta}^{\text{van}}\right\|_2^2\right]}{\mathbb{E}_{x\in X}\left[\left\|\boldsymbol{\delta}^{\text{van}}\right\|_2^2\right]}$	**96.82%**	3.18%	100%

(a)



	$S = A^{\text{rob}}$	$S = B^{\text{rob}}$	$S = A^{\text{rob}} \cup B^{\text{rob}}$
$\frac{\mathbb{E}_{x\in X}\left[\left\|S^T\boldsymbol{x}\right\|_2^2\right]}{\mathbb{E}_{x\in X}\left[\left\|\boldsymbol{x}\right\|_2^2\right]}$	**98.08%**	1.98%	100%
$\frac{\mathbb{E}_{x\in X}\left[\left\|S^T\boldsymbol{\delta}^{\text{rob}}\right\|_2^2\right]}{\mathbb{E}_{x\in X}\left[\left\|\boldsymbol{\delta}^{\text{rob}}\right\|_2^2\right]}$	**93.56%**	6.44%	100%

(b)

Figure 6.1: Dominant adversarial perturbation subspaces in robust vs. non-robust networks: *(top plots)* average squared projections of training images $\mathbb{E}_{\boldsymbol{x}\in X}[\langle \boldsymbol{x}, \boldsymbol{p}_j^{\alpha}\rangle^2]$ and DeepFool perturbations $\mathbb{E}_{\boldsymbol{x}\in X}[\langle \boldsymbol{\delta}^{\alpha}, \boldsymbol{p}_j^{\alpha}\rangle^2]$ on the perturbation basis P^{α} for (a) non-robust ResNet-18 $\alpha = \text{van}$, and (b) robust ResNet-18 $\alpha = \text{rob}$, where $\langle\cdot\rangle$ denotes dot product. *(bottom tables)* Fraction of average squared norm within the two orthogonal subspaces $A^{\alpha} = \{\boldsymbol{p}_1^{\alpha}, \ldots, \boldsymbol{p}_{1200}^{\alpha}\}$, $B^{\alpha} = \{\boldsymbol{p}_{1201}^{\alpha}, \ldots, \boldsymbol{p}_{3072}^{\alpha}\}$, together spanning entire perturbation basis P^{α} for (a) non-robust network $\alpha = \text{van}$, and (b) robust network $\alpha = \text{rob}$, where $S^T\boldsymbol{a}$ denotes projection of vector $\boldsymbol{a}$ on subspace S.

to the problem:

$$\arg\min_{\boldsymbol{\delta}^{\alpha}} \|\boldsymbol{\delta}^{\alpha}\|_2 \quad \text{s.t.} \quad f_{\theta}^{\alpha}(\boldsymbol{x} + \boldsymbol{\delta}^{\alpha}) \neq f_{\theta}^{\alpha}(\boldsymbol{x}), \alpha \in \{\text{van}, \text{rob}\} \tag{6.1}$$

Note that DeepFool perturbations are the vectors with approaximately the smallest ℓ_2 magnitude. Hence, they can be viewed as directions to the nearest decision boundary for their corresponding inputs.

Further, following [188], we compute the singular vector basis $\mathcal{P}^{\alpha} = \{\boldsymbol{p}_1^{\alpha}, \ldots, \boldsymbol{p}_D^{\alpha}\}$ for the set of adversarial (DeepFool) perturbations $\Delta^{\alpha} = \{\boldsymbol{\delta}_1^{\alpha}, \ldots, \boldsymbol{\delta}_D^{\alpha}\}$ $\forall i, \alpha \in \{\text{van}, \text{rob}\}$. Again, the singular vectors $\boldsymbol{p}_i^{\alpha}$ are ordered in descending order of the corresponding singular values. We refer to the basis $\mathcal{P}^{\alpha}$ as the *perturbation basis*.

Figure 6.1(a) shows the mean squared projections of the images $\boldsymbol{x}_i$ (red) and DeepFool perturbations $\boldsymbol{\delta}_i^{\text{van}}$ (blue) onto the perturbation basis $\mathcal{P}^{\text{van}}$, averaged over the dataset X. We find that the adversarial perturbations for f_{θ}^{van} (blue) lie in a subspace that is (mostly) *orthogonal* to that of the

dominant image subspace (red). Indeed, it can be seen that these adversarial perturbations exploit dimensions with small input projections. One might conjecture that the dimensions having small input projections may not be critical for a classification task. However, [188] demonstrates that the small image projections on the dominant singular vectors $\mathcal{P}^{\text{van}}$ do in fact contribute significantly to the network's natural accuracy. Thus, small perturbations in those dimensions cause a catastrophic drop in the network's robust accuracy. In contrast, for the robust deep net f_θ^{rob}, Fig. 6.1(b) indicates that the adversarial perturbations and images predominantly lie in the *same* subspace.

We quantify this orthogonality (SOAP) property by computing the ratio of the squared norm of the projections in the two mutually orthogonal subspaces $A^\alpha = \{\boldsymbol{p}_j^\alpha\}_{j=1}^{1200}$ and $B^\alpha = \{\boldsymbol{p}_j^\alpha\}_{j=1201}^{3072}$ of basis $\mathcal{P}^\alpha = A^\alpha \bigcup B^\alpha$. Hence, for a given image $\boldsymbol{x}_i$,

$$\|\boldsymbol{x}_i\|_2^2 = \|(A^\alpha)^T \boldsymbol{x}_i\|_2^2 + \|(B^\alpha)^T \boldsymbol{x}_i\|_2^2 \tag{6.2}$$

where $S^T \boldsymbol{a}$ denotes the projection of vector $\boldsymbol{a}$ onto the subspace S, and $\|\cdot\|_2$ denotes ℓ_2 norm. Hence, the ratio $\|(A^\alpha)^T \boldsymbol{x}_i\|_2^2 / \|\boldsymbol{x}_i\|_2^2$ indicates the extent to which a given vector $\boldsymbol{x}_i$ lies in the subspace A^α. Note that A^α is the dominant subspace for the DeepFool perturbations since the singular vectors are ordered in the descending order of the corresponding singular values.

The table in Fig. 6.1(a) shows that the vanilla network f_θ^{van} has 96.82% of the average squared ℓ_2 norm of the perturbations $\boldsymbol{\delta}_i$ lying in A^{van} while 97.58% of the average squared ℓ_2 norm of the training images lie in the orthogonal subspace B^{van}. Thus, the DeepFool perturbations and the training images lie almost entirely in mutually orthogonal subspaces. In contrast, Fig. 6.1(b) shows that the adversarially trained robust network f_θ^{rob} has 93.56% of the average squared ℓ_2 norm of the adversarial perturbations *and* 98.08% of that of the training images lie in the *same* subspace A^{rob}. Thus, both images and the DeepFool perturbations for a robust network predominantly lie in the same subspace.

The distinction between non-robust and robust networks observed in Fig. 6.1 corroborates the generative behavior of adversarial perturbations of robust networks [190, 43]: adversarial perturbations of robust networks exhibit semantics similar to images because they predominantly lie in the same subspace. In the case of non-robust networks, the perturbations remain seman-

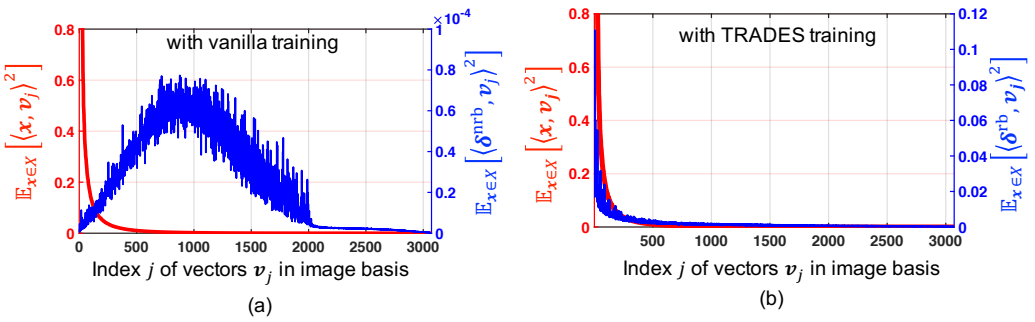


Figure 6.2: Average squared projections of training images $\mathbb{E}_{\boldsymbol{x}\in X}[\langle \boldsymbol{x}, \boldsymbol{v}_j\rangle^2]$ and DeepFool perturbations $\mathbb{E}_{\boldsymbol{x}\in X}[\langle \boldsymbol{\delta}^\alpha, \boldsymbol{v}_j^\alpha\rangle^2]$ on the singular vector basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{3072}\}$ of training images for: (a) ResNet-18 trained with standard training (α = van), and (b) ResNet-18 trained with TRADES training, (α = rob).

tically uninterpretable because of their orthogonality to the image subspace.

The distinction between the alignment of adversarial perturbations and the images for f_θ^{van} and f_θ^{rob} is also observed when projecting onto the image basis $\mathcal{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_D\}$ as shown in Fig. 6.2. The image basis $\mathcal{V}$ is obtained via a singular vector decomposition of the training images and vectors $\boldsymbol{v}_i$ in $\mathcal{V}$ are ordered in descending order of the corresponding singular values. Figure 6.2(a) shows the orthogonality of the DeepFool perturbations observed after vanilla training, while Fig. 6.2(b) shows the collinearity between the perturbation and image subspaces after TRADES training.

6.3 Distinction between Different Perturbation Models

In this section, we employ subspace methods to comprehend the distinction between ℓ_∞, ℓ_2 and ℓ_1 perturbation types. For each input $\boldsymbol{x}_i \in \mathbb{R}^D$ in dataset X, consider adversarial perturbations $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, and $\boldsymbol{\gamma}_i$ bounded within ℓ_∞, ℓ_2, and ℓ_1 norms, respectively. We can employ standard PGD attacks to obtain these perturbations.

We begin with the following hypothesis illustrated in Fig. 6.3: *The perturbations $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ corresponding to input $\boldsymbol{x}$ have directions that differ significantly if the curvature of the decision boundary is high in the neighborhood of $\boldsymbol{x}$. Conversely, if the curvature of the decision boundary is low, the perturbations $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ tend to lie in similar directions.*

In fact, the curvature of the decision boundary of deep nets has been stud-

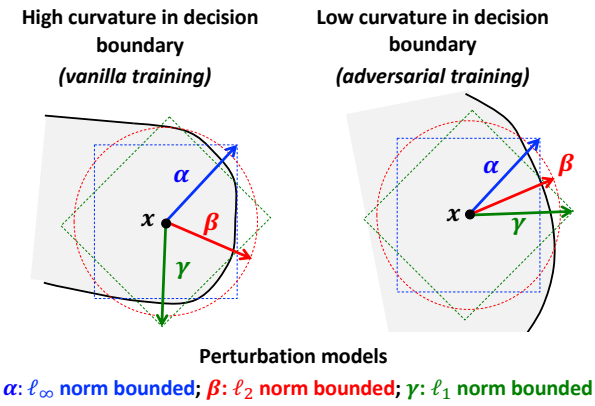


Figure 6.3: Illustration of the impact of AT on the curvature of the classifier decision boundary curvature and therefore on the orientation of perturbations $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ for an input $\boldsymbol{x}$.

ied thoroughly [191, 192, 188]. For example, Moosavi-Dezfooli *et al.* [191] observed high curvature of the decision boundary around most inputs after vanilla training and employed it to explain the existence of universal perturbations [189]. Furthermore,Moosavi-Dezfooli *et al.* [192] made a remarkable observation that employing single-attack AT alone reduces the curvature of the decision boundary and regularizing decision boundary curvature leads to gains in adversarial robustness.

Since single-attack AT reduces the curvature of the decision boundary, we test our hypothesis by studying the following two networks on CIFAR-10 data: a *non-robust* ResNet18 f_θ^{van} trained using vanilla training, and a *robust* ResNet18 f_θ^{rob} trained using the TRADES [22] single-attack AT framework employing ℓ_∞ perturbations.

We compute perturbations $\boldsymbol{\alpha}_i^\kappa$, $\boldsymbol{\beta}_i^\kappa$, and $\boldsymbol{\gamma}_i^\kappa$ for each $\boldsymbol{x}_i \in X$ for both networks, *i.e.*, $\kappa \in \{\text{van}, \text{rob}\}$. Now we treat them as vectors in $\mathbb{R}^D$ and study their relative orientations via a subspace analysis. We compute the singular vector basis $\mathcal{P}^\kappa = \{\boldsymbol{p}_1^\kappa, \ldots, \boldsymbol{p}_D^\kappa\}$ for the set of ℓ_2 bounded perturbations $\Delta^\kappa = \{\boldsymbol{\beta}_1^\kappa, \ldots, \boldsymbol{\beta}_D^\kappa\}$ $\forall i, \kappa \in \{\text{van}, \text{rob}\}$. The singular vectors $\boldsymbol{p}_i^\kappa$ are ordered in descending order of their singular values.

We plot the normalized mean squared projections of the three perturbation types on the perturbation basis $\mathcal{P}^\kappa$ for $\kappa \in \{\text{van}, \text{rob}\}$ in Fig. 6.4(a) and Fig. 6.4(b), respectively. The contrast is clear immediately. The perturbations of a vanilla trained network roll-off gradually to occupy a larger

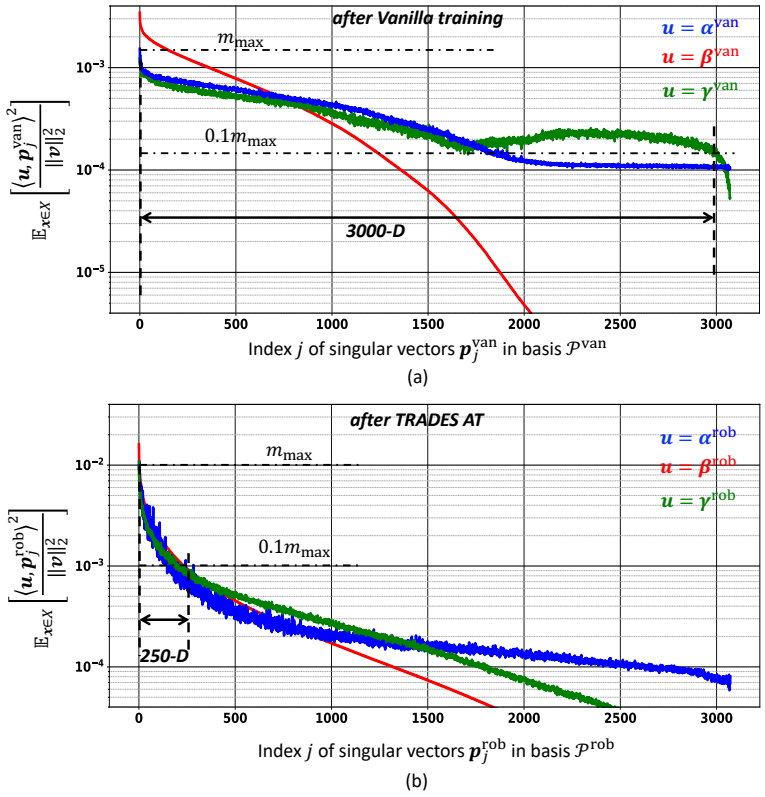


Figure 6.4: Normalized mean squared projections of three perturbation types on the singular vector basis $\mathcal{P}^{\kappa}$ of $\boldsymbol{\beta}^{\kappa}$ perturbations): (a) after vanilla training ($\kappa \equiv$ van), and (b) after TRADES training ($\kappa \equiv$ rob) for ResNet18 on CIFAR-10.

subspace as indicated in Fig. 6.4(a). Specifically, the projections of $\boldsymbol{\alpha}^{\text{van}}$ and $\boldsymbol{\gamma}^{\text{van}}$ occupy almost all 3000 directions in the basis $\mathcal{P}^{\kappa}$, *i.e.*, their mean squared projections are within $\sim 10\%$ of the maximum mean squared projection value $m_{\max}$. This shows that the dominant singular vectors of $\boldsymbol{\beta}^{\text{van}}$ are not well-aligned with $\boldsymbol{\alpha}^{\text{van}}$ and $\boldsymbol{\gamma}^{\text{van}}$. This misalignment between the perturbation vectors implies that all three would need to be included during training (multi-attack AT) in order to provide robustness against the union of perturbation models. This is exactly the approach taken by Maini *et al.* [107] resulting the significant ($10\times$) complexity increase over single-attack AT.

In contrast, with TRADES AT, all three perturbations types are *squeezed* into a much *smaller* subspace spanning only the top 250 singular vectors in the perturbation basis $\mathcal{P}^{\text{rob}}$. Outside these 250 dimensions, the mean squared projections fall to $< 10\%$ of their maximum value. Thus, single-ℓ_∞ attack AT enhances the alignment between different perturbations which in turn

improves robustness. However, this improvement is small, e.g., while robust accuracy of 50% is achieved against ℓ_∞ perturbations, robust accuracy of only 15% is realized against ℓ_1 attacks. Nevertheless, this type of subspace analysis hints at other methods to enhance the alignment between various perturbation vectors within a single-attack AT framework. The discovery of such methods will provide robustness to union of perturbation models at the complexity of single-attack AT.

In summary, the results in Fig. 6.4 validate the hypothesis that single-attack AT increases the alignment of different perturbation types *on average* due to the reduction in the decision boundary curvature around most inputs.

6.4 Discussion

In this chapter, we carried out subspace analysis of adversarial perturbations. Specifically, we proposed and validated two hypotheses about geometric orientations of adversarial perturbations. The SOAP hypothesis shows that the decision boundary of vanilla trained deep net gets warped around the natural images in the dimensions orthogonal to dominant image subspace. This also aligns well with the observation that deep nets overfit in the dimensions having low data variance.

The second hypothesis conveys that the different types of adversarial perturbations are squeezed into a smaller subspace even after an adversarial training employing only one type of perturbations. We exploit this insight in Chapter 7 to minimize the training cost of adversarially robust deep nets via shaped noise augmented processing.

CHAPTER 7

EFFICIENT AND ROBUST DEEP NET TRAINING VIA NOISE SHAPING

7.1 Overview

Today *adversarial training* (AT) provides state-of-the-art (SOTA) empirical defense against adversarial perturbations. For this, adversarial perturbations are used during training to optimize a *robust* loss function [13, 22, 43, 44]. Early AT frameworks [13, 22] were 7×-to-10× more computationally demanding than vanilla training. More recent works [43, 44, 47] have significantly reduced the computational demands of AT via *single-step attacks* and *superconvergence*.

However, today's AT frameworks predominantly focus on a *single-attack*, *i.e.*, they seek robustness to a single perturbation, typically ℓ_∞-bounded [43, 44, 45, 22, 46, 47, 48, 49, 50, 51, 52, 53, 54, 193]. This results in low performance against other perturbations such as ℓ_2, ℓ_1, or the union of $(\ell_\infty, \ell_2, \ell_1)$. Indeed, as shown in Fig. 7.1, four SOTA single-attack AT frameworks (*black markers*) employing only ℓ_∞-bounded perturbations achieve low adversarial accuracy $\mathcal{A}_{\text{adv}}^{(U)}$ of $\approx$ 15%-to-20% against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations. Recent extensions in AT [107, 106, 194] do seek higher $\mathcal{A}_{\text{adv}}^{(U)}$ but only at the expense of a 6×-to-10× increase in the total training time (*blue markers in Fig. 7.1*). The large training time of these AT frameworks has inhibited their application to large-scale datasets such as ImageNet, *e.g.*, [107, 106] show results for MNIST and CIFAR-10 only, while [194] only additionally shows 64×64 ImageNet-100 results.

The high training time for AT frameworks arises from two sources: (i) the need to employ larger networks, *e.g.*, MSD [107] with ResNet-18 achieves higher $\mathcal{A}_{\text{adv}}^{(U)}$ than PAT [194] with ResNet-50 (see Fig. 7.1); and (ii) the need to incorporate multiple perturbations during each attack step and a higher overall number of attack steps, *e.g.*, 50 in MSD [107], 20 in AVG [106].

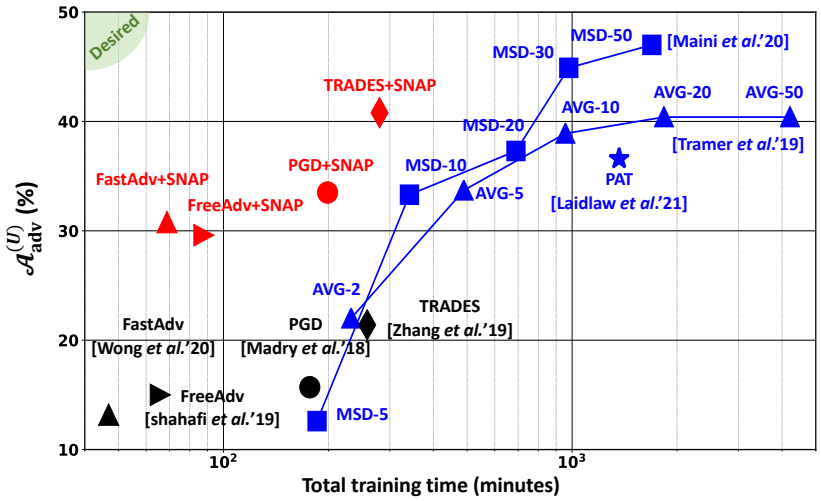


Figure 7.1: Adversarial accuracy ($\mathcal{A}_{\text{adv}}^{(U)}$) against union of ($\ell_\infty$, ℓ_2, ℓ_1) vs. measured wall-clock total training time on CIFAR-10 with different AT frameworks on single NVIDIA TESLA P100 GPU. $\epsilon = (0.031, 0.5, 12)$ for (ℓ_∞, ℓ_2, ℓ_1) perturbations, respectively. SNAP enhances robustness with a small increase in training time. All frameworks except PAT employ ResNet-18.

Obviously one can always reduce the number of attack steps in MSD/AVG to proportionally reduce training time. Doing so results in training time and $\mathcal{A}_{\text{adv}}^{(U)}$ to rapidly approach the training complexity and $\mathcal{A}_{\text{adv}}^{(U)}$ of standard AT frameworks, *e.g.*, a five-step MSD and two-step AVG is equivalent in training time and accuracy to PGD and TRADES, respectively. Notwithstanding the expensive nature of 50-step multi-attack training, today MSD [107] achieves a SOTA $\mathcal{A}_{\text{adv}}^{(U)}$ of 47% with ResNet-18 on CIFAR-10.

This poses a question: Can we approach the high robustness of multiple-attack AT such as 50-step MSD against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations while maintaining the low training time of fast single-attack AT frameworks such as FreeAdv [43] and FastAdv [44]?

In our quest to answer this question we find that noise augmentation using adequately shaped noise within standard single-attack AT frameworks employing ℓ_∞-bounded perturbations significantly improves robustness against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations. The improvement appears to be a consequence of a well-established byproduct of AT frameworks – the reduction in the curvature of the decision boundary of networks trained using single-attack AT [191, 192]. We confirm this connection by quantifying the impact of single-attack AT on the geometric orientations of different perturbations.

Based on this insight, we propose Shaped Noise Augmented Processing (SNAP) – *a method to enhance robustness against the union of perturbation types by augmenting single-attack AT frameworks*. SNAP prepends a deep net with a shaped noise (SN) augmentation layer (see Fig. 7.2) whose distribution parameter Σ is learned with that of the network (θ) within any standard single-attack AT framework. SNAP improves the robustness of four SOTA ℓ_∞-AT frameworks against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations by 15%-to-20% on CIFAR-10 (*red markers in Fig. 7.1*) with only a modest ($\sim$ 10%) increase in training time. This expands the capabilities of widely popular single-attack ℓ_∞ AT frameworks to providing robustness to the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations without sacrificing training efficiency. We validate SNAP's benefits via thorough comparisons with *nine SOTA adversarial training and randomized smoothing frameworks* across different operating regimes on both CIFAR-10 and ImageNet.

One tangible outcome of our work – we demonstrate *for the first time* ResNet-50 (ResNet-101) networks on ImageNet that achieve $\mathcal{A}_{\text{adv}}^{(U)} = 32\%$ (35%) against the union of $(\ell_\infty(\epsilon = 2/255), \ell_2(\epsilon = 2.0), \ell_1(\epsilon = 72.0))$ perturbations.

Our code is available at https://github.com/adpatil2/SNAP.

7.2 Shaped Noise Augmented Processing (SNAP)

We show that single-attack AT can be enhanced to address multiple perturbations by introducing noise to appropriately *wiggle* the ℓ_∞-bounded perturbations (Fig. 7.2(a)). However, to do so, the noise distribution needs to be *chosen* and *shaped* appropriately to minimize its impact on natural accuracy and robustness to ℓ_∞-bounded perturbations.

We experiment with both ℓ_∞ and ℓ_2 perturbations in single-attack AT frameworks and find ℓ_∞-AT to be suitable for our proposed shaped noise augmentation (see Sec. 7.3.2 for details). Hence, in this section, we describe SNAP for single-attack AT frameworks employing ℓ_∞ perturbations.

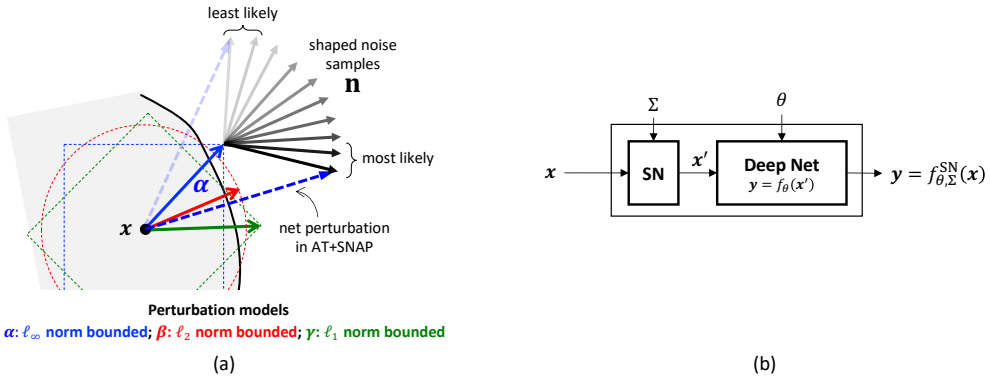


Figure 7.2: SNAP: (a) intuition underlying SNAP (not an exact depiction), and (b) SNAPnet $f^{\text{SN}}_{\theta,\Sigma}(\boldsymbol{x})$ constructed from a given deep net $f_\theta(\boldsymbol{x})$ by prepending a shaped noise (SN) augmentation layer which perturbs the primary input $\boldsymbol{x}$ with noise $\mathbf{n}$ whose distribution parameter Σ is learned during AT along with the base network parameter θ.

7.2.1 SNAPnet

A deep net $f_\theta(\boldsymbol{x}) : \mathbb{R}^D \to \{0,1\}^C$ parametrized by θ maps the input $\boldsymbol{x} \in \mathbb{R}^D$ to a one-hot vector $\boldsymbol{y} \in \{0,1\}^C$ over C classes.

We construct a SNAP-based deep net (SNAPnet) $f^{\text{SN}}_{\theta,\Sigma}(\boldsymbol{x})$ by introducing an additive shaped noise (SN) layer (Fig. 7.2(b)), where the noise distribution parameter Σ is learned during training. Formally,

$$\boldsymbol{y} = f^{\text{SN}}_{\theta,\Sigma}(\boldsymbol{x}) = f_\theta\big(\boldsymbol{x} + \mathbf{n}\big) = f_\theta\big(\boldsymbol{x} + V\Sigma\mathbf{n}_0\big) \tag{7.1}$$

where $\mathbf{n}_0 \sim \mathcal{L}(0, \mathbf{I}_{D\times D})$ is an isotropic Laplace noise vector with zero mean and identity covariance matrix, $\Sigma = \text{Diag}[\sigma_1, \ldots, \sigma_D]$ is a distribution parameter denoting its per-dimension standard deviation, $\mathbf{I}_{D\times D}$ denotes the $D \times D$ identity matrix, and $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_D]$ denotes a basis in $\mathbb{R}^D$. We also studied Gaussian and Uniform distributed $\mathbf{n}_0$, but empirically find the Laplace distribution to yield better results (Sec. 7.3.2). We use $V = \mathbf{I}_{D\times D}$ for all our experiments everywhere except in Sec.7.5.2 where we study other options for V.

The final classification decision d is computed via

$$d = \arg\max_c \Big[\mathbb{E}_{\mathbf{n}}\big[\boldsymbol{y}\big]\Big]_c \tag{7.2}$$

Algorithm 1 Training SNAPnet

Input: training set X; basis $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_D]$; total noise power P_{noise}; mini-batch size r; baseline training method BASE; noise variance update frequency U_f; Total number of epochs T

Initialize: noise variances $\Sigma_0 = \text{Diag}[\sigma_{1,0}, \ldots, \sigma_{D,0}]$.

Output: robust network $f^{\text{SN}}_{\theta,\Sigma}$, noise variances $\Sigma_T = \text{Diag}[\sigma^2_{1,T}, \ldots, \sigma^2_{D,T}]$.

1: **for** epoch $t = 1 \ldots T$ **do**

2: **for** mini-batch $B = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_r\}$ **do** $\theta \leftarrow \text{BASE}_{\ell_\infty}\left(f^{\text{SN}}_{\theta,\Sigma_t}(\{\boldsymbol{x}_i\}_{i=1}^r), \theta\right)$ ▷ BASE() Training

3: **end for**

4: **if** $t \mod U_f = 0$ **then** ▷ SNAP Distribution Update once every U_f epochs

5: **for** mini-batch $B = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_r\}$ **do**

6: $\{\boldsymbol{x}^{\text{adv}}_i\}_{i=1}^r \leftarrow \text{PGD}^{(K)}_{\ell_2}\left(f^{\text{SN}}_{\theta,\Sigma_t}(\{\boldsymbol{x}_i\}_{i=1}^r)\right)$; $\boldsymbol{\eta}_i = \boldsymbol{x}^{\text{adv}}_i - \boldsymbol{x}_i \;\forall\; i \in \{1, \ldots, r\}$

7: $\gamma_j \leftarrow \gamma_j + \sum_{i=1}^r \left(\langle \boldsymbol{v}_j, \boldsymbol{\eta}_i \rangle\right)^2 \quad \forall j \in \{1, \ldots, D\}$ ▷ Accumulate projections; See Eq. (7.3)

8: **end for**

9: $\sigma^2_{j,t+1} = P_{\text{noise}} \frac{\sqrt{\gamma_j}}{\sum_{k=1}^D \sqrt{\gamma_k}} \quad \forall j \in \{1, \ldots, D\}$ ▷ Normalize accumulated projections; See Eq. (7.3)

10: **else**

11: $\Sigma_{t+1} \leftarrow \Sigma_t$

12: **end if**

13: **end for**

where $[\boldsymbol{a}]_c$ denotes the c-th element of vector $\boldsymbol{a}$. Note, the shaped noise perturbs the input $\boldsymbol{x}$ with a noise source $\mathbf{n} = V\Sigma\mathbf{n}_0$ (Eq. (7.1)). The distribution parameter Σ is learned in the presence of any standard AT method [13, 22, 43] used for learning deep net parameters θ as described next.

7.2.2 Training SNAPnet

Algorithm 1 summarizes the procedure for training SNAPnet $f^{\text{SN}}_{\theta,\Sigma}(\boldsymbol{x})$. In each epoch, an arbitrary AT method BASE() (line 2) updates network parameters θ with input perturbed by noise $\mathbf{n}$. Here BASE() can be any established AT framework [13, 22, 43, 44] employing ℓ_∞ perturbation.

The SNAP parameter Σ is updated once every $U_f = 10$ epochs via a *SNAP distribution update* (lines 4-10). In this update, the per-dimension noise variance σ^2_j is updated proportional to the root mean squared projection of

the adversarial perturbations $\boldsymbol{\eta}$ on the basis V given a total noise constraint $\sum_{j=1}^{D} \sigma_j^2 = P_{\text{noise}}$, where P_{noise} denotes the total noise power. Formally,

$$\sigma_j^2 \propto \sqrt{\mathbb{E}_{\boldsymbol{x} \in X}\big(\langle \boldsymbol{\eta}, \boldsymbol{v}_j \rangle^2\big)} \quad \text{s.t.} \quad \sum_{j=1}^{D} \sigma_j^2 = P_{\text{noise}} \tag{7.3}$$

where $\boldsymbol{\eta}$ is the ℓ_2 norm-bounded PGD adversarial perturbation for the given input $\boldsymbol{x} \in X$ (line 6). Note that these ℓ_2 perturbations are employed *only* for noise shaping and are distinct from the ℓ_∞ perturbations employed by BASE() AT (line 2). Also, ℓ_∞ perturbations cannot be used here since their projections are constant $\forall j$ when $V = \mathbf{I}_{D \times D}$, whereas employing ℓ_1 perturbations leads to poor shaping due to high sparsity.

Thus, in SNAP, the average squared ℓ_2 norm of the noise vector $\mathbf{n}$ is held constant at P_{noise} while adapting the noise variances in the individual dimensions so as to align the noise vectors with the adversarial perturbations *on average*. Intuitively, the decision boundary is pushed aggressively in those directions.

7.2.3 Remarks

Note that the SNAP distribution update is distinct from BASE() AT. Hence, SNAP does not require any hyperparameter tuning in BASE(). For fairness to baselines we keep all hyperparameters identical when introducing SNAP in all our experiments. However, SNAP introduces a new hyperparameter P_{noise}, which permits to trade adversarial robustness $\mathcal{A}_{\text{adv}}^{(U)}$ for natural accuracy $\mathcal{A}_{\text{nat}}$. This trade-off is explored in Sec. 7.3.3.

The computational overhead of SNAP is small ($\sim 10\%$) since the *SNAP Distribution Update* occurs once in 10 epochs using just 20% of the training data to update the noise standard deviations σ_j.

7.3 Experimental Results

7.3.1 Setup

Following experimental settings of prior work [22, 43, 107], we employ a ResNet-18 network for CIFAR-10 experiments and both ResNet-50 and ResNet-101 networks for ImageNet experiments. Accuracy on clean test data is referred to with $\mathcal{A}_{\text{nat}}$ and accuracy on adversarially perturbed test data is referred to via $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$, $\mathcal{A}_{\text{adv}}^{(\ell_2)}$, and $\mathcal{A}_{\text{adv}}^{(\ell_1)}$, for ℓ_∞, ℓ_2, and ℓ_1 norm bounded perturbations, respectively. Accuracy against the *union* of all three perturbations is denoted by $\mathcal{A}_{\text{adv}}^{(U)}$.

For a fair robustness comparison, our evaluation setup closely follows the setup of [107] for CIFAR-10 data: (1) choose norm bounds $\epsilon = (0.031, 0.5, 12.0)$ for (ℓ_∞, ℓ_2, ℓ_1) perturbations, respectively; (2) scale norm bounds for images to lie between $[0, 1]$; (3) choose the PGD attack configuration to be *100 iterations with 10 random restarts* for all perturbation types;[1] and (4) estimate $\mathcal{A}_{\text{adv}}^{(U)}$ as the fraction of test data that is *simultaneously* resistant to all three perturbation models.

Following the guidelines of [14], we carefully design *adaptive* PGD attacks that target the full defense – SN layer – since SNAPnet is end-to-end differentiable. Specifically, we backpropagate to primary input $\boldsymbol{x}$ through the SN layer (see Fig. 7.2). Thus, the final shaped noise distribution is exposed to the adversary. We also account for the expectation $\mathbb{E}_{\mathbf{n}}[\cdot]$ in Eq. (7.2) by explicitly averaging deep net logits over $N_0(= 8)$ noise samples *before* computing the gradient, which eliminates any gradient obfuscation, and is known to be the strongest attack against noise augmented models [195].

On CIFAR-10 data, we compare with the following seven key SOTA AT frameworks: PGD [13], TRADES [22], FreeAdv [43], FastAdv [44], AVG [106], MSD [107], PAT [194]. We also compare with two randomized smoothing frameworks [196, 195]. Thanks to their GitHub code releases, we first successfully reproduce their results with a ResNet-18 network in our environment. In the case of PAT [194], we evaluate and compare with their pretrained ResNet-50 model on CIFAR-10. We compare all training times on a single NVIDIA P100 GPU. On ImageNet data, we primarily compare to

[1] Following [107], we also run all attacks on a subset of the first 1000 test examples with 10 random restarts for CIFAR-10 data.

Table 7.1: ResNet-18 CIFAR-10 results showing the impact of SNAP augmentation of PGD [13] AT framework with ℓ_∞ (*top*) and ℓ_2 (*bottom*) perturbations where [G], [U], and [L], denote shaped Gaussian, Uniform, and Laplace noise.

Method	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ $\epsilon = 0.03$	$\mathcal{A}_{\text{adv}}^{(\ell_2)}$ $\epsilon = 0.5$	$\mathcal{A}_{\text{adv}}^{(\ell_1)}$ $\epsilon = 12$	$\mathcal{A}_{\text{adv}}^{(U)}$
PGD AT with ℓ_∞ perturbations					
PGD	84.6	**48.8**	62.3	15.0	15.0
+SNAP[G]	80.7	45.7	66.9	34.6	31.9
+SNAP[U]	**85.1**	42.7	**66.7**	28.6	26.6
+SNAP[L]	83.0	44.8	**68.6**	**40.1**	**35.6**
PGD AT with ℓ_2 perturbations					
PGD	**89.3**	28.8	**67.3**	31.8	25.1
+SNAP[G]	83.0	**35.0**	65.8	39.9	30.2
+SNAP[U]	86.4	32.3	66.7	30.2	25.0
+SNAP[L]	84.8	33.4	66.1	**42.5**	**30.8**

FreeAdv [43]. We train ResNet-50 and its SNAPnet version with FreeAdv on a Google Cloud server with four NVIDIA P100 GPUs to compare their accuracy and training times. Our code and pretrained models are available at https://github.com/adpatil2/SNAP.

7.3.2 Impact of Noise Distribution and Model of BASE() AT Perturbations

In this subsection, we first study the impact of employing ℓ_∞ vs. ℓ_2 perturbations in BASE AT() (see line 2 in Alg. 1) on $\mathcal{A}_{\text{adv}}^{(U)}$. For each choice, we further experiment with three distributions for the SN layer in Fig. 7.2(b) viz. Gaussian, Uniform, and Laplace. We do not consider ℓ_1 perturbations in BASE AT() since Maini *et al.* [107] showed that employing ℓ_1 single-attack AT achieves very low robustness to all attacks. We choose PGD [13] AT as BASE AT() for this ablation study. For a fair comparison across the noise distributions, we fix $P_{\text{noise}} = 160$, enforcing all noise vectors to have the same average ℓ_2 norm. For each distribution, the noise is shaped per the procedure summarized in Alg. 1.

As observed in Table 7.1, ℓ_∞-PGD AT achieves much lower $\mathcal{A}_{\text{adv}}^{(U)}$ than ℓ_2-PGD AT, an observation also reported by Maini *et al.* [107]. With SNAP, however, we find that there is an interaction between the perturbation model

Table 7.2: ResNet-18 CIFAR-10 results showing the impact of SNAP augmentation of established ℓ_∞-AT frameworks. The computational overhead of SNAP is limited to $\sim 10\%$.

Method	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ $\epsilon = 0.03$	$\mathcal{A}_{\text{adv}}^{(\ell_2)}$ $\epsilon = 0.5$	$\mathcal{A}_{\text{adv}}^{(\ell_1)}$ $\epsilon = 12$	$\mathcal{A}_{\text{adv}}^{(U)}$
High Complexity AT with ℓ_∞ perturbations					
PGD	**84.6**	**48.8**	62.3	15.0	15.0
+SNAP	83.0	44.8	**68.6**	**40.1**	**35.6**
TRADES	**82.1**	**50.2**	59.6	19.8	19.7
+SNAP	80.9	45.2	**66.9**	**46.6**	**41.2**
Low Complexity AT with ℓ_∞ perturbations					
FreeAdv	81.7	**46.1**	59	15.0	15.0
+SNAP	**83.5**	39.7	**66.2**	**34.3**	**29.6**
FastAdv	**85.7**	**46.2**	60.0	13.2	13.2
+SNAP	84.2	40.4	**67.9**	**36.6**	**30.8**

in PGD AT and the noise distribution in SNAP. For instance, SNAP[U] enhances $\mathcal{A}_{\text{adv}}^{(U)}$ by 11% with ℓ_∞-PGD AT while not achieving any improvement with ℓ_2-PGD AT. In fact, SNAP appears to be particularly suitable for ℓ_∞-AT, since it always improves $\mathcal{A}_{\text{adv}}^{(U)}$ by 11%-to-20.6% irrespective of the noise distribution.

Finally, of the three noise distributions, we find the Laplace distribution to be distinctly superior, achieving the highest $\mathcal{A}_{\text{adv}}^{(U)}$ (35.6% and 30.8%) due to a significant improvement in $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ for both ℓ_∞ and ℓ_2 PGD AT, respectively. The superiority of the Laplace distribution in achieving high $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ stems from its heavier tail compared to the Gaussian and Uniform distributions with the same variance. Shaped Laplace noise generates the highest fraction of extreme values in a given noise sample. Hence, it is more effective in improving accuracy against ℓ_1-bounded attacks, which are the strongest when perturbing few pixels by a large magnitude [107, 106]. We discuss this further in the Sec. 7.5.3. Henceforth, unless otherwise mentioned, we choose Laplace noise for SNAP and ℓ_∞ perturbations for BASE() AT as the default setting since it achieves the highest $\mathcal{A}_{\text{adv}}^{(U)}$.

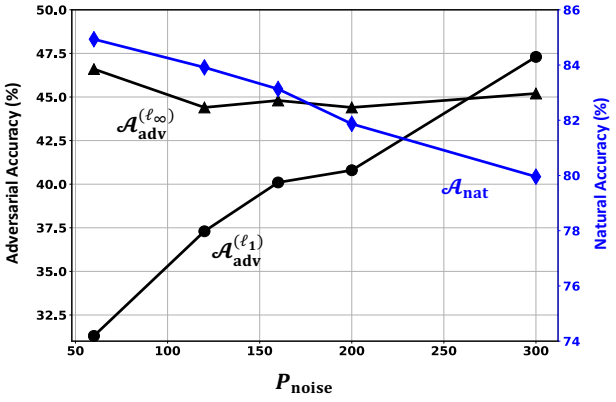


Figure 7.3: ResNet-18 CIFAR-10 results: adversarial accuracy $\mathcal{A}_{\text{adv}}^{(\ell_1)}$, $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$, and natural accuracy $\mathcal{A}_{\text{nat}}$ vs. total noise power P_{noise} for PGD+SNAP.

7.3.3 Impact of P_{noise}

Next, we explore the impact of the SNAP hyperparameter P_{noise}, which constrains the average squared ℓ_2 norm of the noise vector $\mathbf{n}$. It enables to trade between adversarial and natural accuracy.

Figure 7.3 shows that, as P_{noise} increases, $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ improves from 31% to 47%, accompanied by a graceful (5%) drop in $\mathcal{A}_{\text{nat}}$ and a small drop of 2% in $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ that stabilizes to $\approx 45\%$. These results show: (1) SNAP preserves the impact of ℓ_∞ perturbations which is not surprising since PGD AT [13] explicitly includes those, and (2) P_{noise} provides an explicit knob to control the $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}$ trade-off. Henceforth, we choose P_{noise} values that incur $< 1.5\%$ drop in $\mathcal{A}_{\text{nat}}$ for all SNAP+AT experiments.

7.3.4 SNAP Augmented SOTA AT Frameworks

Table 7.2 shows the effectiveness of SNAP for four SOTA AT frameworks: high complexity frameworks, such as PGD [13], TRADES [22], and low complexity frameworks such as FreeAdv [43], FastAdv [44]. All are trained against ℓ_∞ attacks with $\epsilon = 0.031$. As expected, while they achieve high $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$, their $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ and $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ are lower.

For high-complexity AT, SNAP enhances $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ and $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ by $\sim 6\%$ and $\sim 25\%$, respectively, while incurring only a drop of $\sim 5\%$ in $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$. Thus overall, SNAP improves robustness ($\mathcal{A}_{\text{adv}}^{(U)}$) by $\sim 20\%$ against the *union* of

the three perturbation models. Note that this robustness improvement comes at only a $\sim 1\%$ drop in $\mathcal{A}_{\text{nat}}$ (see Table 7.2). For low-complexity ATs, SNAP improvements in union robustness ($\mathcal{A}_{\text{adv}}^{(U)}$) are also significant ($\sim 15\%$). Again, presence of SNAP improves $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ and $\mathcal{A}_{\text{adv}}^{(\ell_1)}$. This time the drop in $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ is $\sim 7\%$. We believe this is due to the fact that these frameworks employ weaker single-step attacks during training. Note that in the case of FreeAdv+SNAP, we actually observe a $\sim 2\%$ *increase* in $\mathcal{A}_{\text{nat}}$, a trend we also observe in the ImageNet experiments described later.

7.3.5 Robustness vs. Training Complexity

Next we quantify adversarial robustness vs. training time trade-offs. Table 7.3 shows that SNAP augmentation of single-attack AT frameworks achieves the highest $\mathcal{A}_{\text{adv}}^{(U)}$, when training time is constrained to 12 hours (sets **B**, **C**, **D**, and **E**).

For instance, TRADES+SNAP achieves a 4% higher $\mathcal{A}_{\text{adv}}^{(U)}(= 41\%)$ than MSD-20 with 2 hours *lower* training time (Set **B** in Table 7.3). Similarly, PGD+SNAP achieves a 2% higher $\mathcal{A}_{\text{adv}}^{(U)}$ than MSD-10 while having a similar training time (Set **C**). Note that both PGD and TRADES here use 100 training epochs with standard step learning rate (LR) schedule, while MSD frameworks employ a cyclic learning rate schedule to achieve superconvergence in 50 epochs.

In Set **D**, *following* Maini *et al.* [107], we employ a cyclic learning rate schedule for PGD, TRADES, as well as for PGD+SNAP and TRADES+SNAP to achieve convergence in 50 epochs. Improvements in $\mathcal{A}_{\text{adv}}^{(U)}$ for PGD+SNAP and TRADES+SNAP are similar to those in Sets **B** and **C**. Most notably, PGD+SNAP with cyclic learning rate achieves $\sim$ 20% and 11.5% *higher* $\mathcal{A}_{\text{adv}}^{(U)}$ than MSD-5 and AVG-2, respectively, while having a similar training time ($\sim$ 3 hours). Set **E** augments the data from Table 7.2 with training times. FastAdv+SNAP and FreeAdv+SNAP achieve a high $\mathcal{A}_{\text{adv}}^{(U)} \sim 30\%$, while preserving the training efficiency of both FastAdv and FreeAdv. Notably, FastAdv+SNAP achieves 18% higher $\mathcal{A}_{\text{adv}}^{(U)}$ than MSD-5, while being $\sim 2.7\times$ more efficient to train.

Table 7.3: CIFAR-10 results for comparing adversarial accuracy $\mathcal{A}_{\text{adv}}^{(U)}$ vs. training time (on single NVIDIA P100 GPU) for different AT frameworks and the improvements by introducing proposed SNAP technique. All frameworks except PAT [194] (which employs ResNet-50) employ ResNet-18.

Method	LR schedule	Epochs	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(U)}$	Total time (minutes)
Set A: Total Time $\geq$ 12 Hrs					
AVG 50 Step [106]	cyclic	50	84.8	40.4	4217
AVG 20 Step [106]	cyclic	50	**85.6**	40.4	1834
AVG 10 Step [106]	cyclic	50	**86.7**	38.9	**956**
PAT [194]	step	100	82.4	36.6	1364
MSD 50 Step [107]	cyclic	50	81.7	**47.0**	1693
MSD 30 Step [107]	cyclic	50	82.4	**44.9**	**978**
Set B: 8 Hrs $<$ Total Time $<$ 12 Hrs					
AVG 5 Step [106]	cyclic	50	**87.8**	33.7	489
MSD 20 Step [107]	cyclic	50	83.0	37.3	690
TRADES [22]	step	100	82.0	19.7	**516**
TRADES+SNAP	step	100	80.9	**41.2**	566
Set C: 5 Hrs $<$ Total Time $<$ 8 Hrs					
MSD 10 Step [107]	cyclic	50	83.6	33.3	**342**
PGD [13]	step	100	**84.6**	15.0	354
PGD+SNAP	step	100	83.0	**35.6**	403
Set D: 2 Hrs $<$ Total Time $<$ 5 Hrs					
AVG 2 Step [106]	cyclic	50	**88.4**	22.0	232
MSD 5 Step [107]	cyclic	50	**84.0**	12.6	**185**
PGD [13]	cyclic	50	82.8	15.7	**177**
TRADES [22]	cyclic	50	80.0	21.4	258
PGD+SNAP	cyclic	50	82.3	**33.5**	199
TRADES+SNAP	cyclic	50	78.8	**40.8**	280
Set E: Total Time $<$ 2 Hrs					
FreeAdv [43]	step	200	81.7	15.0	**66**
FastAdv [44]	cyclic	50	**85.7**	13.2	**47**
FreeAdv+SNAP	step	200	83.5	**29.6**	88
FastAdv+SNAP	cyclic	50	**84.2**	**30.8**	69

Table 7.4: ImageNet results: Iso-hyperparameter introduction of SNAP yields $\sim 20\%$ improvement in adversarial accuracy ($\mathcal{A}_{\text{adv}}^{(U)}$) with modest impact on training time for ResNet-50 and ResNet-101.

Training	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ $\epsilon = 2/255$	$\mathcal{A}_{\text{adv}}^{(\ell_2)}$ $\epsilon = 2.0$	$\mathcal{A}_{\text{adv}}^{(\ell_1)}$ $\epsilon = 72.0$	$\mathcal{A}_{\text{adv}}^{(U)}$	Total time (minutes)
ResNet-50						
FreeAdv [43]	61.7	**47.8**	19.9	14.8	12.6	**3590**
FreeAdv+SNAP	**66.8**	46.1	**37.8**	**37.4**	**32.4**	3756
ResNet-101						
FreeAdv [43]	65.4	**51.8**	22.8	18.8	16.1	**5678**
FreeAdv+SNAP	**69.7**	50.3	**41.1**	**40.2**	**35.4**	5904

7.3.6 ImageNet Results

Thanks to SNAP's low computational overhead combined with FreeAdv's fast training time, we are for the first time able to report adversarial accuracy of ResNet-50 and ResNet-101 against the union of $(\ell_\infty, \ell_2, \ell_1)$ attacks on ImageNet.

We closely follow the evaluation setup of Shafahi *et al.* [43]. Specifically, we use a 100-step PGD attack, one of the strongest adversaries considered by Shafahi *et al.* [43], and evaluate on the entire test set. We first reproduce FreeAdv [43] results using the *same* hyperparameters and then introduce SNAP.

In order to clearly demonstrate the contrast between robustness to different perturbation models, we evaluate with $\epsilon = (2/255, 2.0, 72.0)$ for $(\ell_\infty, \ell_2, \ell_1)$ attacks, respectively.[2] As shown in Table 7.4, FreeAdv achieves a high $\mathcal{A}_{\text{adv}}^{(\ell_\infty)} = 47.8\%$ with ResNet-50, but a lower $\mathcal{A}_{\text{adv}}^{(\ell_2)} = 20\%$ and $\mathcal{A}_{\text{adv}}^{(\ell_1)} = 15\%$, and consequently, a low $\mathcal{A}_{\text{adv}}^{(U)}$ of 12.6% against the union of the perturbations. In contrast, FreeAdv+SNAP improves $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ and $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ by 17% and 22%, respectively, accompanied by a 5% improvement in $\mathcal{A}_{\text{nat}}$ and a small 2% loss in $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$. This results in an overall robustness improvement of 20% against the union of the perturbation models, setting a first benchmark for ResNet-50 on ImageNet. Upon increasing the network to ResNet-101, both natural and adversarial accuracies improve by $\approx 4\%$ for FreeAdv, a trend also observed by Shafahi *et al.* [43]. SNAP further improves FreeAdv's results for $\mathcal{A}_{\text{nat}}$ and $\mathcal{A}_{\text{adv}}^{(U)}$ by 4.3% and 19.3%.

[2]Note that ℓ_2 and ℓ_1 norms of PGD perturbation with ℓ_∞ norm of 2/255 can be as large as ~ 3.0 and ~ 1100 for images of size $224 \times 224 \times 3$.

Table 7.5: ResNet-18 SVHN results showing the impact of SNAP augmentation of ℓ_∞-PGD [13] AT frameworks. Adding SNAP improves $\mathcal{A}_{\text{adv}}^{(U)}$ by $\sim 30\%$ while having only a small impact on $\mathcal{A}_{\text{nat}}$ and $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$.

Method	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ $\epsilon = 0.03$	$\mathcal{A}_{\text{adv}}^{(\ell_2)}$ $\epsilon = 0.5$	$\mathcal{A}_{\text{adv}}^{(\ell_1)}$ $\epsilon = 8$	$\mathcal{A}_{\text{adv}}^{(U)}$
PGD	**89.9**	**45.3**	34.9	4.8	4.8
PGD+SNAP	89.3	44.0	**67.4**	**48.3**	**36.3**

Table 7.6: ResNet-18 CIFAR-10 results showing a comparison between MNG [197] and PGD+SNAP (from Table 7.2). All MNG numbers are exactly as reported in their paper. We reevaluate PGD+SNAP with our PGD attacks using the new ϵ values used by [197]. PGD+SNAP achieves 3%, 2%, 4.5% higher $\mathcal{A}_{\text{nat}}$, $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$, $\mathcal{A}_{\text{adv}}^{(\ell_1)}$, respectively, while being at least $\sim 40\%$ faster in terms of epoch time. †: Note that MNG time is measured on NVIDIA GeForce RTX 2080Ti (by [197]), while PGD+SNAP is measured on NVIDIA Tesla P100. An RTX 2080Ti has *20% more* CUDA cores than a Tesla P100.

Method	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ $\epsilon = 0.03$	$\mathcal{A}_{\text{adv}}^{(\ell_2)}$ $\epsilon = 0.31$	$\mathcal{A}_{\text{adv}}^{(\ell_1)}$ $\epsilon = 8$	Time per Epoch (seconds)
MNG [197]	79.8	43.9	**75.8**	53.8	$354^\dagger$
PGD+SNAP	**83.1**	**45.9**	74.1	**58.3**	**240**

7.3.7 SVHN Results

Table 7.5 shows PGD and PGD+SNAP results on SVHN data. We train both PGD and PGD+SNAP models for 100 epochs using a piece-wise LR schedule. We start with an initial LR of 0.01 and decay it once at the 95th epoch.

In Table 7.5, we observe a trend that is similar to our observations for CIFAR-10 and ImageNet results. In particular, for SVHN, SNAP turns out to be even more effective, with $\sim 30\%$ improvement in $\mathcal{A}_{\text{adv}}^{(U)}$ while almost preserving both $\mathcal{A}_{\text{nat}}$ and $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$.

7.3.8 Comparison with Madaan *et al.* [197]

The meta-noise generator (MNG) [197] employs a multi-layer deep net to generate noise samples during AT. Importantly, MNG still employs multiple attacks during training, but samples only one of the attacks randomly at a time to reduce the training cost.

Table 7.7: ResNet-18 CIFAR-10 results showing SNAP's impact on the prediction complexity, where N_0 denotes the number of noise samples employed to estimate $\mathbb{E}[\cdot]$ in Eq. (7.2). We find that for mere accuracy estimation, even a single forward pass ($N_0 = 1$) suffices. $\pm$xx denotes the standard deviation over 10 independent test runs.

Method	$\mathcal{A}_{\text{nat}}$ (%)
TRADES	81.7
TRADES+SNAP	
$N_0 = 1$	80.1±0.22
$N_0 = 2$	80.3±0.14
$N_0 = 4$	80.7±0.12
$N_0 = 8$	80.9±0.10
$N_0 = 16$	80.9±0.08

However, they have yet to release their code or pretrained models even though their work was posted on arXiv a year ago. Absence of public codes from Madaan *et al*. [197] makes it difficult to clearly compare with their work, especially in terms of training time. Nonetheless, in this section, our goal is to ensure that the comparison is fair. Table 7.6 reports natural and adversarial accuracy of MNG against $(\ell_\infty, \ell_2, \ell_1)$ attacks as reported by Madaan *et al*. [197]. We find that PGD+SNAP achieves 3%, 2%, 4.5% higher $\mathcal{A}_{\text{nat}}$, $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$, and $\mathcal{A}_{\text{adv}}^{(\ell_1)}$, respectively. Note that Madaan *et al*. [197] evaluate $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ and $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ against PGD-50 attacks, whereas here we employ PGD-100 attacks and, following their protocol, evaluate on the entire CIFAR-10 dataset with a single restart. Furthermore, epoch time for PGD+SNAP is 1.4× smaller than that of MNG [197] even though MNG time was measured on a more recent NVIDIA RTX 2080Ti, which has 20% more CUDA cores than the Tesla P100 GPU that we used for PGD+SNAP.

Importantly, a key advantage of SNAP is its scalability. We are able to report robust ResNet-50 and ResNet-101 networks on ImageNet (Table 7.4), whereas Madaan *et al*. [197] report results only up to 64×64 TinyImageNet.

7.3.9 Impact of SNAP on Prediction Complexity

While SNAP augmentation has a modest impact on the training time (Table 7.3), here we check whether it could *potentially* increase the model prediction complexity due to the need to estimate the expectation $\mathbb{E}[\cdot]$ in Eq. (7.2).

As expected, by increasing N_0, the deviation of the $\mathcal{A}_{\text{nat}}$ estimate reduces (see Table 7.7). However, we find that for accuracy estimation, a single forward pass ($N_0 = 1$) suffices. Specifically, an $\mathcal{A}_{\text{nat}}$ estimate with $N_0 = 1$ is within 1% of the $\mathcal{A}_{\text{nat}}$ estimate with $N_0 = 16$. Furthermore, even with $N_0 = 1$, the standard deviation of $\mathcal{A}_{\text{nat}}$ is as low as $\sim 0.2\%$. Thus, the impact of SNAP on prediction complexity can be very small.

7.4 Robustness Stress Tests

We conduct robustness stress tests to confirm that the benefits of SNAP are sustained for a range of attack norm-bounds, larger number of attack steps, and even for "gradient-free" attacks. For these experiments, we consider networks trained using TRADES and TRADES+SNAP (rows in Table 7.2), since they achieve the highest $\mathcal{A}_{\text{adv}}^{(U)}$ among the four SOTA AT frameworks.

7.4.1 Sweeping Norm-bounds and Number of Attack Steps

We sweep the number of PGD attack steps (K) and norm-bounds (ϵ) for all three perturbations ($\ell_\infty, \ell_2, \ell_1$) to confirm that the robustness gains from SNAP are achieved for a wider range of attack norm bounds, and are sustained even after increasing attack steps.

Figure 7.4(a)-(c) validates the Table 7.2 conclusion that TRADES+SNAP achieves large gains ($\sim 20\%$) in $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ and $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ with a small ($\sim 4\%$) drop in $\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$. Furthermore, this conclusion holds for a large range of ϵ values for all three perturbations. Additionally, the gain in $\mathcal{A}_{\text{adv}}^{(\ell_2)}$ due to SNAP at $\epsilon = 1.2$ is greater than the one reported in Table 7.2 for $\epsilon = 0.5$.

Now we increase the attack steps K to 500 and observe the impact on adversarial accuracy against ($\ell_\infty, \ell_2, \ell_1$) perturbations in Fig. 7.4(d,e,f), respectively. In all cases, we observe hardly any change of the adversarial accuracy beyond $K = 100$. Hence, we have chosen $K = 100$ for all our experiments.

Recall we employ 10 random restarts as recommended by [107] for *all* our adversarial accuracy evaluations on CIFAR-10 data.

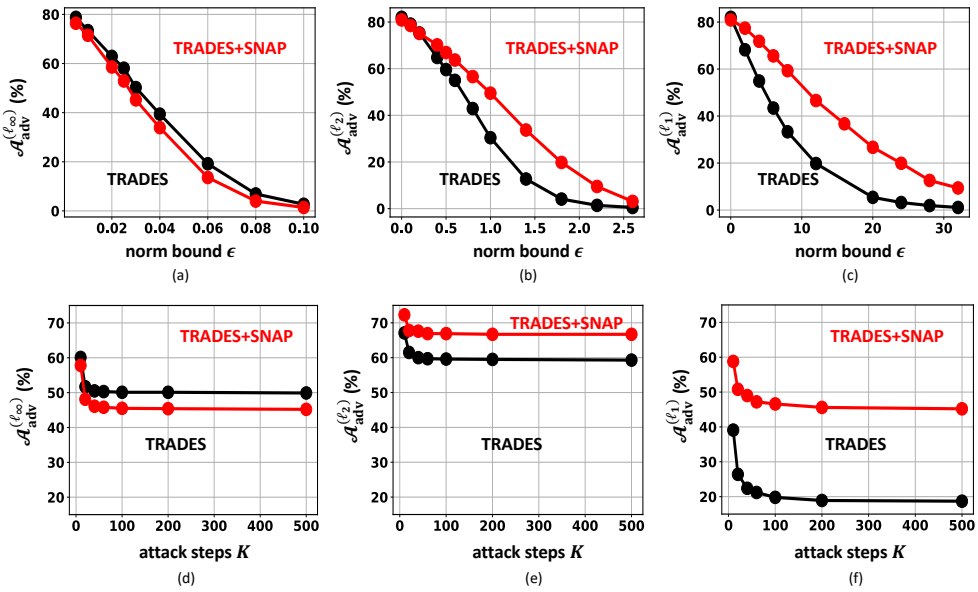


Figure 7.4: ResNet-18 CIFAR-10 results: Adversarial accuracy vs. norm bound ϵ for: (a) ℓ_∞, (b) ℓ_2, (c) ℓ_1 PGD-100 attack. Adversarial accuracy vs. attack steps K for (d) ℓ_∞ ($\epsilon = 0.031$), (e) ℓ_2 ($\epsilon = 0.5$), (f) ℓ_1 ($\epsilon = 12$) PGD-100 attacks.

7.4.2 Evaluating Robustness Against New Attacks

We evaluate adversarial accuracy against the recent DDN [31], Boundary [32], and Square [33] attacks. The DDN attack was shown to be one of the SOTA gradient-based attacks, while boundary attack is one of the strongest "gradient-free" attacks. Of all the attacks considered in [107], PGD turns out to be the strongest for ℓ_∞ and ℓ_1 perturbations. Hence, in this section, we evaluate against ℓ_2 norm-bounded DDN, boundary, and Square attacks.

Following [107], we use the FoolBox [198] implementation of the boundary attack, which uses 25 trials per iteration. For the DDN attack, we use 100 attack steps with appropriate logit averaging for $N_0 = 8$ noise samples *before* computing the gradient in each step (similar to our PGD attack implementations). As mentioned in the Sec. 7.3.1, it eliminates any gradient obfuscation due to the presence of noise.

Table 7.8 shows that SNAP improves adversarial accuracy against the DDN attack by $\sim 6\%$. This is similar to improvements seen against ℓ_2-PGD attack in Table 7.2. Similarly, TRADES+SNAP achieves 3.5% (4.5%) higher adversarial accuracy than TRADES against the Boundary [32] (Square [33])

Table 7.8: ResNet-18 CIFAR-10 results showing natural accuracy (%) and adversarial accuracy (%) against ℓ_2 norm bounded DDN attack [31], boundary attack [32], and Square [33] for TRADES and TRADES+SNAP networks from Table 7.2.

	TRADES	**TRADES+SNAP**
Natural Accuracy	**82.1**	80.9
DDN [31] ($\epsilon = 0.5$)	59.7	**65.8**
Boundary [32] ($\epsilon = 0.5$)	63.5	**67.0**
Square [33] ($\epsilon = 0.5$)	68.2	**72.7**

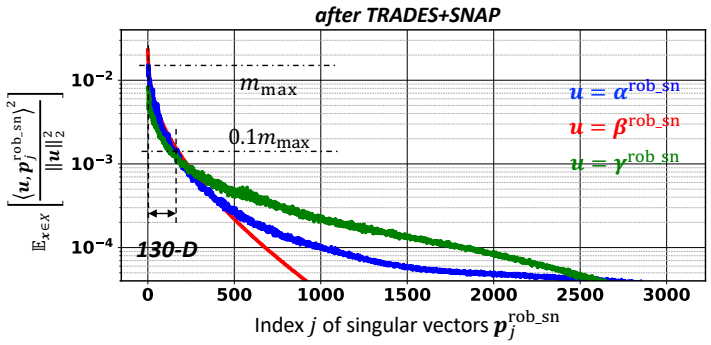


Figure 7.5: Normalized mean squared projections of three perturbation types on the singular vector basis $\mathcal{P}^{\kappa}$ of ℓ_2 perturbations of ResNet18 on CIFAR-10 after TRADES+SNAP training ($\kappa \equiv$ rob_sn). The singular vectors $\boldsymbol{p}_i^{\kappa}$ comprising $\mathcal{P}^{\kappa} = \{\boldsymbol{p}_1^{\kappa}, \ldots, \boldsymbol{p}_D^{\kappa}\}$ are ordered in descending order of their singular values.

attack.

7.5 Additional Investigations

7.5.1 Subspace Analysis of Adversarial Perturbations for TRADES+SNAP Model

In this section, we carry out a subspace analysis of adversarial perturbations (similar to Sec. 6.3 in Chapter 6) for TRADES+SNAP. We confirm that our hypothesis in Sec. 6.3 holds even after SNAP augmentation of TRADES. Following the same experimental setup and the notation from Sec. 6.3, we compute perturbations $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, and $\boldsymbol{\gamma}_i$ for each $\boldsymbol{x}_i \in X$ for ResNet-18 trained using TRADES+SNAP, *i.e.*, $\kappa \equiv$ rob_sn. We compute the singular vector

Table 7.9: ResNet-18 CIFAR-10 results showing the impact of noise shaping basis V for ℓ_∞-PGD [13] AT framework with SNAP. In this table, SNAP[G], SNAP[U], and SNAP[L] denote shaped noise augmentations with Gaussian, Uniform, and Laplace noise distributions, respectively, and U_{img} refers to the singular vector basis of the training images.

Method	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(\ell_\infty)}$ $\epsilon = 0.03$	$\mathcal{A}_{\text{adv}}^{(\ell_2)}$ $\epsilon = 0.5$	$\mathcal{A}_{\text{adv}}^{(\ell_1)}$ $\epsilon = 12$	$\mathcal{A}_{\text{adv}}^{(U)}$
PGD	84.6	48.8	62.3	15.0	15.0
Noise shaping basis $V = \mathbf{I}_{D\times D}$					
+SNAP[G]	80.7	**45.7**	66.9	34.6	31.9
+SNAP[U]	**85.1**	42.7	**66.7**	28.6	26.6
+SNAP[L]	83.0	44.8	**68.6**	**40.1**	**35.6**
Noise shaping basis $V = U_{\text{img}}$					
+SNAP[G]	81.7	**48.9**	67.5	**29.8**	**28.7**
+SNAP[U]	**82.0**	46.6	**67.8**	27.8	25.7
+SNAP[L]	81.7	46.8	65.9	28.5	27.4

basis $\mathcal{P}^\kappa$ for the set of ℓ_2 bounded perturbations $\Delta^\kappa = \{\boldsymbol{\beta}_1^\kappa, \ldots, \boldsymbol{\beta}_{|X|}^\kappa\}$. Figure 7.5 plots the normalized mean squared projections of the three types of perturbation vectors on the singular vector basis $\mathcal{P}^\kappa$ of a TRADES+SNAP trained ResNet-18. We find that the projections generally follow the same trend as those for a TRADES-trained network which are shown in Fig. 6.4(b). However, we also notice that after SNAP augmentation, the three perturbation types get squeezed into an even smaller 130-dimensional subspace, *i.e.*, projections are $< 10\%$ of the maximum projection value for all dimensions beyond the first 130 dimensions.

7.5.2 Impact of Noise Shaping in the Image Basis

Recall that, for all experiments till now, we chose the noise shaping basis $V = \mathbf{I}_{D\times D}$, *i.e.*, the noise was shaped and added in the standard basis in $\mathbb{R}^D$, where $\mathbf{I}_{D\times D}$ denotes the identity matrix (see Eq. (7.1) and Algorithm 1).

In this section, we explore the shaped noise augmentation in the *image basis*, *i.e.*, singular vector basis of the training set images. Specifically, we choose $V = U_{\text{img}} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D]$, where U_{img} denotes the singular vector basis of the images in the training set. Thus, the sampled noise vector $\mathbf{n}_0$ (see Eq. (7.1)) is scaled by direction-wise standard deviation matrix Σ and

rotated by U_{img} before being added to the input image $\boldsymbol{x}$.

The rationale for choosing $V = U_{\text{img}}$ is as follows: Recent works [190, 43, 199] have demonstrated the generative behavior of adversarial perturbations of networks trained with single-attack AT, *i.e.*, adversarial perturbations of robust networks exhibit semantics similar to the input images. Thus, the perturbation basis of the robust networks trained with single-attack AT seems to be aligned with the image basis (see Chapter 6).

We repeat the experiments in Table 7.1 while keeping all the settings *identical* except for choosing $V = U_{\text{img}}$ instead of $V = \mathbf{I}_{D\times D}$. Table 7.9 shows the results. The first three rows correspond to $V = \mathbf{I}_{D\times D}$ and are reproduced from Table 7.1. Note that, in order to preserve $\mathcal{A}_{\text{nat}} > 81\%$, we need to reduce $P_{\text{noise}} = 60$ when $V = U_{\text{img}}$, since the noise is now pixel-wise correlated.

In Table 7.9, we notice that $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ is significantly reduced when $V = U_{\text{img}}$ as compared to the case $V = \mathbf{I}_{D\times D}$. More interestingly, all three types of noise distributions result in similar values for $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ when $V = U_{\text{img}}$. We discuss this phenomenon in the next section, *i.e.*, Sec. 7.5.3.

Table 7.9 shows that the orientation of a noise vector is as important as its distribution. The simpler choice of $V = \mathbf{I}_{D\times D}$ turns out to be more effective.

7.5.3 Understanding the effectiveness of SNAP[L] for ℓ_∞ AT

In this subsection, we conduct additional studies to further understand the following two observations in SNAP: (i) shaped Laplace noise is particularly effective (Table 7.1), and (ii) rotating noise vectors ($V = U_{\text{img}}$) reduces their effectiveness (Table 7.9). We study the properties of the noise vector $\mathbf{n}$ for different noise distributions.

We conjecture that the Laplace distribution is most effective because of its heavier tail compared to Gaussian and Uniform distributions of the same variance. A long-tailed distribution will generate more large magnitude elements in a vector drawn from it and hence is more effective in emulating a strong ℓ_1-norm bounded perturbation. Furthermore, the standard (unrotated) basis preserves this unique attribute of samples drawn from such distributions.

This conjecture is validated by Fig. 7.6(a) which shows that noise samples drawn from the Laplace distribution in the standard basis have the highest

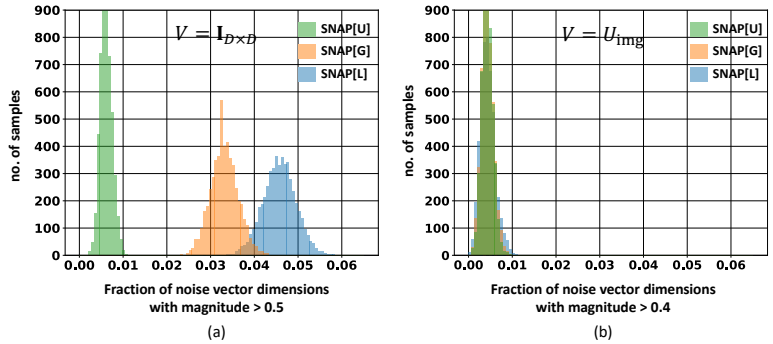


Figure 7.6: ResNet18 CIFAR-10 results: histograms of the fraction of noise vector dimensions with magnitude (a) > 0.5 when $V = \mathbf{I}_{D\times D}$, and (b) > 0.4 when $V = U_{\text{img}}$. Histograms are plotted for 5000 random noise samples $\mathbf{n}$. The three shaped noise distributions are from the corresponding networks in Table 7.9.

average number of dimensions with large (> 0.5) magnitudes, followed by Gaussian and Uniform distributions. This correlates well with the results in Table 7.1 and Table 7.9 (first three rows), in that $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ is the highest for Laplace followed by those for Gaussian and Uniform. Additionally, the use of $V = U_{\text{img}}$ dissolves this distinction between the three distributions as shown in Fig. 7.6(b) which explains the similar (and lower) $\mathcal{A}_{\text{adv}}^{(\ell_1)}$ values for all three distributions in Table 7.9.

Thus, we confirm that the type of noise plays an important role in robustifying single-attack ℓ_∞ AT frameworks to the union of multiple perturbation models. Specifically, the noise vectors with higher fraction of noise dimensions with larger magnitudes are better at complementing ℓ_∞ AT frameworks.

7.5.4 Evaluating Common Corruptions and Functional Attack

In this section, we check if there are any other downsides of SNAP when it improves robustness against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations. In particular, we check if SNAP improvements are achieved at the cost of a drop in accuracy against common corruptions [10] or functional adversarial attacks [200].

We use corrupted images provided by [10] to estimate accuracy in the presence of common corruptions ($\mathcal{A}_{\text{cc}}$). We average the accuracy numbers across different corruption strengths and types. Also, we use the ReColorAdv

Table 7.10: ResNet-18 CIFAR-10 results showing natural accuracy $\mathcal{A}_{\text{nat}}$, adversarial accuracy $\mathcal{A}_{\text{adv}}^{(U)}$ against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations, accuracy $\mathcal{A}_{\text{cc}}$ in the presence of common corruptions [10], and adversarial accuracy $\mathcal{A}_{\text{adv}}^{(\text{f})}$ against a functional adversarial attack ReColorAdv [200]. All accuracy numbers are in %. In this table, $\mathbf{I}_{D\times D}$ denotes D-dimensional identity matrix, while U_{img} denotes singular vector basis of the training images. We find that SNAP augmentations of ℓ_∞-PGD significantly ($\approx 20\%$) improve $\mathcal{A}_{\text{adv}}^{(U)}$ while preserving both $\mathcal{A}_{\text{cc}}$ and $\mathcal{A}_{\text{adv}}^{(\text{f})}$.

Method	$\mathcal{A}_{\text{nat}}$	$\mathcal{A}_{\text{adv}}^{(U)}$	$\mathcal{A}_{\text{cc}}$	$\mathcal{A}_{\text{adv}}^{(\text{f})}$ ReColorAdv
Vanilla	**94.5**	0.0	72.0	0.9
ℓ_∞-PGD	84.6	15.0	**75.6**	53.5
Noise shaping basis $V = \mathbf{I}_{D\times D}$				
+SNAP[G]	80.7	31.9	72.8	**55.1**
+SNAP[U]	**85.1**	26.6	75.0	46.9
+SNAP[L]	83.0	**35.6**	75.3	51.3
Noise shaping basis $V = U_{\text{img}}$				
+SNAP[G]	81.7	28.7	73.6	54.5
+SNAP[U]	82.0	25.7	73.1	54.0
+SNAP[L]	81.7	27.4	73.4	**55.3**

setup of [194] to estimate accuracy against functional adversarial attacks ($\mathcal{A}_{\text{adv}}^{(\text{f})}$). We also make it *adaptive* to our defense framework via appropriate noise averaging (similar to our adaptive PGD attacks [195]) to eliminate any gradient obfuscations. As observed in Table 7.10, SNAP augmentations of PGD AT generally preserve both $\mathcal{A}_{\text{cc}}$ and $\mathcal{A}_{\text{adv}}^{(\text{f})}$. In particular, 20.6% improvement in $\mathcal{A}_{\text{adv}}^{(U)}$ via PGD+SNAP[L] (with $V = \mathbf{I}_{D\times D}$) is accompanied with the same $\mathcal{A}_{\text{cc}}$ and only a 2.2% lower $\mathcal{A}_{\text{adv}}^{(\text{f})}$ (= 51.3%) compared to PGD AT. In contrast, vanilla training achieves an $\mathcal{A}_{\text{adv}}^{(\text{f})}$ of only 0.9%. Even with $V = U_{\text{img}}$, PGD+SNAP[L] achieves a 1.8% higher $\mathcal{A}_{\text{adv}}^{(\text{f})}$ along with a 12.4% improvement in $\mathcal{A}_{\text{adv}}^{(U)}$. Note that all $\mathcal{A}_{\text{adv}}^{(U)}$ numbers are idential to the ones reported in Sec. 7.5.2.

We conclude that SNAP augmentation of PGD AT improves $\mathcal{A}_{\text{adv}}^{(U)}$ by up to 20% while preserving its robustness against common corruptions and functional adversarial attacks. Thus, SNAP expands the capabilities of ℓ_∞ AT frameworks without any significant downside. However, further work is required to improve robustness to a larger class adversarial attacks, such as rotation [201], texture [202], etc., simultaneously.

7.6 Relationship between SNAP and Randomized Smoothing

In this section, we discuss a relationship between SNAP and Randomized Smoothing (RS) [196, 195], another popular noise augmentation approach that employs isotropic Gaussian noise in order to obtain certification bounds on adversarial robustness. Specifically, we show that SNAP[G] is closely related to RS, *i.e.*, SNAP[G] can be viewed as RS with shaped Gaussian noise. In Sec. 7.6.1, we first extend the generalize the bound derived in [196] (for isotropic Gaussian noise) and use it to justify our noise shaping approach in Eq. (7.3). We further provide geometric intuition (in Sec. 7.6.2) demonstrating the why shaped noise inherently leads to improved robustness compared to isotropic noise for Gaussian distribution. At the end, in Sec. 7.6.3, we demonstrate that the noise shaping in SNAP enables significantly better $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ trade-off compared due to RS.

7.6.1 Theoretical Justification for Noise Shaping

We define a smoothed classifier [196] as follows:

Definition 7.1. Given a classifier $f : \mathbb{R}^D \to \mathcal{Y}$, for a given input $\boldsymbol{x}$, a smoothed classifier g is defined as:

$$g(\boldsymbol{x}) = \arg\max_{c \in \mathcal{Y}} \Pr\{f(\boldsymbol{x} + \mathbf{n}) = c\}$$

where $\mathbf{n} \sim \mathcal{N}(0, C_\sigma)$, $C_\sigma \in \mathbb{R}^{D \times D}$ denotes the covariance matrix, and $\mathcal{Y}$ denotes the set of discrete class labels.

Notice that SNAPnet decision rule (see Eq. (7.2)) is same as smoothed classifier above when (i) SNAP employs Gaussian distributed noise (SNAP[G]), and (ii) $C_\sigma = \text{Diag}[\sigma_1, \ldots, \sigma_D]$, where dimensionwise noise standard deviation are shaped according to Eq. (7.3). In [196], a certification bound for a smoothed classifier was obtained for the isotropic case, *i.e.*, when $C_\sigma = \sigma^2 \mathbf{I}_{D \times D}$. Here, we first generalize this bound for the anisotropic case where C_σ is an arbitrary covariance matrix as stated in Theorem 7.1. We then employ Theorem 7.1 to obtain a theoretical justification for noise shaping (as per Eq. (7.3)) with Gaussian noise distribution.

Theorem 7.1. *Let $f : \mathbb{R}^D \to \mathcal{Y}$ be an arbitrary function either deterministic or random, $\mathbf{n} \sim \mathcal{N}(0, C_\sigma)$ with a covariance matrix $C_\sigma \in \mathbb{R}^{D\times D}$, and $g(\boldsymbol{x}) = \arg\max_c \Pr(f(\boldsymbol{x}+\mathbf{n}) = c)$. For a specific $\boldsymbol{x} \in \mathbb{R}^D$, if $\exists\ c_A \in \mathcal{Y}$ and $\rho_A, \rho_B \in [0,1]$ such that:*

$$\Pr\{f(\boldsymbol{x}+\mathbf{n}) = c_A\} \geq \rho_A \geq \rho_B \geq \max_{c\neq c_A} \Pr\{f(\boldsymbol{x}+\mathbf{n}) = c\} \tag{7.4}$$

then, for any $\boldsymbol{\xi} \in \mathbb{R}^D$, we obtain $g(\boldsymbol{x}+\boldsymbol{\xi}) = c_A$ if

$$\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}} < \frac{\boldsymbol{\Phi}^{-1}(\rho_A) - \boldsymbol{\Phi}^{-1}(\rho_B)}{2} \tag{7.5}$$

where $\boldsymbol{\Phi}$ denotes the cumulative density function of a standard 1-D Gaussian distribution $\mathcal{N}(0,1)$.

Proof. The detailed proof provided in Appendix C follows the sequence of arguments provided in [196] for the isotropic case. □

Since Theorem 7.1 holds for an arbitrary Gaussian distribution $\mathbf{n} \sim \mathcal{N}(0, C_\sigma)$, we treat C_σ as a design variable and ask the following question: *what is a good choice for C_σ in order to enhance robustness under a fixed total power P_{noise} constraint?* Equation (7.5) provides a hint. It suggests that an ideal choice for C_σ is one that minimizes the LHS of Eq. (7.5) when $\boldsymbol{\xi} = \boldsymbol{\eta}$, *i.e.*, $\boldsymbol{\xi}$ is an adversarial perturbation, *e.g.*, a PGD adversarial perturbation. More formally, we would like to choose C_σ as a solution to the following constrained optimization problem:

$$C_\sigma^* = \arg\min_{C_\sigma} \mathbb{E}_{\boldsymbol{x}\in X}\left[\boldsymbol{\eta}^T C_\sigma^{-1} \boldsymbol{\eta}\right] \quad \text{s.t.} \quad \mathrm{Tr}[C_\sigma] = P_{\text{noise}} \tag{7.6}$$

where $\mathrm{Tr}[C_\sigma]$ denotes the trace of matrix C_σ, and $\boldsymbol{\eta}$ being the PGD adversarial perturbation is a function of the input $\boldsymbol{x} \in X$.

Without loss of generality, we decompose $C_\sigma = U\Sigma U^T$ using an arbitrary orthonormal basis $U = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_D]$ of $\mathbb{R}^D$, where $\Sigma = \mathrm{Diag}[\sigma_1^2, \ldots, \sigma_D^2]$. Note: $P_{\text{noise}} = \sum_{j=1}^D \sigma_j^2 = \mathrm{Tr}[\Sigma]$. Employing this decomposition for C_σ, we express the objective function in Eq. (7.6) as:

$$\mathbb{E}_{\boldsymbol{x}\in X}\left[\boldsymbol{\eta}^T C_\sigma^{-1} \boldsymbol{\eta}\right] = \mathbb{E}_{\boldsymbol{x}\in X}\left[\boldsymbol{\eta}^T U \Sigma^{-1} U^T \boldsymbol{\eta}\right] = \sum_{j=1}^D \frac{\mathbb{E}_{\boldsymbol{x}\in X}\left(\langle \boldsymbol{\eta}, \boldsymbol{u}_j\rangle^2\right)}{\sigma_j^2} \tag{7.7}$$

For a specific orthonormal basis U, the optimal $\Sigma_U = \text{Diag}[\sigma^2_{\boldsymbol{u}_1}, \ldots, \sigma^2_{\boldsymbol{u}_D}]$ can be obtained as follows:

$$\sigma^2_{\boldsymbol{u}_j} \propto \sqrt{\mathbb{E}_{\boldsymbol{x}\in X}\big(\langle \boldsymbol{\eta}, \boldsymbol{u}_j\rangle^2\big)} \quad \forall \quad j \in \{1, \ldots, D\} \quad \text{s.t.} \quad \sum_{j=1}^{D} \sigma^2_{\boldsymbol{u}_j} = P_{\text{noise}} \quad (7.8)$$

which is indeed the exact noise variance allocation rule in SNAP (see lines 6-10 in Alg. 1 and Eq. (7.3)). Note that this rule suggests assigning variance in proportion to the root mean-squared projections of $\boldsymbol{\eta}$ in the directions $\boldsymbol{u}_j$. While the optimality of this rule is derived here only for Gaussian noise distribution, we find that this noise shaping rule also works well with other (Laplace, Uniform) distributions (as demonstrated in Sec. 7.3). In Sec. 7.6.2, we provide a geometric intuition for the role of such noise shaping in achieving higher robustness.

7.6.2 Geometric Intuition: Spherical vs. Shaped Gaussian Noise

In this section, we provide a geometric intuition (see Fig. 7.7) to demonstrate the effectiveness of noise shaping. For simplicity, we consider Gaussian noise augmented two-dimensional binary linear classifier for this analysis. We show that the classifier with SNAP[G] forces the decision boundary to be farther away from the input in order to achieve the same decision confidence compared to any other shape for augmented Gaussian noise.

Consider three binary (classes c_A and c_B) linear classifiers $f^\alpha : \mathbb{R}^2 \to \{c_A, c_B\}$ in Fig. 7.7, where $\alpha \in \{\text{wht}, \text{wrst}, \text{sn}\}$ identifies the noise distribution type under a fixed power constraint $P_{\text{noise}}(\sigma^2_{1(\alpha)} + \sigma^2_{2(\alpha)})$ defined as follows ($\hat{\boldsymbol{\eta}}$ is a unit vector in the dominant adversarial perturbation direction, and $\hat{\boldsymbol{\eta}}^\perp$ is its orthogonal direction):

1. $\alpha = \text{wht}$ (Fig. 7.7(a)): The classifier is augmented with white Gaussian noise $\mathbf{n}_{(\text{wht})}$ such that $P_{\text{noise}} = 2\sigma^2$ and $\sigma^2_{1(\text{wht})} = \sigma^2_{2(\text{wht})} = \sigma^2$.

2. $\alpha = \text{wrst}$ (Fig. 7.7(b)): The classifier is augmented with shaped noise $\mathbf{n}_{(\text{wrst})}$ such that the noise variance $\sigma^2_{1(\text{wrst})}$ ($\sigma^2_{2(\text{wrst})}$) along $\hat{\boldsymbol{\eta}}$ ($\hat{\boldsymbol{\eta}}^\perp$) is *decreased* (*increased*) and $P_{\text{noise}} = \sigma^2_{1(\text{wrst})} + \sigma^2_{2(\text{wrst})} = 2\sigma^2$. Note that this variance allocation is the *opposite* (worst) to the one in Eq. (7.3).

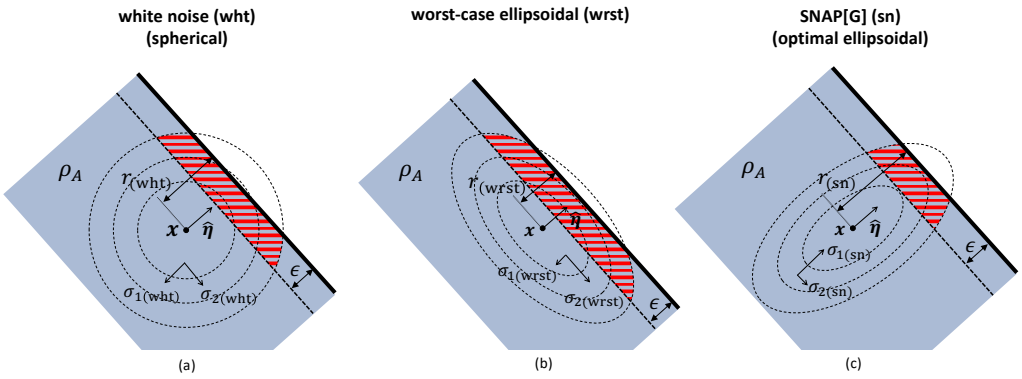


Figure 7.7: Geometric intuition regarding the impact of noise shaping on the robustness of binary linear classifiers f^{α} ($\alpha \in \{\text{wht}, \text{wrst}, \text{sn}\}$) augmented with noise $\mathbf{n}_{(\alpha)}$ of a fixed power $P_{\text{noise}} = (\sigma^2_{1(\alpha)} + \sigma^2_{2(\alpha)})$ when subject to the adversarial perturbation $\boldsymbol{\eta} = \epsilon\hat{\boldsymbol{\eta}}$ where $\hat{\boldsymbol{\eta}}$ is the unit vector in direction of the dominant adversarial perturbation for input $\boldsymbol{x}$: (a) white noise ($\alpha =$ wht) with $\sigma_{1(\text{wht})} = \sigma_{2(\text{wht})}$, (b) worst-case ellipsoidal noise ($\alpha =$ wrst) with $\sigma_{1(\text{wrst})} < \sigma_{2(\text{wrst})}$, and (c) SNAP[G] ($\alpha =$ sn) where $\sigma_{1(\text{sn})} > \sigma_{2(\text{sn})}$. Since the confidence $\Pr\{f^{\alpha}(\boldsymbol{x} + \mathbf{n}_{(\alpha)}) = c_A\} = \rho_{A(\alpha)} = \rho_A, \forall \alpha \in \{\text{wht}, \text{wrst}, \text{sn}\}$ and $\sigma_{1(\text{sn})} > \sigma_{1(\text{wht})} > \sigma_{1(\text{wrst})}$, it can be shown that the distance r from $\boldsymbol{x}$ to the decision boundary satisfies $r_{(\text{sn})} > r_{(\text{wht})} > r_{(\text{wrst})}$. Furthermore, it can also be shown that the confidence $\widehat{\rho}_{A(\alpha)}$ in the presence of an adversarial input $\boldsymbol{x} + \epsilon\hat{\boldsymbol{\eta}}$ satisfies $\widehat{\rho}_{A(\text{sn})} > \widehat{\rho}_{A(\text{wht})} > \widehat{\rho}_{A(\text{wrst})}$ as can be intuited from the striped red area.

3. $\alpha =$ en (Fig. 7.7(c)): The classifier is augmented with shaped (Gaussian) noise $\mathbf{n}_{(\text{sn})}$ such that the noise variance $\sigma^2_{1(\text{sn})}$ ($\sigma^2_{2(\text{sn})}$) along $\hat{\boldsymbol{\eta}}$ ($\hat{\boldsymbol{\eta}}^{\perp}$) is *increased* (*decreased*) and $P_{\text{noise}} = \sigma^2_{1(\text{sn})} + \sigma^2_{2(\text{sn})} = 2\sigma^2$. Note that this variance allocation is the one prescribed SNAP in Eq. (7.3).

If $(\boldsymbol{x}, c_A)$ is an (input, class label) pair and all three classifiers f^{α} achieve the same confidence level ρ_A in classifying $\boldsymbol{x}$ correctly, *i.e.*,

$$\Pr\{f^{\alpha}(\boldsymbol{x} + \mathbf{n}_{(\alpha)}) = c_A\} = \rho_A \quad \forall \quad \alpha \in \{\text{wht}, \text{wrst}, \text{sn}\} \tag{7.9}$$

then ρ_A corresponds to the mass of the probability density function of $\mathbf{n}^{\alpha}$ on the side of the decision boundary corresponding to the class label c_A as indicated in grey in Fig. 7.7. Therefore, the distance $r_{(\alpha)}$ of $\boldsymbol{x}$ to the decision boundary of f^{α} is given by $r_{(\alpha)} = \sigma_{1(\alpha)}\Phi^{-1}(\rho_A)$ where $\sigma^2_{1(\alpha)}$ is the variance of noise in the direction of $\hat{\boldsymbol{\eta}}$. Since $\sigma_{1(\text{en})} > \sigma_{1(\text{wht})} > \sigma_{1(\text{wrst})}$, therefore $r_{\text{sn}} > r_{\text{wht}} > r_{\text{wrst}}$, *i.e.*, SNAP[G] pushes the decision region further away

as compared to any other shaped noise augmentation, including white noise. Additionally, it can be shown that the confidence $\widehat{\rho}_{A(\alpha)}$ in the presence of an adversarial input $\boldsymbol{x} + \epsilon\hat{\boldsymbol{\eta}}$ is given by:

$$\widehat{\rho}_{A(\alpha)} = \Pr\{f^{\alpha}(\boldsymbol{x} + \epsilon\boldsymbol{\eta} + \mathbf{n}_{(\alpha)}) = c_A\} = \Phi\left(\Phi^{-1}(\rho_A) - \frac{\epsilon}{\sigma_{1(\alpha)}}\right) \tag{7.10}$$

and since $\sigma_{1(\text{sn})} > \sigma_{1(\text{wht})} > \sigma_{1(\text{wrst})}$ it follows that $\widehat{\rho}_{A(\text{sn})} > \widehat{\rho}_{A(\text{wht})} > \widehat{\rho}_{A(\text{wrst})}$. The striped red area in Fig. 7.7 provides a geometric intuition underlying this result. This shows that SNAP[G] provides the highest confidence for the adversarially perturbed input $\boldsymbol{x}+\epsilon\hat{\boldsymbol{\eta}}$ as compared to any other shaped Gaussian noise augmentation including white noise. Recall that all three classifiers f^{α} classify clean input $\boldsymbol{x}$ with equal confidence of ρ_A. Thus, the *reduction* in the classifier's confidence $(\rho_A - \widehat{\rho}_{A(\alpha)})$ due to adversarial perturbation $\epsilon\hat{\boldsymbol{\eta}}$ is the *minimum* for the SNAP[G] classifier, compared to a classifier with any other Gaussian noise augmentation. This indicates the superior robustness of SNAP[G].

7.6.3 Quantitative Comparison

In this subsection, we compare with two SOTA RS frameworks, namely, RandSmooth [196], and SmoothAdv [195]. They employ isotropic Gaussian noise. In Fig. 7.8(a), we find that PGD+SNAP[L] achieves a better $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ trade-off compared to both RandSmooth [196], and SmoothAdv [195]. Specifically, note that SmoothAdv [195] can also be viewed as isotropic Gaussian augmentation of ℓ_2-PGD AT. Importantly, PGD+SNAP[L] achieves a 12% higher $\mathcal{A}_{\text{adv}}^{(U)}$ for the same $\mathcal{A}_{\text{nat}}$. This demonstrates the efficacy of *shaped noise* in SNAP, which enhances the robustness to the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations.

In order to further quantify importance of *noise shaping*, we also compare ℓ_∞-PGD+SNAP[L] with ℓ_∞-PGD+Iso[L], a stronger baseline alternative consisting of *isotropic* Laplace noise augmentation, *i.e.*, *without any noise shaping*. Specifically, in Iso[L], the noise standard deviation is *identical* in each direction, *i.e.*, $\Sigma = \text{Diag}\left[\sqrt{\frac{P_{\text{noise}}}{D}}, \ldots, \sqrt{\frac{P_{\text{noise}}}{D}}\right]$. Note that such distributions have recently been explored for RS [203].

Figure 7.8(b) plots the $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ trade-off for PGD+SNAP (*red*

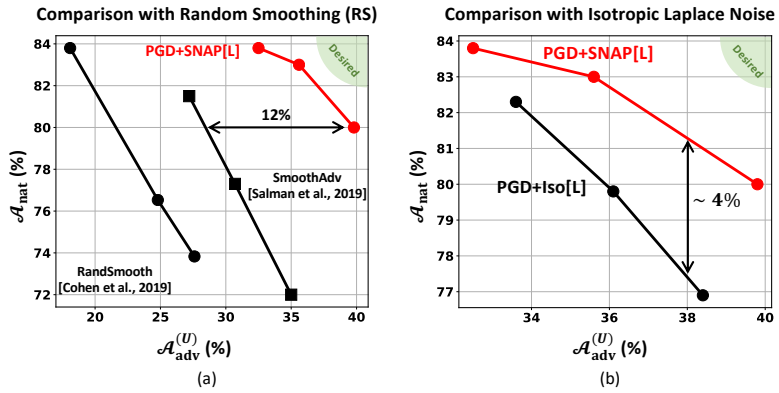


Figure 7.8: ResNet-18 CIFAR-10 results: (a) $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ for RandSmooth [196], SmoothAdv [195], and PGD+SNAP[L]; (b) $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ for PGD+SNAP[L] and PGD+Iso[L], where Iso[L] denotes a baseline SNAP alternative employing isotropic Laplace noise augmentation, *i.e.*, without noise shaping. PGD+SNAP[L] achieves better $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ trade-off due to noise shaping.

curve) and PGD+Iso[L] (*black curve*) by sweeping P_{noise}. We find that PGD+SNAP[L] achieves a better $\mathcal{A}_{\text{nat}}$ vs. $\mathcal{A}_{\text{adv}}^{(U)}$ trade-off compared to PGD+Iso[L] by making more efficient use of noise power via noise shaping. Specifically, for $\mathcal{A}_{\text{adv}}^{(U)} \approx 38$, PGD+SNAP[L] achieves a $\sim 4\%$ higher $\mathcal{A}_{\text{nat}}$.

7.7 Related Works

We categorize works on adversarial robustness of DNNs into following three categories:

Low-complexity Adversarial Training: The high computational needs of AT frameworks has spurred significant efforts in reducing their complexity [47, 43, 44, 46]. FreeAdv [43] updates weights while accumulating multiple attack iterations. FastAdv [44] employs *appropriate* use of single-step attacks, while Zheng *et al.* [46] leverage inter-epoch similarity between adversarial perturbations. However, these fast AT methods seek robustness against a single perturbation type, *e.g.*, ℓ_∞ norm-bounded perturbations. In contrast, SNAP expands the capabilities of these AT frameworks by enhancing robustness to the union of three perturbation types $(\ell_\infty, \ell_2, \ell_1)$, while preserving their efficiency.

Robustness against Union of Perturbation Models: The focus on the

robustness against the union of multiple perturbation types is relatively new. Kang *et al.* [204] studied transferability between different perturbation types, while Jordan *et al.* [205] considered combination attacks with low perceptual distortion. Stutz *et al.* [206] proposed a modification in AT to *detect* images with different models of perturbations via confidence thresholding, but they do not attempt to *classify* perturbed images correctly. For accurate classification in the presence of different perturbation models, Tramer *et al.* [106] studied empirical and theoretical trade-offs involved in including multiple perturbation types simultaneously during training. Maini *et al.* [107] further built upon this work to propose the multi steepest descent (MSD) AT framework which chooses one among the three perturbation models $(\ell_\infty, \ell_2, \ell_1)$ in each attack iteration during training, achieving SOTA adversarial accuracy on CIFAR-10 against the union of the $(\ell_\infty, \ell_2, \ell_1)$ perturbation models, albeit at a high $(10\times)$ training time. In contrast, SNAP provides high robustness against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbation models using established single-attack ℓ_∞ AT frameworks. This enables to showcase the benefits of our approach on large-scale datasets such as ImageNet.

Recently, Laidlaw *et al.* [194] developed a novel AT framework (PAT) with low perceptual distortion attacks to demonstrate impressive generalization to unseen attacks. In contrast, we focus on extending the capabilities of widely popular ℓ_∞-AT frameworks to providing robustness against the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations, while preserving their training efficiency.

Noise Augmentation: Multiple recent works have investigated the role of randomization in enhancing adversarial robustness [99, 207, 208, 209] with theoretical guarantees. Another prominent line of work in this category is randomized smoothing [196, 195, 210, 203], where random noise is used as a tool to compute certification bounds. Rusak *et al.* [211] also explored the role of noise augmentation for improving the robustness against common-corruptions [10]. In contrast, in SNAP, noise augmentation is used as a means to enable widely popular ℓ_∞-AT frameworks to efficiently achieve high robustness against the union of multiple norm-bounded perturbations. As is the characteristic of AT works, our results are primarily empirical in nature. Hence, we follow recent guidelines [14, 107] to evaluate the accuracy against the strongest possible adversaries. We do explicitly compare ℓ_∞-AT+SNAP with randomized smoothing approaches in the Sec. 7.6.3.

7.8 Discussion

In this chapter, we demonstrate how shaped noise can be introduced in deep nets to enhance their adversarial robustness vs. cost trade-off. Specifically, we propose SNAP as an augmentation that generalizes the effectiveness of ℓ_∞-AT to the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations. SNAP's strength is its simplicity and efficiency. Consequently, this work sets a first benchmark for ResNet-50 and ResNet-101 networks which are resilient to the union of $(\ell_\infty, \ell_2, \ell_1)$ perturbations on ImageNet. Note that norm-bounded perturbations include a large class of attacks, *e.g.*, gradient-based [13, 31, 106, 107, 30, 11], decision-based [32] and black-box [33] attacks.

More work is needed to extend the proposed SNAP technique to attacks beyond norm-bounded additive perturbations, *e.g.*, functional [200, 212], rotation [201], texture [202], etc. It is important to note that SNAP is meant to be an efficient technique for improving ℓ_∞-AT, and *not* a new defense. Indeed defending against a large variety of attacks simultaneously remains an open problem, with encouraging results from recent efforts [107, 194].

Another limitation of our approach is that its benefits are demonstrated empirically. It is an inevitable consequence of a lack of any theoretical guarantees for underlying AT frameworks. An interesting direction of future work is to explore whether any theoretical guarantees can be derived for anisotropic shaped noise distributions in SNAP by building upon the recent developments in randomized smoothing [195, 203]. This could be a potential avenue for bridging the gap between certification bounds and empirical adversarial accuracy.

CHAPTER 8

CONCLUSION AND FUTURE WORK

The robustness and cost challenges of deep nets discussed in Chapter 1 are well-recognized today. In response, a large number of works have independently focused on either the accuracy vs. robustness trade-off, or the accuracy vs. cost trade-off. However, as we discussed in Chapter 1, these two trade-offs start getting interwoven as one strives to push the limits of either robustness or cost. Hence, there is a need to explicitly focus on the robustness vs. cost trade-off. However it remains little explored today. In this dissertation, we attempt to fill this gap by developing techniques to improve the robustness vs. cost trade-off in the presence of both hardware noise and data noise.

8.1 Summary of Contributions

In this dissertation, we studied and improved the robustness vs. cost trade-off in diverse problem settings, ranging from the error-resilient beyond-CMOS spintronic implementations to the efficient training of robust deep nets. Specifically, we showed that both data noise and hardware noise can be shaped to improve robustness vs. cost trade-off. For example, in spintronic hardware implementations, we showed that the device-level trade-off between switching error rate, energy, and delay can be exploited to shape the error distribution at the system-level. We achieved 1000× higher tolerance to the device error rate and 3× smaller energy per decision via proposed efficient statistical error compensation. Furthermore, we also demonstrated an improvement in the robustness vs. cost trade-off via a synergistic combination of a novel spin-based analog circuit design and system-level classifier ensemble techniques. The resulting spin channel networks achieve up to 10× higher energy-efficiency compared to CMOS digital implementations.

At the other end of the design stack, we proposed a shaped noise augmentation technique to achieve high adversarial robustness for deep nets, while maintaining their training cost comparable to the standard vanilla training. Here we demonstrated that the noise can be used as a feature and its effectiveness can be enhanced by shaping its distribution. We achieved the best adversarial robustness vs. training cost trade-off compared to the nine state-of-the-art (SOTA) related works. Specifically, we achieved 4× smaller training time while maintaining the same robustness compared to the SOTA approach published just last year. Furthermore, We analyzed SNR of in-memory computing (IMC) employing resistive crossbar memory arrays. We identified SNR optimal array and sensing circuit configurations by understanding their impact on quantization vs. clipping noise trade-off. We considered both voltage and current driven crossbars, and provided design guidelines based on our analysis.

8.2 Future Prospects

Based on our work in this dissertation, we identify following four directions for promising future work:

Theoretical Guarantees on Robustness Enhancements via Noise Shaping: The enhancements in the robustness vs. cost trade-offs via noise shaping (see *e.g.* Chapter 2 and Chapter 7) demonstrated in this dissertation, while significant, are primarily empirical in nature. We believe one can derive theoretical guarantees for such robustness improvements via noise shaping. For example, in Sec. 7.6, we already discussed the relation between our proposed noise shaping techniques and randomized smoothing (RS), a promising approach for providing theoretical guarantees on adversarial robustness of deep nets [196]. We also provide the analysis for binary linear classifier showing the efficacy of the shaped noise in achieving better robustness than the isotropic noise. While the early work on RS focused on isotropic Gaussian noise [196, 195, 213, 210], more recently a larger variety of noise distributions and randomization techniques are explored [203, 214, 215, 216, 217]. These extensions can be leveraged to derive theoretical guarantees on robustness in the presence of noise shaping. This can be one of the promising ways to bridge the gap between empirical robust accuracy and certified robust

accuracy.

Noise Shaping and Error Compensation for In-Memory Computing (IMC): We discussed in Chapter 4 and Chapter 5 how computation SNR in resistive crossbar based in-memory computing implementations suffers due to the increased vulnerability to conductance and input DAC variations. We further observed that this challenge is particularly exacerbated in MRAM due to its lower conductance values. Since the root cause of all these variations is the fundamental device-level unreliability, we believe one needs to employ system-level noise shaping and error compensation techniques to improve the SNR. For resistive crossbar based IMCs, such schemes would inevitably involve some sort of on-chip chip-specific adaptive mechanism. For example, recent work [21] proposed adaptive write mechanism to compensate the write-time conductance variations in RRAM-based IMC. Similarly, even Joshi *et al*. [8] employ on-chip adaptive activation normalization to compensate for the resistance drifts in PCM devices. While these are encouraging initial works, we believe there is a much broader scope to employ error compensation techniques [4, 5, 7] for in-memory computing architectures.

Exploiting Hardware Noise for Enhancing Adversarial Robustness: In this dissertation, while we consider the impact of both hardware and data noise, we do so in the separate problem setups. In addition to our findings in Chapter 7, recent theoretical work [207, 209] has underscored the importance of randomness in enhancing adversarial robustness of deep nets. As discussed in Chapters 2–5, emerging logic and memory devices (*e.g*. spintronic, RRAM, and others) are inherently unreliable. So, an interesting avenue to explore is whether their hardware noise can be exploited to enhance adversarial robustness of deep nets. An ideal place to start this exploration is the deep net inference implementations employing IMC architectures, where its analog computation provides an additional source of hardware noise. Indeed, the recent work by Cherupally *et al*. [218] exploring the impact of IMC noise on adversarial robustness is a good first step. However, they only evaluate against weaker black-box attacks. Hence, much more work and a rigorous evaluation against the strongest white-box attacks needs to occur in order to develop techniques for exploiting hardware noise for adversarial robustness.

Finally, we would like to note that in all our contributions improving the robustness vs. cost trade-off in this dissertation, we did incur small drop in the accuracy. It seemed inevitable since any enhancement in the adversarial robustness that sustained the scrutiny of the strongest attacks was always accompanied with a drop in the natural accuracy (recall, *e.g.* AT works [22, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]). In the research community, the working hypothesis has been that such robustness may inherently be at the odds with the accuracy [55]. Furthermore, the robustness and efficiency of ML systems are still nowhere near human capabilities despite the plethora of works have been focusing on improving them in the past few years. For instance, a human brain performs incredibly complex inference tasks accurately, efficiently (consuming $\sim$ tens of Watts of power), and is largely robust to visual irregularities/corruptions such as fog, abnormal positions, irregular patches, and others. Thus, achieving human-level accuracy, robustness, and efficiency in the inference tasks remains to be a widely open problem even today. The recent efforts have been encouraging. For instance, demonstrated efficacy of adversarial training in learning robust features [190, 78, 219], possibility of recovering the accuracy drop in AT via large unlabeled data [220], and initial attempts at thinking about accuracy, robustness, and efficiency simultaneously [193, 221]. However, much more work needs to occur in this direction.

APPENDIX A

DERIVATION OF EQ. (5.4)

We apply the Kirchhoff's current law at the ith SL in Fig. 5.2 to obtain:

$$I_{\mathrm{SL},i} = \sum_{j=1}^{N} \frac{(V_j - R_s I_{\mathrm{SL},i})}{R_{i,j}} \tag{A.1}$$

where the bitcell resistance $R_{i,j} = \frac{1}{G_{i,j}}$. After substituting the value of $R_{i,j}$ and rearranging the terms, we get,

$$I_{\mathrm{SL},i} = \sum_{j=1}^{N} V_j G_{i,j} - R_s I_{\mathrm{SL},i} \left[\sum_{j=1}^{N} G_{i,j} \right] \tag{A.2}$$

Substituting $\left[\sum_{j=1}^{N} G_{i,j} \right] = \frac{1}{R_{\mathrm{arr},i}}$, we get,

$$I_{\mathrm{SL},i} = \sum_{j=1}^{N} V_j G_{i,j} - \frac{R_s}{R_{\mathrm{arr},i}} I_{\mathrm{SL},i} \tag{A.3}$$

$$\left[\frac{R_{\mathrm{arr},i} + R_s}{R_{\mathrm{arr},i}} \right] I_{\mathrm{SL},i} = \sum_{j=1}^{N} V_j G_{i,j} \tag{A.4}$$

$$I_{\mathrm{SL},i} = \left[\frac{R_{\mathrm{arr},i}}{R_{\mathrm{arr},i} + R_s} \right] \sum_{j=1}^{N} V_j G_{i,j} \tag{A.5}$$

Now we introduce the VDAC constraint, $V_{2k-1} = -V_{2k} \quad k \in \{1, \ldots, \frac{N}{2}\}$, discussed in Sec. 5.3 to obtain

$$I_{\mathrm{SL},i} = \left[\frac{R_{\mathrm{arr},i}}{R_{\mathrm{arr},i} + R_s} \right] \sum_{k=1}^{\frac{N}{2}} V_{2k} \Delta G_{i,2k} \tag{A.6}$$

where $\Delta G_{i,2k} = G_{i,2k-1} - G_{i,2k}$. Note that in the absence of any variations or noise, $I_{\mathrm{SL},i} = I_{\mathrm{sig},i}$ (see Eq. (5.1)). Thus, we get,

$$I_{\mathrm{sig},i} = \left[\frac{R_{\mathrm{arr},i}}{R_{\mathrm{arr},i} + R_s}\right] \sum_{k=1}^{\frac{N}{2}} V_{2k} \Delta G_{i,2k} \tag{A.7}$$

which is same as the Eq. (5.4).

APPENDIX B

DERIVATION OF EQ. (5.12)

Figure B.1 shows current driven $2 \times N$ crossbar. Based on charge conservation principle, we know that,

$$I_{\mathrm{SL},1} + I_{\mathrm{SL},2} = \sum_{i=1}^{N} I_i \tag{B.1}$$

Let $I_{\mathrm{BC},i,j}$ denote the current though a bitcell in ith row and jth column. By applying Kirchhoff's voltage law at the jth BL in Fig. B.1, we get,

$$I_{\mathrm{SL},1} R_s + I_{\mathrm{BC},1,j} R_{1,j} = I_{\mathrm{SL},2} R_s + I_{\mathrm{BC},2,j} R_{2,j} \tag{B.2}$$

Note that from Kirchhoff's current law at the jth BL, we have $I_{\mathrm{BC},2,j} = I_j - I_{\mathrm{BC},1,j}$. Substituting, we get,

$$I_{\mathrm{SL},1} R_s + I_{\mathrm{BC},1,j} R_{1,j} = I_{\mathrm{SL},2} R_s + (I_j - I_{\mathrm{BC},1,j}) R_{2,j} \tag{B.3}$$

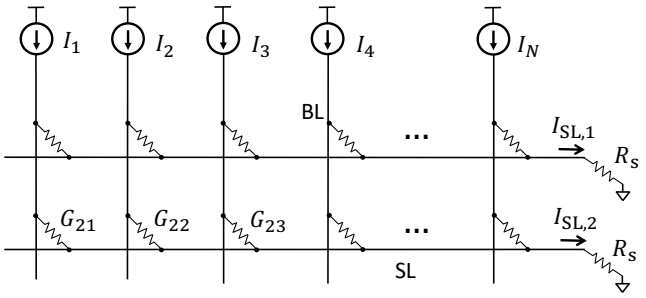


Figure B.1: Current driven $2 \times N$ crossbar.

Rearranging the terms, we get,

$$I_{\mathrm{BC},1,j} = \frac{\big[I_{\mathrm{SL},2} - I_{\mathrm{SL},1}\big] R_s}{R_{1,j} + R_{2,j}} + \frac{R_{2,j} I_j}{R_{1,j} + R_{2,j}} \tag{B.4}$$

Substituting $R_{i,j} = \frac{1}{G_{i,j}}$ in the second term, we get,

$$I_{\mathrm{BC},1,j} = \frac{\big[I_{\mathrm{SL},2} - I_{\mathrm{SL},1}\big] R_s}{R_{1,j} + R_{2,j}} + \frac{G_{1,j} I_j}{G_{1,j} + G_{2,j}} \tag{B.5}$$

Now, we apply Kirchhoff's current law at the 1st SL to obtain

$$I_{\mathrm{SL},1} = \sum_{j=1}^{N} I_{\mathrm{BC},1,j} \tag{B.6}$$

We substitute the expression for $I_{\mathrm{BC},1,j}$ from Eq. (B.5) and rearrange the terms to obtain

$$I_{\mathrm{SL},1} = R_s \big[I_{\mathrm{SL},2} - I_{\mathrm{SL},1}\big] \left[\sum_{j=1}^{N} \frac{1}{R_{1,j} + R_{2,j}}\right] + \sum_{j=1}^{N} \frac{G_{1,j} I_j}{G_{1,j} + G_{2,j}} \tag{B.7}$$

Let

$$\frac{1}{R_{\mathrm{arr}}} \triangleq \left[\sum_{j=1}^{N} \frac{1}{R_{1,j} + R_{2,j}}\right] \tag{B.8}$$

Substituting and rearranging the terms, we get,

$$I_{\mathrm{SL},1}\left[1 + \frac{R_s}{R_{\mathrm{arr}}}\right] = \frac{R_s}{R_{\mathrm{arr}}} I_{\mathrm{SL},2} + \sum_{j=1}^{N} \frac{G_{1,j} I_j}{G_{1,j} + G_{2,j}} \tag{B.9}$$

Now we have two equations, namely Eq. (B.1) and Eq. (B.9), to solve for the two unknowns $I_{\mathrm{SL},1}$ and $I_{\mathrm{SL},2}$. Substituting $I_{\mathrm{SL},2} = \left(\sum_{j=1}^{N} I_j\right) - I_{\mathrm{SL},1}$ in Eq. (B.9) and rearranging the terms, we get,

$$I_{\mathrm{SL},1}\left[1 + \frac{2R_s}{R_{\mathrm{arr}}}\right] = \frac{R_s}{R_{\mathrm{arr}}}\left[\sum_{j=1}^{N} I_j\right] + \sum_{j=1}^{N} \frac{G_{1,j} I_j}{G_{1,j} + G_{2,j}} \tag{B.10}$$

Further rearranging the terms, we get,

$$I_{\mathrm{SL},1} = \frac{R_s}{R_{\mathrm{arr}} + 2R_s}\left[\sum_{j=1}^{N} I_j\right] + \frac{R_{\mathrm{arr}}}{R_{\mathrm{arr}} + 2R_s}\left[\sum_{j=1}^{N} \frac{G_{1,j} I_j}{G_{1,j} + G_{2,j}}\right] \tag{B.11}$$

Similarly,

$$I_{\mathrm{SL},2} = \frac{R_s}{R_{\mathrm{arr}} + 2R_s}\left[\sum_{j=1}^{N} I_j\right] + \frac{R_{\mathrm{arr}}}{R_{\mathrm{arr}} + 2R_s}\left[\sum_{j=1}^{N} \frac{G_{2,j} I_j}{G_{1,j} + G_{2,j}}\right] \tag{B.12}$$

We know that in the absence of any variations or noise $I_{\mathrm{SL},i} = I_{\mathrm{sig},i}$, giving us the Eq. (5.12).

APPENDIX C

PROOF OF THEOREM 7.1

In this appendix, we prove Theorem 7.1. As noted earlier, this theorem is a general version of the certification bound proved in [196] for isotropic Gaussian noise. Hence, the proof here broadly follows a similar outline as that of [196]. The main distinction is due to the relaxation of isotropy of the distribution. We provide the complete proof for the sake of completeness.

C.1 Preliminary Lemmas

We begin by restating the following Neyman-Pearson lemma:

Lemma C.1 (**Neyman-Pearson [222, 196]**). *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be random variables in* $\mathbb{R}^D$ *with densities* $\mu_{\mathbf{x}}$ *and* $\mu_{\mathbf{y}}$. *Let* $h : \mathbb{R}^D \to \{0, 1\}$ *be a random or deterministic function. Then:*

1. *If* $S = \left\{ \boldsymbol{z} \in \mathbb{R}^D : \frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})} \leq t \right\}$ *for some* $t > 0$ *and* $\mathbb{P}(h(\mathbf{x}) = 1) \geq \mathbb{P}(\mathbf{x} \in S)$, *then* $\mathbb{P}(h(\mathbf{y}) = 1) \geq \mathbb{P}(\mathbf{y} \in S)$.
2. *If* $S = \left\{ \boldsymbol{z} \in \mathbb{R}^D : \frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})} \geq t \right\}$ *for some* $t > 0$ *and* $\mathbb{P}(h(\mathbf{x}) = 1) \leq \mathbb{P}(\mathbf{x} \in S)$, *then* $\mathbb{P}(h(\mathbf{y}) = 1) \leq \mathbb{P}(\mathbf{y} \in S)$.

Proof. See [196]. □

Now, we prove the following theorem for non-isotropic Gaussian distributions with different means, as a special case of the above lemma.

Lemma C.2 (**Neyman-Pearson lemma for Gaussians with different means**). *Let* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{x}, C_\sigma)$ *and* $\mathbf{y} \sim \mathcal{N}(\boldsymbol{x} + \boldsymbol{\xi}, C_\sigma)$, *where* $C_\sigma \in \mathbb{R}^{D \times D}$ *denotes the covariance matrix. Let* $h : \mathbb{R}^D \to \{0, 1\}$ *be any deterministic or random function. Then:*

1. *If* $S = \left\{ \boldsymbol{z} \in \mathbb{R}^D : \boldsymbol{\xi}^T C_\sigma \boldsymbol{z} \leq \beta \right\}$ *for some* β *and* $\mathbb{P}(h(\mathbf{x}) = 1) \geq \mathbb{P}(\mathbf{x} \in S)$, *then* $\mathbb{P}(h(\mathbf{y}) = 1) \geq \mathbb{P}(\mathbf{y} \in S)$.

2. *If* $S = \left\{ \boldsymbol{z} \in \mathbb{R}^D : \boldsymbol{\xi}^T C_\sigma \boldsymbol{z} \geq \beta \right\}$ *for some* β *and* $\mathbb{P}(h(\mathbf{x}) = 1) \leq \mathbb{P}(\mathbf{x} \in S)$, *then* $\mathbb{P}(h(\mathbf{y}) = 1) \leq \mathbb{P}(\mathbf{y} \in S)$.

Proof. This lemma is a straightforward application of Lemma C.1 when $\mu_{\mathbf{x}}$ and $\mu_{\mathbf{y}}$ are Gaussians with different means. Specifically, the ratio of densities $\frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})}$ reduces to a linear equation $\boldsymbol{\xi}^T C_\sigma \boldsymbol{z}$ for any given $\boldsymbol{z}$ when the densities are Gaussians with different means.

It can be shown as follows:

$$\frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})} = \frac{\exp\left\{ -\frac{1}{2}\big[\boldsymbol{z} - (\boldsymbol{x} + \boldsymbol{\xi})\big] C_\sigma^{-1} \big[\boldsymbol{z} - (\boldsymbol{x} + \boldsymbol{\xi})\big] \right\}}{\exp\left\{ -\frac{1}{2}\big[\boldsymbol{z} - \boldsymbol{x}\big] C_\sigma^{-1} \big[\boldsymbol{z} - \boldsymbol{x}\big] \right\}} \tag{C.1}$$

$$= \exp\left[\frac{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z} + \boldsymbol{z}^T C_\sigma^{-1} \boldsymbol{\xi}}{2} - \left(\frac{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{x} + \boldsymbol{x}^T C_\sigma^{-1} \boldsymbol{\xi}}{2} + \frac{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}}{2} \right) \right] \tag{C.2}$$

Note that $\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z} = (\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z})^T = \boldsymbol{z}^T C_\sigma^{-1} \boldsymbol{\xi}$, since $\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z}$ is a scalar quantity and C_σ^{-1} is a symmetric matrix. Hence, substituting, we obtain

$$\frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})} = \exp\left[\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z} - \boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{x} - \frac{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}}{2} \right] \tag{C.3}$$

For any given $t > 0$, we can choose $\beta = \ln t + \boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{x} + \frac{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}}{2}$, such that

$$\frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})} \leq t \Leftrightarrow \boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z} \leq \beta \tag{C.4}$$

$$\frac{\mu_{\mathbf{y}}(\boldsymbol{z})}{\mu_{\mathbf{x}}(\boldsymbol{z})} \geq t \Leftrightarrow \boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{z} \geq \beta \tag{C.5}$$

which concludes the proof. □

C.2 Theorem Proof

Now, we restate the Theorem 7.1 and provide its proof.

Theorem 7.1. *Let $f : \mathbb{R}^D \to \mathcal{Y}$ be an arbitrary function either deterministic or random, $\mathbf{n} \sim \mathcal{N}(0, C_\sigma)$ with a covariance matrix $C_\sigma \in \mathbb{R}^{D\times D}$, and $g(\boldsymbol{x}) = \arg\max_c \Pr(f(\boldsymbol{x}+\mathbf{n}) = c)$. For a specific $\boldsymbol{x} \in \mathbb{R}^d$, if $\exists\ c_A \in \mathcal{Y}$ and $\rho_A, \rho_B \in [0, 1]$ such that:*

$$\Pr\{f(\boldsymbol{x}+\mathbf{n}) = c_A\} \geq \rho_A \geq \rho_B \geq \max_{c\neq c_A} \Pr\{f(\boldsymbol{x}+\mathbf{n}) = c\} \tag{C.6}$$

then, for any $\boldsymbol{\xi} \in \mathbb{R}^d$, we obtain $g(\boldsymbol{x}+\boldsymbol{\xi}) = c_A$ if

$$\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}} < \frac{\boldsymbol{\Phi}^{-1}(\rho_A) - \boldsymbol{\Phi}^{-1}(\rho_B)}{2} \tag{C.7}$$

where $\boldsymbol{\Phi}$ denotes the cumulative density function of a standard 1-D Gaussian distribution $\mathcal{N}(0, 1)$.

Proof. From the definition of $g(\boldsymbol{x})$, we know that $g(\boldsymbol{x}+\boldsymbol{\xi}) = c_A$, iff

$$\Pr[f(\boldsymbol{x}+\boldsymbol{\xi}+\mathbf{n}) = c_A] > \max_{c\neq c_A} \Pr(f(\boldsymbol{x}+\boldsymbol{\xi}+\mathbf{n}) = c) \tag{C.8}$$

Thus, we need to show that $\Pr[f(\boldsymbol{x}+\boldsymbol{\xi}+\mathbf{n}) = c_A] > \Pr(f(\boldsymbol{x}+\boldsymbol{\xi}+\mathbf{n}) = c)$ for every class $c \neq c_A$. Without loss of generality, let us carry out the analysis for one such class c_B.

We start by defining the following two simplifying notations:

$$\mathbf{x}_{\text{nat}} = \boldsymbol{x} + \mathbf{n} \tag{C.9}$$

$$\mathbf{x}_{\text{per}} = \boldsymbol{x} + \boldsymbol{\xi} + \mathbf{n} \tag{C.10}$$

Note that $\mathbf{x}_{\text{nat}}$ and $\mathbf{x}_{\text{per}}$ are random vectors due to randomness of $\mathbf{n}$, whereas $\boldsymbol{x}$ and $\boldsymbol{\xi}$ are deterministic vectors.

Using this notation and from Eq. (7.4), we know that

$$\Pr[f(\mathbf{x}_{\text{nat}})] \geq \rho_A \quad \text{and} \quad \Pr[f(\mathbf{x}_{\text{nat}})] \leq \rho_B \tag{C.11}$$

Our goal is to show that

$$\Pr[f(\mathbf{x}_{\text{per}}) = c_A] > \Pr[f(\mathbf{x}_{\text{per}} = c_B)] \tag{C.12}$$

We proceed by constructing the following two half-spaces:

$$A := \left\{ \boldsymbol{z} : \boldsymbol{\xi}^T C_\sigma^{-1}(\boldsymbol{z} - \boldsymbol{x}) \leq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}}\right] \boldsymbol{\Phi}^{-1}(\rho_A) \right\} \tag{C.13}$$

$$B := \left\{ \boldsymbol{z} : \boldsymbol{\xi}^T C_\sigma^{-1}(\boldsymbol{z} - \boldsymbol{x}) \geq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}}\right] \boldsymbol{\Phi}^{-1}(1 - \rho_B) \right\} \tag{C.14}$$

Sec. C.2 shows that $\Pr[\mathbf{x}_{\text{nat}} \in A] = \rho_A$. Therefore, using Eq. (C.11) we know that $\Pr[f(\mathbf{x}_{\text{nat}} = c_A)] \geq \Pr[\mathbf{x}_{\text{nat}} \in A]$. Hence, we can apply Lemma C.2 with $h(\boldsymbol{z}) := \mathbf{1}[f(\boldsymbol{z}) = c_A]$ to conclude:

$$\Pr[f(\mathbf{x}_{\text{per}} = c_A)] \geq \Pr[\mathbf{x}_{\text{per}} \in A] \tag{C.15}$$

Similarly, Sec. C.2 shows that $\Pr[\mathbf{x}_{\text{nat}} \in B] = \rho_B$. Therefore, using Eq. (C.11) we know that $\Pr[f(\mathbf{x}_{\text{nat}} = c_B)] \leq \Pr[\mathbf{x}_{\text{nat}} \in B]$. Hence, we can apply Lemma C.2 with $h(\boldsymbol{z}) := \mathbf{1}[f(\boldsymbol{z}) = c_B]$ to conclude:

$$\Pr[f(\mathbf{x}_{\text{per}} = c_B)] \leq \Pr[\mathbf{x}_{\text{per}} \in B] \tag{C.16}$$

Now, to prove Eq. (C.12), we need to show that $\Pr[\mathbf{x}_{\text{per}} \in A] > \Pr[\mathbf{x}_{\text{per}} \in B]$, since that would complete the following chain of inequalities:

$$\Pr[f(\mathbf{x}_{\text{per}} = c_A)] \geq \Pr[\mathbf{x}_{\text{per}} \in A] > \Pr[\mathbf{x}_{\text{per}} \in B] \geq \Pr[f(\mathbf{x}_{\text{per}} = c_B)] \tag{C.17}$$

We can compute the following:

$$\Pr[\mathbf{x}_{\text{per}} \in A] = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^{-1}(\rho_A) - \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}} \right) \tag{C.18}$$

$$\Pr[\mathbf{x}_{\text{per}} \in B] = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^{-1}(\rho_B) + \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}} \right) \tag{C.19}$$

Finally, algebra shows that $\Pr[\mathbf{x}_{\text{per}} \in A] > \Pr[\mathbf{x}_{\text{per}} \in B]$ iff:

$$\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1} \boldsymbol{\xi}} < \frac{\boldsymbol{\Phi}^{-1}(\rho_A) - \boldsymbol{\Phi}^{-1}(\rho_B)}{2} \tag{C.20}$$

which concludes the proof. □

First, we compute $\Pr[\mathbf{x}_{\text{nat}} \in A]$ below:

$$\begin{aligned}
\Pr[\mathbf{x}_{\text{nat}} \in A] &= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}(\mathbf{n}) \leq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(\rho_A)\right\} \\
&= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}\mathcal{N}(0, C_\sigma) \leq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(\rho_A)\right\} \\
&= \Pr\left\{\boldsymbol{\xi}^T \mathcal{N}(0, C_\sigma^{-1} C_\sigma (C_\sigma^{-1})^T) \leq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(\rho_A)\right\} \\
&= \Pr\left\{\mathcal{N}(0, \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}) \leq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(\rho_A)\right\} \\
&= \Pr\left\{\big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\mathcal{N}(0,1) \leq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(\rho_A)\right\} \\
&= \Pr\left\{\mathcal{N}(0,1) \leq \boldsymbol{\Phi}^{-1}(\rho_A)\right\} = \boldsymbol{\Phi}\big(\boldsymbol{\Phi}^{-1}(\rho_A)\big) = \rho_A
\end{aligned}$$

Similarly, now we compute $\Pr[\mathbf{x}_{\text{nat}} \in B]$:

$$\begin{aligned}
\Pr[\mathbf{x}_{\text{nat}} \in B] &= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}(\mathbf{n}) \geq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} \\
&= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}\mathcal{N}(0, C_\sigma) \geq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} \\
&= \Pr\left\{\boldsymbol{\xi}^T \mathcal{N}(0, C_\sigma^{-1} C_\sigma (C_\sigma^{-1})^T) \geq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} \\
&= \Pr\left\{\mathcal{N}(0, \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}) \geq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} \\
&= \Pr\left\{\big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\mathcal{N}(0,1) \geq \big[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\big]\boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} \\
&= \Pr\left\{\mathcal{N}(0,1) \geq \boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} = 1 - \boldsymbol{\Phi}\big(\boldsymbol{\Phi}^{-1}(1-\rho_B)\big) = \rho_B
\end{aligned}$$

Now, we compute $\Pr[\mathbf{x}_{\text{per}} \in A]$:

$$\begin{aligned}
\Pr[\mathbf{x}_{\text{nat}} \in A] &= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}(\mathbf{n}+\boldsymbol{\xi}) \leq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(\rho_A)\right\} \\
&= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}\mathcal{N}(0,C_\sigma) \leq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(\rho_A) - \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}\right\} \\
&= \Pr\left\{\mathcal{N}(0,\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}) \leq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(\rho_A) - \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}\right\} \\
&= \Pr\left\{\left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\mathcal{N}(0,1) \leq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\left(\boldsymbol{\Phi}^{-1}(\rho_A) - \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right)\right\} \\
&= \Pr\left\{\mathcal{N}(0,1) \leq \boldsymbol{\Phi}^{-1}(\rho_A) - \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right\} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^{-1}(\rho_A) - \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right)
\end{aligned}$$

Finally, we compute $\Pr[\mathbf{x}_{\text{per}} \in B]$:

$$\begin{aligned}
\Pr[\mathbf{x}_{\text{per}} \in B] &= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}(\mathbf{n}+\boldsymbol{\xi}) \geq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(1-\rho_B)\right\} \\
&= \Pr\left\{\boldsymbol{\xi}^T C_\sigma^{-1}\mathcal{N}(0,C_\sigma) \geq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(1-\rho_B) - \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}\right\} \\
&= \Pr\left\{\mathcal{N}(0,\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}) \geq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(1-\rho_B) - \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}\right\} \\
&= \Pr\left\{\left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\mathcal{N}(0,1) \geq \left[\sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right]\boldsymbol{\Phi}^{-1}(1-\rho_B) - \boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}\right\} \\
&= \Pr\left\{\mathcal{N}(0,1) \geq \boldsymbol{\Phi}^{-1}(1-\rho_B) - \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right\} \\
&= \Pr\left\{\mathcal{N}(0,1) \leq \boldsymbol{\Phi}^{-1}(\rho_B) + \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right\} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^{-1}(\rho_B) + \sqrt{\boldsymbol{\xi}^T C_\sigma^{-1}\boldsymbol{\xi}}\right)
\end{aligned}$$

REFERENCES

[1] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] M. Meeker, "Kleiner Perkins internet trends, 2017," *Available at: http://www.kpcb.com/internet-trends*, 2017.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al., "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[4] A. D. Patil, S. Manipatruni, D. E. Nikonov, I. A. Young, and N. R. Shanbhag, "Error-resilient spintronics via the Shannon-inspired model of computation," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 5, no. 1, pp. 10–18, 2019.

[5] N. R. Shanbhag, N. Verma, Y. Kim, A. D. Patil, and L. R. Varshney, "Shannon-inspired statistical computing for the nanoscale era," in *Proceedings of the IEEE*, vol. 107, no. 1. IEEE, 2019, pp. 90–107.

[6] J. Choi, A. D. Patil, R. A. Rutenbar, and N. R. Shanbhag, "Analysis of error resiliency of belief propagation in computer vision," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 1060–1064.

[7] R. Abdallah, N. R. Shanbhag et al., "An energy-efficient ECG processor in 45 nm CMOS using statistical error compensation," *Solid-State Circuits, IEEE Journal of*, vol. 48, no. 11, pp. 2882–2893, 2013.

[8] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, no. 1, pp. 1–13, 2020.

[9] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[10] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations*, 2018.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017.

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

[14] F. Tramer, N. Carlini, W. Brendel, and A. Madry, "On adaptive attacks to adversarial example defenses," *arXiv preprint arXiv:2002.08347*, 2020.

[15] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable DNN accelerator with un-reliable ReRAM," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1769–1774.

[16] B. Zhang, L.-Y. Chen, and N. Verma, "Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 1388–1392.

[17] 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)*, 2016, pp. 770–778.

[18] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

[19] N. Pinckney, L. Shifren, B. Cline, S. Sinha, S. Jeloka, R. G. Dreslinski, T. Mudge, D. Sylvester, and D. Blaauw, "Near-threshold computing in FinFET technologies: Opportunities for improved voltage scalability," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016, pp. 1–6.

[20] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming Moore's law through energy efficient integrated circuits," in *Proceedings of the IEEE*, vol. 98, no. 2. IEEE, 2010, pp. 253–266.

[21] S. K. Gonugondla, A. D. Patil, and N. R. Shanbhag, "SWIPE: Enhancing robustness of ReRAM crossbars for in-memory computing," in *Proceedings of the 39th International Conference on Computer-Aided Design*, 2020, pp. 1–9.

[22] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning (ICML)*, 2019.

[23] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W.-S. Ku, and A. Nguyen, "Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4845–4854.

[24] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1802–1811.

[25] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 1–7.

[26] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[27] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," in *International Conference on Learning Representations (ICLR)*, 2017.

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

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

[30] P.-Y. Chen, Y. Sharma, H. Zhang, J. Yi, and C.-J. Hsieh, "EAD: Elastic-net attacks to deep neural networks via adversarial examples," in *Thirty-second AAAI Conference on Artificial Intelligence*, 2018.

[31] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, "Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4322–4330.

[32] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," in *International Conference on Learning Representations*, 2018.

[33] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *European Conference on Computer Vision*. Springer, 2020, pp. 484–501.

[34] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 2206–2216.

[35] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," *International Conference on Learning Representations (ICLR)*, 2018.

[36] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. D. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," in *International Conference on Learning Representations (ICLR)*, 2018.

[37] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "PixelDefend: Leveraging generative models to understand and defend against adversarial examples," in *International Conference on Learning Representations (ICLR)*, 2018.

[38] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," in *International Conference on Learning Representations (ICLR)*, 2018.

[39] S. Sen, B. Ravindran, and A. Raghunathan, "EMPIR: Ensembles of mixed precision deep networks for increased robustness against adversarial attacks," *arXiv preprint arXiv:2004.10162*, 2020.

[40] T. Pang, K. Xu, C. Du, N. Chen, and J. Zhu, "Improving adversarial robustness via promoting ensemble diversity," in *International Conference on Machine Learning (ICLR)*, 2019.

[41] Y. Yang, G. Zhang, D. Katabi, and Z. Xu, "ME-Net: Towards effective adversarial robustness with matrix estimation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 7025–7034.

[42] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International Conference on Machine Learning (ICML)*, 2018.

[43] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *arXiv preprint arXiv:1904.12843*, 2019.

[44] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," in *International Conference on Machine Learning (ICLR)*, 2020.

[45] C. Xie and A. Yuille, "Intriguing properties of adversarial training at scale," in *International Conference on Learning Representations*, 2020.

[46] H. Zheng, Z. Zhang, J. Gu, H. Lee, and A. Prakash, "Efficient adversarial training with transferable adversarial examples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1181–1190.

[47] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Accelerating adversarial training via maximal principle," *arXiv preprint arXiv:1905.00877*, 2019.

[48] Y.-Y. Yang, C. Rashtchian, H. Zhang, R. Salakhutdinov, and K. Chaudhuri, "A closer look at accuracy vs. robustness," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[49] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. Mann, "Fixing data augmentation to improve adversarial robustness," *arXiv preprint arXiv:2103.01946*, 2021.

[50] S. Gowal, C. Qin, J. Uesato, T. Mann, and P. Kohli, "Uncovering the limits of adversarial training against norm-bounded adversarial examples," *arXiv preprint arXiv:2010.03593*, 2020.

[51] B. Vivek and R. V. Babu, "Single-step adversarial training with dropout scheduling," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 947–956.

[52] J. Zhang, X. Xu, B. Han, G. Niu, L. Cui, M. Sugiyama, and M. Kankanhalli, "Attacks which do not kill training make adversarial learning stronger," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 278–11 287.

[53] S. Gui, H. Wang, C. Yu, H. Yang, Z. Wang, and J. Liu, "Model compression with adversarial robustness: A unified optimization framework," *arXiv preprint arXiv:1902.03538*, 2019.

[54] M. Guo, Y. Yang, R. Xu, Z. Liu, and D. Lin, "When NAS meets robustness: In search of robust architectures against adversarial attacks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 631–640.

[55] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *International Conference on Learning Representations (ICLR)*, 2019.

[56] D. Stutz, M. Hein, and B. Schiele, "Disentangling adversarial robustness and generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6976–6987.

[57] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," *Automata Studies*, vol. 34, pp. 43–98, 1956.

[58] B. Hajek and T. Weller, "On the maximum tolerable noise for reliable computation by formulas," *IEEE Transactions on Information Theory*, vol. 37, no. 2, pp. 388–391, 1991.

[59] N. Pippenger, "Reliable computation by formulas in the presence of noise," *IEEE Transactions on Information Theory*, vol. 34, no. 2, pp. 194–197, 1988.

[60] W. S. Evans and L. J. Schulman, "On the maximum tolerable noise of k-input gates for reliable computation by formulas," *IEEE Transactions on Information Theory*, vol. 49, no. 11, pp. 3094–3098, 2003.

[61] R. A. Abdallah and N. R. Shanbhag, "Robust and energy-efficient DSP systems via output probability processing," in *Computer Design (ICCD), 2010 IEEE International Conference on*. IEEE, 2010, pp. 38–44.

[62] E. P. Kim, J. Choi, N. R. Shanbhag, and R. A. Rutenbar, "Error resilient and energy efficient MRF message-passing-based stereo matching," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 897–908, 2015.

[63] S. Zhang and N. R. Shanbhag, "Embedded algorithmic noise-tolerance for signal processing and machine learning systems via data path decomposition," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3338–3350, 2016.

[64] Z. Wang, R. E. Schapire, and N. Verma, "Error adaptive classifier boosting (EACB): Leveraging data-driven training towards hardware resilience for signal inference," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 4, pp. 1136–1145, 2015.

[65] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pJ/decision 3.12 TOPS/W robust in-memory machine learning classifier with on-chip training," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 490–492.

[66] Z. He, J. Lin, R. Ewetz, J.-S. Yuan, and D. Fan, "Noise injection adaption: End-to-end ReRAM crossbar non-ideal effect adaption for neural network mapping," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.

[67] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[68] 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*. IEEE, 2009, pp. 248–255.

[69] 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*, 2017, pp. 4700–4708.

[70] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1251–1258.

[71] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1492–1500.

[72] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2752–2761.

[73] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[74] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[75] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[76] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[77] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*. PMLR, 2015, pp. 448–456.

[78] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 819–828.

[79] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[80] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2736–2744.

[81] 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*, 2017, pp. 5058–5066.

[82] X. Dong, S. Chen, and S. J. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *arXiv preprint arXiv:1705.07565*, 2017.

[83] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," *arXiv preprint arXiv:1810.05270*, 2018.

[84] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695.

[85] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Hydra: Pruning adversarially robust neural networks," *arXiv preprint arXiv:2002.10509*, 2020.

[86] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-nets: Learned quantization for highly accurate and compact deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 365–382.

[87] S. Zhu, L. H. Duong, and W. Liu, "XOR-Net: An efficient computation pipeline for binary neural network inference on edge devices," in *2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2020, pp. 124–131.

[88] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8612–8620.

[89] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks." in *MLSys*, 2019.

[90] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *International Conference on Machine Learning*. PMLR, 2018, pp. 550–559.

[91] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 734–10 742.

[92] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2018.

[93] Y. Yang, S. You, H. Li, F. Wang, C. Qian, and Z. Lin, "Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6667–6676.

[94] B. Moons, P. Noorzad, A. Skliar, G. Mariani, D. Mehta, C. Lott, and T. Blankevoort, "Distilling optimal neural networks: Rapid search in diverse spaces," *arXiv preprint arXiv:2012.08859*, 2020.

[95] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen et al., "Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 965–12 974.

[96] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan, and S. Han, "MCUnet: Tiny deep learning on IoT devices," *arXiv preprint arXiv:2007.10319*, 2020.

[97] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2498–2533, 2013.

[98] D. Nikonov and I. Young, "Benchmarking of beyond-CMOS exploratory devices for logic integrated circuits," *Exploratory Solid-State Computational Devices and Circuits, IEEE Journal on*, 2015.

[99] Z. He, A. S. Rakin, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[100] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[101] S. Yu, X. Guan, and H.-S. P. Wong, "On the stochastic nature of resistive switching in metal oxide RRAM: Physical modeling, Monte Carlo simulation, and experimental characterization," in *2011 International Electron Devices Meeting*. IEEE, 2011, pp. 3–17.

[102] W. H. Butler, T. Mewes, C. K. Mewes, P. Visscher, W. H. Rippard, S. E. Russek, and R. Heindl, "Switching distributions for perpendicular spin-torque devices within the macrospin approximation," *IEEE Transactions on Magnetics*, vol. 48, no. 12, pp. 4684–4700, 2012.

[103] S. Manipatruni, D. E. Nikonov, and I. A. Young, "Material targets for scaling all-spin logic," *Physical Review Applied*, vol. 5, no. 1, 2016.

[104] C. Xie and A. Yuille, "Intriguing properties of adversarial training at scale," in *International Conference on Learning Representations*, 2019.

[105] Y. Carmon, A. Raghunathan, L. Schmidt, P. Liang, and J. C. Duchi, "Unlabeled data improves adversarial robustness," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019, pp. 11 192–11 203.

[106] F. Tramèr and D. Boneh, "Adversarial training and robustness for multiple perturbations," in *Advances in Neural Information Processing Systems*, 2019, pp. 5858–5868.

[107] P. Maini, E. Wong, and J. Z. Kolter, "Adversarial robustness against the union of multiple perturbation models," in *International Conference on Machine Learning (ICML)*, 2020.

[108] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.

[109] B. Behin-Aein, D. Datta, S. Salahuddin, and S. Datta, "Proposal for an all-spin logic device with built-in memory," *Nature Nanotechnology*, vol. 5, no. 4, pp. 266–270, 2010.

[110] S. Manipatruni, D. E. Nikonov, R. Ramesh, H. Li, and I. A. Young, "Spin-orbit logic with magnetoelectric nodes: A scalable charge mediated nonvolatile spintronic logic," *arXiv:1512.05428*, 2015.

[111] M. G. Mankalale, Z. Liang, Z. Zhao, C. H. Kim, J.-P. Wang, and S. S. Sapatnekar, "CoMET: Composite-input magnetoelectric-based logic technology," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 3, pp. 27–36, 2017.

[112] C. Grezes, H. Lee, A. Lee, S. Wang, F. Ebrahimi, X. Li, K. Wong, J. A. Katine, B. Ocker, J. Langer et al., "Write error rate and read disturbance in electric-field-controlled magnetic random-access memory," *IEEE Magnetics Letters*, vol. 8, pp. 1–5, 2017.

[113] Y. Xie, B. Behin-Aein, and A. Ghosh, "Numerical fokker-planck simulation of stochastic write error in spin torque switching with thermal noise," in *Device Research Conference (DRC), 2016 74th Annual*. IEEE, 2016, pp. 1–2.

[114] A. F. Vincent, N. Locatelli, J.-O. Klein, W. S. Zhao, S. Galdin-Retailleau, and D. Querlioz, "Analytical macrospin modeling of the stochastic switching time of spin-transfer torque devices," *IEEE Transactions on Electron Devices*, vol. 62, no. 1, pp. 164–170, 2015.

[115] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[116] Z. Pajouhi, S. Venkataramani, K. Yogendra, A. Raghunathan, and K. Roy, "Exploring spin-transfer-torque devices for logic applications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 9, pp. 1441–1454, 2015.

[117] V. Calayir, D. E. Nikonov, S. Manipatruni, and I. A. Young, "Static and clocked spintronic circuit design and simulation with performance analysis relative to CMOS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 393–406, 2014.

[118] Z. Pajouhi, S. Venkataramani, K. Yogendra, A. Raghunathan, and K. Roy, "Exploring spin-transfer-torque devices for logic applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, no. 9, pp. 1441 – 1454, 2014.

[119] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[120] N. R. Shanbhag, R. A. Abdallah, R. Kumar, and D. L. Jones, "Stochastic computation," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 859–864.

[121] R. Hegde and N. R. Shanbhag, "Soft digital signal processing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 9, no. 6, pp. 813–823, 2001.

[122] S. K. Gonugondla, B. Shim, and N. R. Shanbhag, "Perfect error compensation via algorithmic error cancellation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 966–970.

[123] B. Shim, "Error-tolerant digital signal processing," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2005.

[124] G. V. Varatkar and N. R. Shanbhag, "Error-resilient motion estimation architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 10, pp. 1399–1412, 2008.

[125] L. Wang and N. R. Shanbhag, "Low-power filtering via adaptive error-cancellation," *IEEE Transactions on Signal Processing*, vol. 51, no. 2, pp. 575–583, 2003.

[126] A. H. Shoeb, "Application of machine learning to epileptic seizure onset detection and treatment," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[127] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[128] E. P. Kim and N. R. Shanbhag, "Soft N-modular redundancy," *IEEE Transactions on Computers*, vol. 61, no. 3, pp. 323–336, 2012.

[129] E. P. Kim, D. J. Baker, S. Narayanan, D. L. Jones, and N. R. Shanbhag, "Low power and error resilient pn code acquisition filter via statistical error compensation," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE*. IEEE, 2011, pp. 1–4.

[130] S. Ganguly, K. Y. Camsari, and S. Datta, "Evaluating spintronic devices using the modular approach," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 2, pp. 51–60, 2016.

[131] Y. Freund, R. E. Schapire et al., "Experiments with a new boosting algorithm," 1996.

[132] J. Kim, A. Paul, P. Crowell, S. J. Koester, S. S. Sapatnekar, J.-P. Wang, C. H. Kim et al., "Spin-based computing: Device concepts, current status, and a case study on a high-performance microprocessor," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 106–130, 2015.

[133] P. Bonhomme, S. Manipatruni, R. M. Iraei, S. Rakheja, S.-C. Chang, D. E. Nikonov, I. A. Young, and A. Naeemi, "Circuit simulation of magnetization dynamics and spin transport," *IEEE Transactions on Electron Devices*, vol. 61, no. 5, pp. 1553–1560, 2014.

[134] R. E. Schapire, "Explaining AdaBoost," in *Empirical inference*. Springer, 2013, pp. 37–52.

[135] C. Pan and A. Naeemi, "A proposal for energy-efficient cellular neural network based on spintronic devices," *IEEE Transactions on Nanotechnology*, vol. 15, no. 5, pp. 820–827, 2016.

[136] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, "Exploring sub-20nm FinFET design with predictive technology models," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012.

[137] D. Hisamoto, W.-C. Lee, J. Kedzierski, H. Takeuchi, K. Asano, C. Kuo, E. Anderson, T.-J. King, J. Bokor, and C. Hu, "FinFET — A self-aligned double-gate MOSFET scalable to 20 nm," *IEEE Transactions on Electron Devices*, vol. 47, no. 12, pp. 2320–2325, 2000.

[138] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, "Stochastic p-bits for invertible logic," *Physical Review X*, vol. 7, no. 3, p. 031014, 2017.

[139] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[140] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[141] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology," in *Proceedings of the National Academy of Sciences*, 1990.

[142] B. Heisele, T. Poggio, and M. Pontil, "Face detection in still gray images," Center for Biological and Computational Learning, MIT, Tech. Rep., 2000.

[143] R. Venkatesan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan, "Spintastic: Spin-based stochastic logic for energy-efficient computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. IEEE, 2015, pp. 1575–1578.

[144] A. Sengupta, M. Parsa, B. Han, and K. Roy, "Probabilistic deep spiking neural systems enabled by magnetic tunnel junction," *IEEE Transactions on Electron Devices*, vol. 63, no. 7, pp. 2963–2970, 2016.

[145] M. Sharad, D. Fan, K. Aitken, and K. Roy, "Energy-efficient non-boolean computing with spin neurons and resistive memory," *IEEE Tran. on Nanotechnology*, 2014.

[146] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, "Spin-based neuron model with domain-wall magnets as synapse," *IEEE Transactions on Nanotechnology*, vol. 11, no. 4, pp. 843–853, 2012.

[147] S. G. Ramasubramanian, R. Venkatesan, M. Sharad, K. Roy, and A. Raghunathan, "SPINDLE: SPINtronic Deep Learning Engine for large-scale neuromorphic computing," in *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*. ACM, 2014.

[148] A. Sengupta, Y. Shim, and K. Roy, "Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets," *Bio. Cir. and Sys., IEEE Tran. on*, 2016.

[149] S. Parkin and S.-H. Yang, "Memory on the racetrack," *Nature Nanotechnology*, vol. 10, no. 3, pp. 195–198, 2015.

[150] Z. Sun, X. Bi, A. K. Jones, and H. Li, "Design exploration of racetrack lower-level caches," in *Low Power Electronics and Design (ISLPED), 2014 IEEE/ACM International Symposium on*. IEEE, 2014.

[151] J. Chung, J. Park, and S. Ghosh, "Domain wall memory based convolutional neural networks for bit-width extendability and energy-efficiency," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*. ACM, 2016, pp. 332–337.

[152] Y. Wang, H. Yu, L. Ni, G.-B. Huang, M. Yan, C. Weng, W. Yang, and J. Zhao, "An energy-efficient nonvolatile in-memory computing architecture for extreme learning machine by domain-wall nanowire devices," *IEEE Transactions on Nanotechnology*, vol. 14, no. 6, pp. 998–1012, 2015.

[153] Q. Dong, K. Yang, L. Fick, D. Fick, D. Blaauw, and D. Sylvester, "Low-power and compact analog-to-digital converter using spintronic racetrack memory devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 907–918, 2017.

[154] W.-S. Khwa, J.-J. Chen, J.-F. Li, X. Si, E.-Y. Yang, X. Sun, R. Liu, P.-Y. Chen, Q. Li, S. Yu et al., "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3 ns and 55.8 TOPS/W fully parallel product-sum operation for binary DNN edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 496–498.

[155] A. Biswas and A. P. Chandrakasan, "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2018, pp. 488–490.

[156] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, W.-C. Wei, S.-Y. Wu, X. Sun, R. Liu et al., "A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in *IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2019, pp. 396–398.

[157] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A mixed-signal binarized convolutional-neural-network accelerator integrating dense weight storage and multiplication for reduced data movement," in *2018 IEEE Symposium on VLSI Circuits*. IEEE, 2018, pp. 141–142.

[158] Z. Jiang, S. Yin, M. Seok, and J.-s. Seo, "XNOR-SRAM: In-memory computing SRAM macro for binary/ternary deep neural networks," in *2018 IEEE Symposium on VLSI Technology*. IEEE, 2018, pp. 173–174.

[159] H. Jia, Y. Tang, H. Valavi, J. Zhang, and N. Verma, "A microprocessor implemented in 65nm CMOS with configurable and bit-scalable accelerator for programmable in-memory computing," *arXiv preprint arXiv:1811.04047*, 2018.

[160] R. Guo, Y. Liu, S. Zheng, S.-Y. Wu, P. Ouyang, W.-S. Khwa, X. Chen, J.-J. Chen, X. Li, L. Liu, M.-F. Chang, S. Wei, and S. Yin, "A 5.1pJ/neuron 127.3us/inference RNN-based speech recognition processor using 16 computing-in-memory SRAM macros in 65nm CMOS," in *2019 IEEE Symposium on VLSI Circuits*. IEEE, 2019, pp. 120–121.

[161] J. Kim, J. Koo, T. Kim, Y. Kim, H. Kim, S. Yoo, and J.-J. Kim, "Area-efficient and variation-tolerant in-memory BNN computing using 6T SRAM array," in *2019 IEEE Symposium on VLSI Circuits*. IEEE, 2019, pp. 118–119.

[162] X. Si, Y.-N. Tu, W.-H. Huang, J.-W. Su, P.-J. Lu, J.-H. Wang, T.-W. Liu, S.-Y. Wu, R. Liu, Y.-C. Chou, Z. Zhang, S.-H. Sie, W.-C. Wei, Y.-C. Lo, T.-H. Wen, T.-H. Hsu, Y.-K. Chen, W. Shih, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, N.-C. Lien, W.-C. Shih, Y. He, Q. Li, and M.-F. Chang, "A 28nm 64Kb 6T SRAM computing-in-memory macro with 8b MAC operation for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 246–247.

[163] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M.-F. Chang, X. Li, H. Yang, and Y. Liu, "A 65nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 234–235.

[164] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "A 351 TOPS/W and 372.4 GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine learning applications," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 242–243.

[165] J.-W. Su, X. Si, Y.-C. Chou, T.-W. Chang, W.-H. Huang, Y.-N. Tu, R. Liu, T.-W. Lu, Pei-Jungand Liu, J.-H. Wang, Z. Zhang, H. Jiang, S. Huang, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, S.-S. Sheu, S.-H. Li, H.-Y. Lee, S.-C. Chang, S. Yu, and M.-F. Chang, "A 28nm 64Kb inference-training two-way transpose multibit 6T SRAM compute-in-memory macro for AI edge chips," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 240–241.

[166] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nature Communications*, vol. 9, no. 1, pp. 1–7, 2018.

[167] I. Boybat, M. Le Gallo, S. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," *Nature Communications*, vol. 9, no. 1, pp. 1–12, 2018.

[168] C.-X. Xue, W.-H. Chen, J.-S. Liu, J.-F. Li, W.-Y. Lin, W.-E. Lin, J.-H. Wang, W.-C. Wei, T.-W. Chang, T.-C. Chang et al., "A 1Mb multibit ReRAM computing-in-memory macro with 14.6 ns parallel MAC computing time for CNN based AI edge processors," in *IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2019, pp. 388–390.

[169] M. Le Gallo, A. Sebastian, R. Mathis, M. Manica, H. Giefers, T. Tuma, C. Bekas, A. Curioni, and E. Eleftheriou, "Mixed-precision in-memory computing," *Nature Electronics*, vol. 1, no. 4, pp. 246–253, 2018.

[170] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen et al., "A fully integrated analog ReRAM based 78.4 TOPS/W compute-in-memory chip with fully parallel mac computing," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 500–502.

[171] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei, Y.-R. Chen, T.-H. Hsu, Y.-K. Chen, Y.-C. Lo, T.-H. Wen, C.-C. Lo, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, and M.-F. Chang, "A 22nm 2Mb ReRAM compute-in-memory macro with 121-28TOPS/W for multibit MAC computing for tiny AI edge devices," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2020, pp. 244–245.

[172] C.-X. Xue, Y.-C. Chiu, T.-W. Liu, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, S.-Y. Wei, C.-Y. Lee et al., "A CMOS-integrated compute-in-memory macro based on resistive random-access memory for AI edge devices," *Nature Electronics*, vol. 4, no. 1, pp. 81–90, 2021.

[173] V. Naik, K. Lee, K. Yamane, R. Chao, J. Kwon, N. Thiyagarajah, N. Chung, S. Jang, B. Behin-Aein, J. Lim et al., "Manufacturable 22nm FD-SOI embedded MRAM technology for industrial-grade MCU and IOT applications," in *2019 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2019, pp. 2–3.

[174] Y. Long, T. Na, and S. Mukhopadhyay, "ReRAM-based processing-in-memory architecture for recurrent neural network acceleration," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, no. 99, pp. 1–14, 2018.

[175] C. Sakr, Y. Kim, and N. Shanbhag, "Analytical guarantees on numerical precision of deep neural networks," in *International Conference on Machine Learning*, 2017, pp. 3007–3016.

[176] B. Moons, K. Goetschalckx, N. Van Berckelae, and M. Verhelst, "Minimum energy quantized neural networks," in *Asilomar Conference on Signals, Systems and Computer*, 2017.

[177] M. F. Snoeij, A. J. Theuwissen, K. A. Makinwa, and J. H. Huijsing, "Multiple-ramp column-parallel ADC architectures for CMOS image sensors," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2968–2977, 2007.

[178] S. H. Kang and S.-O. Jung, "Embedded STT-MRAM: Device and design," in *More than Moore Technologies for Next Generation Computer Design*. Springer, 2015, pp. 73–99.

[179] S. K. Gonugondla, M. Kang, and N. R. Shanbhag, "A variation-tolerant in-memory machine learning classifier via on-chip training," *IEEE Journal of Solid-State Circuits*, no. 99, pp. 1–11, 2018.

[180] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6T SRAM array." *J. Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.

[181] Q. Dong, Z. Wang, J. Lim, Y. Zhang, Y.-C. Shih, Y.-D. Chih, J. Chang, D. Blaauw, and D. Sylvester, "A 1Mb 28nm STT-MRAM with 2.8 ns read access time at 1.2 V VDD using single-cap offset-cancelled sense amplifier and in-situ self-write-termination," in *IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 480–482.

[182] J. Kim, K. Ryu, S. H. Kang, and S.-O. Jung, "A novel sensing circuit for deep submicron spin transfer torque mram (stt-mram)," *IEEE Transactions on very large scale integration (VLSI) systems*, vol. 20, no. 1, pp. 181–186, 2012.

[183] Q.-K. Trinh, S. Ruocco, and M. Alioto, "Dynamic reference voltage sensing scheme for read margin improvement in STT-MRAMs," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1269–1278, 2018.

[184] L. Bagheriye, S. Toofan, R. Saeidi, and F. Moradi, "A novel sensing circuit with large sensing margin for embedded spin-transfer torque MRAMs," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.

[185] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[186] T. Na, S. H. Kang, and S.-O. Jung, "STT-MRAM sensing: A review," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[187] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Robustness of classifiers to universal perturbations: A geometric perspective," in *International Conference on Learning Representations (ICLR)*, 2019.

[188] S. Jetley, N. Lord, and P. Torr, "With friends like these, who needs adversaries?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[189] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[190] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *arXiv preprint arXiv:1905.02175*, 2019.

[191] S. M. M. Dezfooli, A. Fawzi, O. Fawzi, P. Frossard, and S. Soatto, "Robustness of classifiers to universal perturbations: A geometric perspective," in *International Conference on Learning Representations (ICLR)*, 2018.

[192] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, "Robustness via curvature regularization, and vice versa," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[193] T.-K. Hu, T. Chen, H. Wang, and Z. Wang, "Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference," in *International Conference on Learning Representations (ICLR)*, 2020.

[194] C. Laidlaw, S. Singla, and S. Feizi, "Perceptual adversarial robustness: Defense against unseen threat models," in *International Conference on Learning Representations (ICLR)*, 2018.

[195] H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, "Provably robust deep learning via adversarially trained smoothed classifiers," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 289–11 300.

[196] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International Conference on Machine Learning (ICML)*, 2019.

[197] D. Madaan, J. Shin, and S. J. Hwang, "Learning to generate noise for robustness against multiple perturbations," *arXiv preprint arXiv:2006.12135*, 2020.

[198] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in PyTorch, TensorFlow, and JAX," *Journal of Open Source Software*, vol. 5, no. 53, p. 2607, 2020. [Online]. Available: https://doi.org/10.21105/joss.02607

[199] S. Santurkar, A. Ilyas, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Image synthesis with a single (robust) classifier," in *Advances in Neural Information Processing Systems*, 2019, pp. 1262–1273.

[200] C. Laidlaw and S. Feizi, "Functional adversarial attacks," *Advances in Neural Information Processing Systems*, 2019.

[201] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1802–1811.

[202] A. Bhattad, M. J. Chong, K. Liang, B. Li, and D. A. Forsyth, "Unrestricted adversarial examples via semantic manipulation," *arXiv preprint arXiv:1904.06347*, 2019.

[203] G. Yang, T. Duan, E. Hu, H. Salman, I. Razenshteyn, and J. Li, "Randomized smoothing of all shapes and sizes," in *International Conference on Machine Learning (ICML)*, 2020.

[204] D. Kang, Y. Sun, T. Brown, D. Hendrycks, and J. Steinhardt, "Transfer of adversarial robustness between perturbation types," *arXiv preprint arXiv:1905.01034*, 2019.

[205] M. Jordan, N. Manoj, S. Goel, and A. G. Dimakis, "Quantifying perceptual distortion of adversarial examples," *arXiv preprint arXiv:1902.08265*, 2019.

[206] D. Stutz, M. Hein, and B. Schiele, "Confidence-calibrated adversarial training: Generalizing to unseen attacks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9155–9166.

[207] R. Pinot, L. Meunier, A. Araujo, H. Kashima, F. Yger, C. Gouy-Pailler, and J. Atif, "Theoretical evidence for adversarial robustness through randomization: The case of the exponential family," in *Advances in Neural Information Processing Systems*, 2019.

[208] J. Gilmer, N. Ford, N. Carlini, and E. Cubuk, "Adversarial examples are a natural consequence of test error in noise," in *International Conference on Machine Learning*, 2019, pp. 2280–2289.

[209] R. Pinot, R. Ettedgui, G. Rizk, Y. Chevaleyre, and J. Atif, "Randomization matters. how to defend against strong adversarial attacks," in *International Conference on Machine Learning (ICML)*, 2020.

[210] B. Li, C. Chen, W. Wang, and L. C. Duke, "Certified adversarial robustness with addition Gaussian noise," *Neural Information Processing Systems (NeurIPS)*, 2019.

[211] E. Rusak, L. Schott, R. S. Zimmermann, J. Bitterwolf, O. Bringmann, M. Bethge, and W. Brendel, "A simple way to make neural networks robust against diverse image corruptions," in *European Conference on Computer Vision*. Springer, 2020, pp. 53–69.

[212] C. Xiao, J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song, "Spatially transformed adversarial examples," in *International Conference on Learning Representations*, 2018.

[213] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," *arXiv preprint arXiv:1802.03471*, 2018.

[214] A. Levine and S. Feizi, "Robustness certificates for sparse adversarial attacks by randomized ablation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4585–4593.

[215] J. Jia, X. Cao, B. Wang, and N. Z. Gong, "Certified robustness for top-k predictions against adversarial perturbations via randomized smoothing," *arXiv preprint arXiv:1912.09899*, 2019.

[216] G.-H. Lee, Y. Yuan, S. Chang, and T. S. Jaakkola, "Tight certificates of adversarial robustness for randomly smoothed classifiers," *arXiv preprint arXiv:1906.04948*, 2019.

[217] J. Jia, B. Wang, X. Cao, H. Liu, and N. Z. Gong, "Almost tight L0-norm certified robustness of top-k predictions against adversarial perturbations," *arXiv preprint arXiv:2011.07633*, 2020.

[218] S. K. Cherupally, A. Rakin, S. Yin, M. Seok, D. Fan, and J.-s. Seo, "Leveraging variability and aggressive quantization of in-memory computing for robustness improvement of deep neural network hardware against adversarial input and weight attacks," in *Proceedings of the Annual Design Automation Conference*, 2021.

[219] X. Chen, C. Xie, M. Tan, L. Zhang, C.-J. Hsieh, and B. Gong, "Robust and accurate object detection via adversarial learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 622–16 631.

[220] A. Raghunathan, S. M. Xie, F. Yang, J. Duchi, and P. Liang, "Understanding and mitigating the tradeoff between robustness and accuracy," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7909–7919.

[221] H. N. Wang, T. Chen, S. Gui, T. Hu, J. Liu, and Z. Wang, "Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7449–7461, 2020.

[222] J. Neyman and E. S. Pearson, "On the problem of the most efficient tests of statistical hypotheses," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 231, no. 694-706, pp. 289–337, 1933.