TRANSISTOR-BASED HARDWARE NEURAL NETWORK SYSTEM: SIMULATION AND ANALYSIS

By

ZHIHAO CHEN

B.S., University of Nottingham Ningbo, China, 2021

Ph.D., University of Nottingham Ningbo, China, 2025

Department of Electrical and Electronic Engineering

February 13, 2025

CHEN, Zhihao

Transistor-Based
Hardware Neural Network System:
Simulation and Analysis

Dissertation directed by Prof. James C. Greer & Dr. Amin Farjudian

ABSTRACT

The burgeoning field of artificial intelligence has spurred a shift in computational paradigms, necessitating the development of specialized hardware to support the demanding requirements of neural network processing. This thesis presents a comprehensive study on the design, simulation and analysis of hardware components integral to artificial intelligence systems, focusing on innovation for activation function generator circuits and linear transformation circuits. In this study, we will elucidate the design and performance characteristics of the two types of circuit, and we will provide simulation examples to illustrate their potential as viable alternatives to neural network accelerators.

We commence by detailing the design process of an innovative activation function circuit (AFC) with a pair of complementary metal oxide semiconductor (CMOS) transistors, which produces a novel activation function that exhibits learning performance on par with widely used activation functions employed in machine learning architectures. Through rigorous analysis, we demonstrate the efficacy of our proposed AFC in facilitating efficient information propagation within hardware neural network (HNN) systems.

Subsequently, we introduce our design of the multiply accumulate circuit (MAC), which achieves a state-of-the-art performance in regard of energy efficiency, response time, and silicon footprint in the computation stage of linear transformations operations. The optimisation of the circuit is crucial to reducing the hardware footprint and power consumption, crucial factors in the deployment of HNN systems, especially in resource-constrained environments.

At the system level, our analysis delves into the accumulation of errors within neural networks, providing information on the propagation and impact of these errors on the overall performance of the network. Recognising the limitations of current training methodologies, we propose potential optimisation algorithms aimed at enhancing the robustness and precision of hardware-based artificial intelligence systems. Preliminary results indicate promising improvements, suggesting the viability of our approach.

Lastly, on the basis of our analysis at the system level, we propose several potential applications of this technology. These include, but are not limited to, real-time artificial intelligence processing in edge devices and advanced decision-making systems where low latency and high computing capabilities are paramount.

In conclusion, our research represents a significant stride towards the realisation of efficient and powerful HNN systems. The innovations in activation function generation and linear transformation, coupled with the systematic analysis of error propagation, pave the way for future advancements in the field, with the potential to have a significant impact on the landscape of artificial intelligence implementation across various industries, particularly with regard to enhancing operational efficiency and enabling modifications in a standalone manner.

Keywords: *Hardware Neural Networks, AI Implementation, Neural Network Processing, Error Accumulation Analysis, Edge Computing*

ACKNOWLEDGMENTS

I extend my deepest appreciation to those who have guided and supported me throughout the journey of crafting this research endeavour. In particular, I am profoundly indebted to my esteemed advisors, Prof. Jim Greer and Dr. Amin Farjudian, whose unwavering dedication, insightful counsel, and meticulous attention to detail have been instrumental in shaping this work. Their scholarly acumen and rigorous academic standards have served as a beacon, illuminating the path to the successful completion of this project.

Prof. Greer, with his extensive experience and innovative approach to research methodologies, has consistently challenged me to think critically and expansively. His mentorship has been characterized by a blend of rigour and warmth, fostering an environment conducive to intellectual growth and personal development.

Dr. Farjudian's profound knowledge and keen eye for theoretical nuances have been invaluable, providing me with a robust foundation upon which to build my arguments. His patience in explaining complex concepts and his encouragement during moments of doubt have been sources of strength and inspiration.

Beyond their professional guidance, both advisors have shown genuine concern for my well-being, offering support that transcends the realm of academia. Their mentorship has been a testament to the transformative power of education and I am forever grateful for their investment in my future.

DECLARATION

The author declares that this work has been composed solely by himself and that it has not been submitted, in whole or in part, to any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

The second through fourth sections of Chapter 4, along with the second section of Chapter 6, have been developed into a manuscript titled "Variability Analysis for Hardware Neural Networks," which has been submitted to the esteemed publication IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In the field of machine learning (ML), a prevalent practice is to employ artificial neural networks (ANNs) for a variety of tasks in business, science, and technology, leveraging their abilities in classification, clustering, and regression [1]. The system has its strength in processing high-dimensional inputs with inherent complexity.

ANNs have shown great potential in a multitude of applications due to their efficient hierarchical structure. Unlike traditional computer architectures, these parallel distributed systems organize instructions and operational details into parameters, commonly known as weights [2, 3]. These parameters describes the strength of connections, or synapses, between neurons. The neurons refer to a set of mathematical abbreviations of functional units within neural networks that perform signal aggregation and non-polynomial transformations before transmitting outputs to other neurons through synaptic connections. A brief introduction on the structure of neural networks and a typical optimisation procedure for the model to fit a given function is provided in Appendix A.

To evaluate the performance of a given neural network, aside from the learning outcome, or the accuracy the system has for a specified problem, latency, fault tolerance and scalability are also vital aspects to be paid special attention to of the system specification [4]. Commonly, the research in this field usually focuses on the following aspects:

- Robustness: The predictive capability when introducing uncertainty and distortions in operation stages during operations
- Transparency: The possibility of interpreting neural network models with explanations of the basis of decisions made
- Extrapolation: The generalisation ability of the model to predict accurately the outward range of data used during calibration

However, aside from the aspects focusing mainly on the performance of the mathematical model, there are also concerns related to the physical implementation of the system. With a physical platform performing the system operations, typically on servers with numerous parallel computing devices, the potential inconsistency between the non-standard computation scheme and the widely adapted von Neumann or Harvard architecture computers has also attracted attention in recent research [5].

The configuration of the neural network system raises important questions about the significance of individual parameters or sets of parameters in influenc-

ing system performance, as well as the implications of a neuron's output given a specific input. Balancing system efficiency with the learning process is a key challenge [6].

The potential redundancy in neural network systems can lead to a high volume of computing operations, which can be energy-consuming. In cases where computing overheads are constrained, efforts are made to simplify the system structures, both in terms of mathematical models and hardware. In addition, the poor transparency in the potentially redundant parameters makes the system more difficult to analyse in terms of its robustness to potential attacks and calculation failures or to correct when errors are introduced or detected [7].

Systematic attempts are being made to assess the efficiency of interconnections in neural networks, with proposed frameworks for analysis and subsequent pruning based on these assessments. Furthermore, ongoing research is being conducted on the geometric and topological properties of neural networks and the impact of their configurations on learning outcomes.

In practical cases, a common strategy employed in practical applications involves using lower-resolution representations of parameters to decrease storage requirements and streamline computations. In addition, linear transformations can be substituted with multiple bit-level operations to enhance computational efficiency [8]. Some researchers are also investigate the mapping of activation functions to look-up tables (LUTs) as a method to improve operational efficiency. field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs) are frequently used to optimise operations and boost efficiency. Particularly, there are proposals for processing-in-memory (PIM) architectures aimed at eliminating the frequent read-and-write operations during neural network operations.

In addition to digital systems, there is an attempt to incorporate analog systems in neural network accelerator designs. In these instances, the linear transformations required for layer-wise communications are altered using specially designed array-based LUTs, enabling faster and more energy-efficient interactions. This system can be more easily integrated into memories or their equivalents, leading to reduced space occupancy. However, there is limited research dedicated to implementing analogueue LUTs, with the majority of them simply mimicking existing activation functions with transfer functions of passive and / or active components.

Although efficiency gains have been observed in various fields, concerns about the overall robustness of these approaches persist. The trade-off between resolution in data representation and the approximations made in computations raises questions about potential performance limitations in ANN systems.

1.1 Motivation and Major Research Contributions

Based on the foregoing discussions, a primary concern related to the implementation of neural networks involves optimising systematic operation efficiency in terms of response time, energy consumption, power usage, and silicon footprint. Moreover, there will be a focus on ensuring the system's resilience to variability in various parameters.

The biological nervous system, which naturally contradictions as ANN,

has shown remarkable learning abilities that surpass even the most sophisticated currently in existence. If we consider that the principles guiding neural networks mirror those governing the human brain, there is the possibility of developing a trainable system only using analog mechanisms. This implies the potential integration of linear and non-linear transformations, as well as learning and updating mechanisms, into a purely analog system.

The aim of applying non-polynomial functions as activation functions between each two linear layers of a neural network model is to prevent it from degrading to a purely linear transformation. It has been proven that activation functions can be chosen to be any non-polynomial functions [9, 10]. Based on the observation of the significant non-linearity in transistor transfer functions, it is conceivable that they could act as an analog look-up table (LUT) for neural networks implemented in hardware. Additionally, due to the mathematical similarity between linear transformations and the operations of a standard summing amplifier network, we are confident in the system's potential as a practical alternative to traditional computing devices.

The implementation of both linear and non-linear operations in neural networks typically takes the form of sub-modules referred to as layers, which are designed to accommodate multiple inputs and outputs. The two sub-modules have the ability to provide prompt responses to potential inputs, enabling the hardware neural network (HNN) system to effectively address latency issues when arranged in a cascaded form and communicated with a standardised form of signal. In particular, using transistors for the system activation functions, there is a possibility of enhanced energy efficiency compared to existing digital designs [11].

In general, the analog system can provide superior computing performance with a more efficient interface than software-based implementations and digital systems employing arithmetic logic units (ALUs) or field-programmable gate arrays (FPGAs).

In the course of our research, the key contributions are as follows:

- Developed a purely analog activation function circuit involving only a pair of complementary metal oxide semiconductors transistors. This circuit exhibits strong thermal resilience in generating a specific transfer function, primarily serving as the activation function circuit and partly supporting the multiply accumulate circuit design.
- Proposed a unique activation function based on the transfer function obtained with the aforementioned circuit, demonstrating equivalent learning capacity to other widely used activation functions.
- Designed a weighting circuit with capacitive components following the configuration and specifications of the AFC for compatibility considerations.
- Presented two versions of the summation circuit based on the design of the weighting circuit to mainly satisfy either stability requirements or efficiency concerns.
- Formulated a model describing the progression of error accumulation within a given neural network, factoring in any potential inaccuracies and noise present during analog processing.

The process of designing a circuit and applying its transfer function as

activation functions of a neural network represents a novel approach to this understudied topic, as will be exemplified in Chapter 4. Based on the extensive literature review conducted, no similar methodologies have been identified.

The MAC offers a technically advanced design solution that uses the contemporary industry technology that is readily available. The circuit is also capable of achieving a state-of-the-art (SOTA) performance as demonstrated through simulations, in regard of its operational efficiencies and scalability in terms of resolution.

Our proposed error accumulation model takes into account various perturbations specified within our design, offering valuable information on predicting errors at different layers of a neural network. Although current methods may not yet fully anticipate drops in learning outcomes without statistical methodologies, the focused efforts in this thesis towards generalisation have significantly advanced knowledge in this particular domain.

Furthermore, ongoing research efforts in the community and future work related to the research discussed in the thesis are focused on developing robust black-box optimisers and evaluating changes in learning efficiency and outcomes in relation to topology, configuration, and perturbations.

1.2 Thesis Outline

This document is divided into six primary chapters, each encompassing the core content and essential framework as delineated hereafter.

1. Chapter 2: We have conducted a thorough review of the existing literature pertaining to the development and applications of hardware neural networks (HNNs). Taking into account the potential risks associated with the non-ideal nature of its components and signal interface in practical integrated circuit designs, we also examined studies on the analysis of various sources and types of disruption, as well as recent efforts to detect, address, and rectify these issues.
2. Chapter 3: We conducted a preliminary discussion regarding the design process and considerations for hardware systems aligned with the corresponding mathematical representations required. Our evaluation encompasses a comprehensive analysis of the system's overall performance, benchmarking against established implementations, and projecting the potential effects of any deficiencies on both specific components and the system in its entirety.
3. Chapter 4: We have put forth a circuit design that features a push-pull linear follower configuration, comprising a pair of complementary metal oxide semiconductor transistors to function as the activation function circuit of the hardware neural network. Analysing the activation function generated in comparison with other commonly used functions, summarising the power consumption and latency of the circuit for nonlinear operations through simulations and evaluating of thermal resilience and tolerance to mismatches for system stability are also highlighted.
4. Chapter 5: An introduction to the multiply and accumulate components pertinent to the multiply accumulate circuit is proposed. This study will encompass an analysis of system scalability with respect to bit resolution

and network size. Additionally, it will involve a comparative assessment of the proposed system against the state-of-the-art designs, focusing on the metrics of time and energy efficiency. Furthermore, we will address the effect of potential inaccuracies resulted from rounding errors and element tolerance, and consider the relevant physical constraints affecting the system.

5. Chapter 6: The assessment of the performance of a HNN inclusive of the two previously mentioned circuits will be conducted under both optimal and less-than-optimal conditions. A model will be proposed to illustrate the accumulation of disturbances on the output side, which will be validated through practical examples. Furthermore, we will explore enhancements to the MAC aimed at fostering improved collaboration with the AFC in real-world scenarios. This will be accompanied by discussions focused on strategies for efficiency optimisation and the robustness of the proposed modifications.
6. Chapter 7: This chapter aims to deliver a comprehensive overview of under-explored topics within the discipline. It will include a thorough review of pertinent literature, emphasizing systematic considerations and prospective research directions. Additionally, it will propose actionable applications grounded in the existing scholarly framework.

Chapter 2

Summary of Related Literature

Neural networks are often regarded as versatile tools for performing a range of complex tasks. In response to energy and time limitations, there is a heightened focus on incorporating specialized hardware to optimise their operation. The advancement of computational power consumption associated with neural networks trading capabilities has prompted concerns regarding energy availability, thereby necessitates a balance between efficiency and performance. Hardware accelerators developed for this purpose have demonstrated their capability in performing certain tasks in practical applications with a more efficient manner. Our analysis in this part encompasses various implementations and their corresponding efficiency against power and energy usage during optimisation and application, responding time and hardware occupation. We will also examine studies addressing error estimation in systems with uncertainties in both signal flow in each layer and the computation process. Furthermore, we delve into discussions regarding error analysis pertaining to device and input inconsistencies and explore strategies aimed at fortifying the system's resilience.

2.1 Overview

Neural networks, especially deep neural network (DNN), have been effectively utilized in a variety of complex tasks such as visual and auditory signal recognition, decision-making, and robotic control [12]. These networks have shown a remarkable ability to automatically extract features from input data in a generalized manner with the derivative-based optimisation algorithm stochastic gradient descent (SGD), which is a simple technique that can significantly enhance the learning performance of neural networks [13].

The energy cost of accessing data in dynamic random-access memorys (DRAMs) is notably higher than the cost of performing linear transformations, causing a push to minimise data flows between computing devices and storage components [14]. Without being prevented by the inclination to employ extensive computing devices and storage with substantial costs and energy expenditures persist in certain sectors, it is not economically feasible to utilise such technology directly for edge devices or similar applications where energy budget is limited [5]. Hence, there is a need for solutions to implement such a machine learning algorithm effectively on the devices currently accessible [15]. In these applications implementing neural network in stand-alone hardware systems, em-

ploying field-programmable gate arrays (FPGAs) [16] or application-specific integrated circuits (ASICs) [17] as interface with off-chip system for storage and computation has emerged as a preferred approach [18, 19].

Furthermore, the inherent attributes of deep neural networks (DNNs) lend themselves well to leveraging analog devices for operations in scenarios that allow for a variety of platforms with precise restrictions. Recent research [20, 21] underscores the feasibility of using analog devices for lower resolution implementations of the model. Thus, a more efficient approach to the implementation of DNN is suggested to involve analog or mixed signal techniques [22].

Although it is true that neural networks can achieve satisfactory results with lower precision levels without significantly affecting learning outcomes [23], it is crucial to carefully consider the design and configurations of DNNs to ensure compatibility with hardware requirements. Furthermore, quantumization is essential in this context for optimal performance [15]. The limited precision of analog systems can lead to a decrease in accuracy, as highlighted in studies by Valavi et al. [18] and Yin et al. [19].

In this study, we will discuss the advancement of HNNs and their ability to balance learning capabilities and efficiency. The chapter will highlight the following aspects:

- Examine the implementations of hardware neural network with FPGA systems and analog systems utilising resistive and capacitive components
- Analysis of energy, time and space efficiency in comparison to the state-of-the-art (SOTA) works on hardware implementations in this field.
- Concerns related to the propagation of errors and their robustness within neural networks, including methods to anticipate differences in output values and decisions.
- Recent advances in classification and correction of distortions and algorithms that incorporate and utilise perturbations to enhance their optimisation processes.

2.2 Development of Hardware Neural Network

The mathematical model neural network has been shown effective on numerous examples of practical applications, with various high-level programming toolboxes put forward to accelerate the optimisation process and boost the accuracy of decision making in these tasks. However, there is a noticeable trade-off between the use of neural networks as general-purpose solvers for complex problems, such as non-deterministic polynomial-hard (NP-Hard) class of problems, and the significant power and space consumption the operations of neural networks require during training and applying processes [24] This trade-off has led to a growing interest in exploring alternative ways of presenting the model of neural network beyond traditional software implementations [25].

In order to design efficient systems that can be implemented in practical devices with improved response time and reduced energy consumption with the learning outcome of decision-making ability remaining unaffected, there is a renewed focus on the parallelism of performing separate operations in a HNN. This approach, which was originally developed in the 1980s, has once again become a compelling and appealing topic of research [26].

In this section, we will examine the utilisation of neural network technology through custom hardware systems and offer an evaluation of their performance based on predetermined criteria. We will explore both digital and analog systems, outlining specific criteria for each. A comparison of the performance of each design will be presented briefly.

2.2.1 Implementations of hardware neural network systems

The increasing need for efficient utilisation of neural networks have necessitated hardware configurations without unnecessary functional redundancies in data transmission processes and unused high-level functionalities in processing elements. Research into the hardware implementation of models that do not require excessive computing power due to software interactions and input / output (I/O) during real-time applications has been a focus of scholarly inquiry for some time [27]. In addition to ongoing efforts to minimize memory usage at the algorithmic level [28, 29], advancements in parallel computing [30–32] and the exploration of processing-in-memory (PIM) architectures [33–35] have been introduced to enhance I/O performance and reduce chip area through the reorganisation of processing elements.

With a well-known background of incorporating neural networks and their variations such as DNNs, convolutional neural networks (CNNs) and recursive neural networks (RNNs), the technology of HNN implementation has garnered interest in tasks focused on efficiency and research efforts [5]. When considering the convolutional layer of a CNN as a sparse fully connected matrix utilising an identical set of weights, or when we extend the temporal representation of a RNN into a spatial domain, we can arrive at a model that resembles a DNN in the traditional context. Thus, in the following article, we will concentrate on the HNN implementations for DNN acceleration.

Being able to enhance operational efficiency beyond traditional methods involving the utilisation of graphics processing units (GPUs) or equivalent devices, the technology of HNN is being applied to edge applications as a promising accelerator to perform the necessary operations involved in the neural network model, where constraints such as latency, energy consumption, and space availability are critical considerations [5].

In practice, field-programmable gate array (FPGA) based HNNs are preferred over digital signal processor (DSP) or application-specific integrated circuit (ASIC) implementations due to their parallel computing capabilities and programmability [27]. This system, utilizing a limited variety of functional blocks, can achieve response time efficiency up to 16 times greater than central processing unit (CPU) based software models [36]. Furthermore, embedded systems offer robust parallel computing at lower costs compared to conventional CPU or GPU based technology [36].

Analog HNNs implementations have gained attention for increasing energy efficiency and response speed compared to digital methods such as FPGAs [37]. This research stream enhances integration with practical applications like sensor systems, avoiding cumbersome conversion between analog and digital signals. Interest in analog approaches has persisted since the 1980s due to these advantages [26].

In HNN applications, resistive components organised in a "crossbar" configuration have drawn significant attention from academia and industry [38]. These components act as efficient LUTs for matrix operations [39, 40]. With components equipped to store and implement weighting information in a single stage, both complementary metal oxide semiconductor (CMOS) or memristor-based summation systems built on the basis of operational amplifiers (Op-Amps) can be arranged in a parallel fashion within the crossbar system [41]. The memristor can have its conductance manipulated by controlling current flow, enabling efficient switching between two states. This allows it to store and implement binary information in a single device, similar to a floating gate transistor, but with a smaller physical footprint and a simplified control system [42].

By modifying connections in crossbar configurations to capacitors or adjustable components while following neural networks mathematical models, the crossbar framework provides an alternative for low static power HNN implementation. A clock-dependent digital pulse signal enables capacitor-based multiply accumulate circuit (MAC) with a resolution of 3 to 6 bits using switched capacitors [43]. Design enhancements include a "digital-to-capacitor converter" for a scaled 9-bit resolution [44]. Additionally, "memcapacitors", inspired by memristors, offer a configurable capacitor design that can more accurately emulate biological neurons [45, 46].

Unlike traditional approaches that merely replicate mathematical models of neural networks, some implementations draw inspiration from biology [47]. These include capacitive components integrated into junctions [48], or as part of the neuron structure [49], utilising modulation levels that convey signals with factors in the frequency domain or phase differences. Optimisation and applications of these implementations typically focus on spiking neural networks (SNNs) [50, 51], which have demonstrated greater computational efficiency than traditional CNNs [52, 53].

In addition to the aforementioned implementations, there are various types of architecture that are less commonly discussed. HNNs with optics-based technology have been developed based on the wavelength and phase of electromagnetic waves, such as light [54]. In addition, there are other designs, ranging from digital phase-domain accelerators [55] to a current-base system compatible with the static random-access memory (SRAM) system [21, 32], each with their own unique characteristics and demonstrated efficiency in specific applications. Certain technologies will be examined in greater detail regarding their operational efficiency and power consumption at specified frequencies.

Among all configurations and designs discussed above, a primary focus is to effectively navigate the delicate balance between the capacity for parallel computing and operational efficiency, particularly concerning response time, power consumption. Constraints arise from factors such as bit precision and design complexity, as well as the cost to integrate non-polynomial look-up tables (LUTs) and memory devices [27]. The endeavour to achieve the requisite computational precision and capabilities is significantly hindered by these physical limitations, which have emerged as principal obstacles to the advancement of HNNs.

There have also been noted concerns regarding the potential limitations and uncertainties surrounding the operational efficacy of memristive HNNs. The

system is purportedly susceptible to noise and exhibits inherent non-polynomial characteristics within its components in the linear operation stage. In addition, there are design limitations related to the voltage and current prerequisites to write information [56]. Furthermore, the implementation of the derivative-based back-propagation learning algorithm is deemed challenging with devices with limited number of states and low resolution, and an escalation in design intricacy accompanies its scalability [57].

Additionally, a more challenging predicament faced by the technology includes the lack of readily available toolboxes on FPGA platforms and the constraints on operating frequency [58]. The absence of appropriate programming tools and a universally accepted standard impedes the progressive development of edge applications utilising neural networks.

In addition to research primarily concentrating on the implementation of the multiply and accumulate operations of neural networks on hardware platforms, there are also studies dedicated to enhancing the computational efficiency of activation function through hardware implementations. The implementation of activation functions generators using hardware alone is often not explored as extensively as the efforts to realise linear transformations within neural networks. With the ease of performing vector-matrix multiplication (VMM) in hardware systems based on Op-Amps, from traditional SRAM to various systems based on memristor or other tunable resistive components, there is little doubt about the feasibility of operating with analog and mixed signal technologies. The research on implementing the non-polynomial aspects of neural network operations is still not well-investigated. Although Mhaskar has demonstrated the ability to use any non-polynomial function as activation functions to approximate any function with high accuracy in mathematics [9, 10], academia primarily focuses on approximating wider-adopted activation functions by designing transfer functions for electronic circuits.

Typically, the hardware implementation of activation functions in analog mode can be categorized into three main types:

- *Binary* [59, 60]
- *Rectified Linear Unit* (*ReLU*) [61–65]
- *Sigmoid* [41]

It has been pointed out that there has been a lack of thorough research regarding the utilisation of a solely analog system design for implementation activation functions. In addition to the three prominent sectors, there is a scarcity of analog implementations of activation functions. A common method utilised to carry out non-polynomial transformations in HNNs implementations involves interfacing with software or FPGAs via converters [41, 61, 65, 66].

However, it is important to note that the precision and resilience of the analog HNN may be compromised in the face of various types of interference. This situation could lead to potential difficulties in layout and packaging, as well as a reduction in signal-to-noise ratio (SNR), which is a critical consideration in all circuit designs [37].

2.2.2 Output performance of hardware neural network implementations

In order to effectively evaluate and compare different implementations of mathematical models of neural networks, it is necessary to establish specific criteria for analysis. Several meta-criteria have been suggested for this purpose, including:

- Ensuring that assessments are comprehensive and applicable across a diverse array of contexts of hardware neural networks.
- Concentrate on the overarching conceptual elements, particularly those pertaining to the domains of learning, application, and the efficacy of generalisation performance.
- Independence from specific problems, despite potential challenges in generalising the problem or criterion.
- Possessing a high level of discrimination power.
- Being measurable using reasonable computing resources.

We should prioritise a hardware implementation's ability to match its corresponding mathematical model, as well as its resilience against noise and systematic errors. Furthermore, it is important to evaluate the efficiency in terms of time, energy, and space for each of the operations to be performed at an expectation level, as well as the scalability of the implementation in terms of the network topology and the resolution and precision of each data presentation and operation process [67].

In this field, a set of criteria is outlined, encompassing factors such as the speed of computation, the rate of parameter updates, and the energy efficiency of each operation or update [47]. The derivation of these parameters and additional criteria are also outlined in the works of Van et al. [67] and Cornu et al. [68].

An effective metric for evaluating implementation on digital hardware hinges on power consumption at a specific clock frequency, correlating with time delay and output quality based on task requirements. In digital designs, the integration of activation functions often bottlenecks efficiency. Many studies propose using LUTs to represent functions [76] or employing piece-wise linear (PWL) functions as activation functions [77] to mitigate this issue. In addition to considering time and energy efficiency, evaluating the performance of networks on different tasks is also crucial in academic research. In Tab. 2.1, when examining FPGA implementations as representatives of digital networks, it becomes evident that there is a wide range of performance and consumption outcomes. However, in many cases, detailed information on the configuration of networks is lacking, making it challenging to determine if the system has reached its peak performance. Therefore, in the empirical evaluation of the operational efficacy of our proposed systems, we shall assign precedence to both temporal efficiency and power consumption metrics, contingent upon the attainment of computational accuracy exceeding 90% relative to the SOTA benchmark, thereby enabling a robust comparative analysis.

The previous information clearly shows that optimising logical operations and computational resources through various algorithms is a key factor in deploying digital HNNs. Additionally, the choice of platforms for executing models and algorithms is crucial for estimating energy, time, and spatial requirements

Table 2.1: Operational efficiency in terms of power consumption at the specified clock frequency of state-of-the-art when implementing neural networks with field-programmable gate array platforms. The data have been collected or estimated based on the assumption that a reasonable predictive capability supported by demonstrative examples can be achieved.

Platform	Literature	Demonstration Task	Prediction Accuracy	Clock Frequency	Power Consumption
Efinix Ti60 FPGA	[69]	MobileNetV1	80.7%	75 MHz	137 mW
		DS-CNN	93.7%		132 mW
Cyclone IV FPGA	[70]	HAR	90%	100 MHz	100 mW (Serial)
					130 mW (Parallel)
ZCU102 FPGA	[71]	X3D	95.9%	142 MHz	26 W
	[72]	C3D	83.2%	160 MHz	N/A
		Slowonly	94.5%		
		R(2+1)D-18	88.7%		
		R(2+1)D-34	92.3%		
		X3D	96.5%		
VU9P FPGA	[73]	Jet-DNN	76.1%	384 MHz	5 W
VCU 118 FPGA	[74]	MNIST	99.2%	600 MHz	7.70 W
		CIFAR-10	87.1%		20.1 W
		CIFAR-100	65.9%		29.8 W
		Tiny ImageNet	46.7%		38.1 W
		ImageNet	40.1%		40.2 W
Zynq 7000 FPGA	[75]	MNIST	98.5%	N/A	0.192 W
		CIFAR-10	89.6%		4.629 W
		Tiny ImageNet	53.4%		1.181 W

for scaled models across industries. Consequently, establishing a universal standard to evaluate diverse HNN implementations is challenging. The next chapter will present proposed criteria for a more effective assessment framework.

Regarding the designs for analog HNNs, a significant challenge lies in the inherent inaccuracies of each element, ranging from parameter storage and application to the non-polynomial response of activation function generation. While prioritizing energy efficiency, we also emphasize the system's precision, as shown by the decision-making accuracies in Tab. 2.2. It is important to fully acknowledge the technology used in the design. Basic estimations suggest that the proposed design can scale across different technologies by examining the square of the ratio of channel length with a feasible process at a given technology [15].

The computational capacity of the network is defined by the connection primitives per second (CPPS), which is determined by the number of operations per second multiplied by the bit resolution of the inputs and parameters [67]. The precision of each signal is a crucial component of computing, along with efficiency in both computation and energy usage. The quality of analog repre-

Table 2.2: Operational efficiency in terms of power consumption at the specified operational speed of state-of-the-art when implementing neural networks with analog methodologies. The data have been collected or estimated based on the assumption that a certain level of precision can be attained, thereby facilitating a reasonable predictive capability supported by demonstrative examples.

Technology	Literature	Demonstration Task	Prediction Accuracy	Power Efficiency	Precision
5 nm CMOS	[78]	N/A		650 $TOPS/W$	7-Bit
22 nm CMOS	[79]	CIFAR-10 CIFAR-100	90.2% 64.15%	45.5 $TOPS/W$	4-Bit
	[21]	N/A		2.99 $TOPS/W$	Analog
40 nm CMOS	[80]	MNIST CIFAR-10	95.7% 81.7%	223.6 $TOPS/W$	Ternary
	[81]	CIFAR-10 CIFAR-100 ImageNet	89% 82% 67%	27.0 $TOPS/W$	Binary
	[82]	ImageNet Flowers102	69.4% 40.0%	2.2 $TOPS/W$	N/A
55 nm CMOS	[83]	CIFAR-10	88.5%	53.2 $TOPS/W$	Ternary
65 nm CMOS	[44]	MNIST	95%	20.83 $TOPS/W$	6-Bit
90 nm CMOS	[46]	"letter classification"	100%	199 $TOPS/W$	Analog
130 nm CMOS	[84]	MNIST	94.4%	78.4 $TOPS/W$	Analog

sentation or computation is often limited by fundamental physical constraints, as well as stability factors that mitigate drifting issues [85]. It is also important to consider non-idealities and variance when training the system off-chip, as neglecting these factors may lead to decreased performance [47].

Furthermore, as illustrated in Tab. 2.2, the validation of performance through the use of small-scale or highly simplified tasks and examples has indicated a decreased emphasis on the execution of extensive training or applications [46] This observation also suggests challenges in ensuring performance or training quality with suboptimal systems [86]. Thus, CPPS will be incorporated into the analysis of the proposed design to enable a more robust comparison with the SOTA benchmarks, without requiring the completion of an identical task or achieving a same decision-making capability.

Our analysis confirms that HNN systems efficiently perform traditional neural network functionalities. However, there is a lack of a universally accepted benchmark for evaluating these implementations. The literature often presents unclear insights into the importance of topology and optimisation algorithms, with minimal evidence of their performance on GPU or similar systems, raising concerns about their element-wise efficiencies. Thus, the next chapter will establish a comprehensive set of criteria for evaluating these systems.

2.3 Robustness Analysis and Optimisation

The occurrence of undesirable behaviours in neural networks due to inherent weaknesses related to the architecture and decision space of the mathematical model [87, 88], corrupted incoming signals and transmission errors [89, 90], or variabilities found in computing devices [91, 92] has garnered significant attention.

This section will mainly delve into the origins of these disturbances in the field of neural networks, as well as the efforts to identify and mitigate them, along with research focused on understanding systematic resilience against these perturbations. Additionally, there are studies exploring the applications of these factors as will be shown in this section. The impact of these issues on system functionality, particularly in the context of implementing HNNs and related fields with less clear boundaries, will also be emphasized.

2.3.1 Error analysis against variability and noise

In the realm of error analysis in the field of neural networks, one must consider the resilience against variations in input or parameters. While the model of neural networks has demonstrated the capability to tackle complex problems with a level of proficiency comparable to that of human beings [93], its susceptibility to noisy signals [94] still serves as a reminder to the academia and industry of its fragility [95]. Numerous experiments are conducted to assess and mitigate the potential against attacks that may be applied to the system, with a major focus on defending against distorted signals from the input side [96–99]. This defence strategy is rooted in the concept of optimising the system by a factor "adversarial loss" as defined by $\rho(\cdot)$ and its neighbourhood [100]. The term $\rho(\cdot)$ at a given parameter matrix P can be formulated as:

$$\rho(P) = \frac{1}{n}\sum_{n}^{i=1} \max_{\left\|x_i' - x_i\right\|_p \leq \varepsilon} \ell\left(f_P\left(x_i'\right), y_i\right), \tag{2.1}$$

where the function $f_P(\cdot)$ represents the neural network along with its parameter matrix P. The distorted input, x_i' falls within the range ε under the p dimensional Lebesgue space norm L_p on the original input x_i's basis, with y_i representing the intended output. The variable n denotes the total number of data examples employed in the optimisation phase.

The concept known as the "loss landscape" refers to a high-dimensional hyper-surface that characterizes the adversarial loss of a specific model and individual parameters within that model [101, 102]. Studies have shown that a flattened loss landscape can improve learning efficiency, generalisability, and resilience to varying inputs [99, 101]. However, it should be noted that existing research mainly emphasises the impact of input distortion on the loss landscape, often overlooking potential variations of the parameters of neural networks [95].

In the realm of general classifiers, the concept of "robustness" primarily pertains to the ability to withstand various forms of distortions such as input variance, regardless of whether there is *a priori* knowledge of the information of the model, as well as geometric variability in the loss landscape [87]. For

example, as demonstrated in a particular case study generated by Moosavi [103], even minor input distortions can sometimes severely impact the final output of a well-trained neural network when there is an understanding of the network configuration.

Researchers have also observed that as a result of different distortions introduced during the training phase, leading to the injection of errors and attacks into the dataset used, the parameter space of the trained model can be influenced even when the loss experiences minimal changes throughout the training interval [104].

As discussed previously, achieving an optimal balance between energy and time efficiency and achieving desired learning outcomes is a critical consideration for HNNs. Consequently, the adoption of parameter quantisation and the implementation of finite precision LUTs has become commonplace in these systems. Nonetheless, it is crucial to acknowledge the potential challenges associated with accuracy degradation and the accumulation of errors that may arise from employing these methodologies.

The process of quantising parameters or producing neural network chips or accelerators has the potential to cause distortion within the parameter space, consequently affecting the accuracy [95,105]. When limited to a finite number of digits for precision, the accuracy and efficiency of both training and applying the neural network will be significantly compromised [91]. By representing inputs and weights in the form of an ideal value supplemented by their respective error terms, denoted $p+\Delta p$ and $x+\Delta x$, respectively, and accounting for an additional error ε_α introduced by the activation function, the resulting distortion observed in the output z can be determined using the following derivation, as will be applied in subsequent chapters:

$$\Delta z = \left(p\Delta x + \Delta p x + \varepsilon_\times\right) \alpha'\left(px\right) + \varepsilon_\alpha, \tag{2.2}$$

where in $\varepsilon_\times$ represents the error caused by limited precision multiplication. Studies further demonstrate the linear aggregation of all error components [91]. Nevertheless, the correlation between the errors introduced into the computation process of the neural network system and its decision-making performance remains to be established.

Especially within the realm of HNNs, there has been a growing interest in investigating the impact of errors, particularly sub-systematic level ones, on the system's overall performance [106]. The majority of errors observed in these systems are typically attributed to manufacturing faults, voltage supply problems, or thermal-electric interactions [107]. It is acknowledged that the linear transforming systems with hardware implementations commonly have inherent limitations in terms of precision, which often introduces Gaussian distributed noise to the parameters presented and the operations [108]. Treating DNN systems as noisy information channel [109], it has been observed that the normalized mutual information between input and output decreases in a nearly linear fashion against the inverse of the SNR.

For FPGAs implementations, there is a need to convert mathematical models from a commonly adapted software-orientated format to a hardware-orientated format to accommodate the platform's inherited limitations. Compression meth-

ods are being explored and applied at each stage of the operations. Through layer-wise quantisation, parameters and signals originally represented in the 32-bit floating point format can be transformed to 16 bits or even 4/8 bits, while maintaining accuracy close to the original SOTA generated on a software platform [110].

It has also been observed that an accuracy decrease of less than 10% compared to the SOTA can be achieved with AlexNet when compressing activation function presentations from 32-bit to 8-bit resolution in hidden layers [111]. In some instances, halving precision does not significantly impact overall performance, and in certain cases, performance may even improve slightly without the need for further adjustment [112].

In more extreme scenarios where data are compressed into balanced ternary or binary representations, over 80% of SOTA performance can be achieved [113–115]. By incorporating LUTs and employing mixed compression techniques, models as small as 8% of their original size can still demonstrate effective performance in the designated task [116].

Examining the cases mentioned above shows that a decrease in the precision of calculations and data representations does not invariably lead to a significant increase in systematic error and learning outcomes within DNNs. Therefore, despite the fact that the significantly lower element tolerance can result in significantly lower data precision in performing the operations needed compared to digital system representations, it is still feasible and worthwhile to advance the development of HNNs using analog circuit configurations.

The architecture and complexity of a neural network can have a counter-intuitive impact on their performance. Research suggests that by adjusting the connections within each layer of a DNN model, it is possible to still maintain an impressive results. For example, a study with an AlexNet structure found that removing up to 70% of connections had no significant impact on system performance, maintaining a 70% accuracy rate compared to its origin [117]. Similarly, fine-tuning a pruning of 90% of connections of one special case did not result in a noticeable decrease in accuracy [118].

However, it is important to note that even minor errors, such as a bit flip, can significantly compromise the accuracy of a neural network in some special cases. Research has shown that a neural network with a small weight error in the form of bit flipping has a high probability of performing poorly, reaching a precision of less than 10% compared to a well-trained model achieving a precision of 85.5%. Similarly, the robustness of CNNs against parameter variance can vary based on factors such as layer size, network depth, and the overall number of parameters [119].

The analysis of these seemingly paradoxical cases suggests that, in general, there exists a significant degree of redundancy within neural networks. This phenomenon may elucidate the ability of neural networks to sustain relatively high levels of learning accuracy, even in the presence of substantial rounding errors and noise. As noted in Guo's observations, robustness and network size are generally viewed as critical factors in assessing network performance [88]. Furthermore, Arechiga's findings indicate a negative correlation between robustness and network size [119]. Consequently, the algorithm for network weight pruning holds considerable promise for enhancing fuzzy systems represented by

analog HNNs. However, due to the limited interpretability associated with neural networks, the process of accurately estimating and optimising their overall robustness remains a considerable challenge.

In light of the abundance of research findings, assessing the reliability of a specific neural network remains a challenging task [120]. This challenge persists even in scenarios where there is only one hidden layer with a maximum size of 20 [121]. An examination of a balanced binary neural network revealed a near-linear accumulation of errors across layers according to a hyper-space model of the system [122]. In this study, the error in each layer was estimated to be the square root of the quadratic sum of perturbations on input and parameters. Various approaches utilizing hyper-sphere [123] or hyper-cube [105] have been put forth. However, these approaches are restricted to binary or balanced binary simplifications of neural networks.

In addition, a sensitivity metric was introduced to measure the variation in the output relative to the changes in the input. This statistic-based analysis allows for the estimation and evaluation of the impact of parameter perturbations on a given neural network [124]. It has also been observed that for DNNs, the introduction of an upper limit for the commonly used *Rectified Linear Unit* (*ReLU*) function in the activation process can effectively prevent potential attacks originating from the input side, thereby avoiding their accumulation within layers [125]. The experiments indicate that the current methods used to evaluate the robustness of neural networks may vary depending on the specific circumstances and settings, and may lack a comprehensive analytical analysis in general.

In conclusion, due to their inherent complexity and limited transparency, neural networks exhibit a range of distinctive characteristics related to their robustness. Research in this area underscores the importance of conducting a comprehensive analysis of this particular issue. Furthermore, it highlights that the analog HNNs retain adequate capabilities to address practical challenges, even when constrained by restrictions in precision and network size.

2.3.2 Robustness optimisation for neural networks

It is well-documented that off-chip learning may experience performance degradation as a result of element non-idealities [37]. Therefore, it is necessary to perform a thorough fine-tuning based on potentially inaccurate inputs, feedback signals from loss functions, back-propagation values, and the actual assignments of each parameter in order to achieve more precise estimations [126].

In the realm of improving neural networks with robustness criteria, the concept of resilience to variations in input or adversarial attacks is highly appealing. In this context, the primary objective of optimising robustness revolves around the capacity to withstand inaccuracies or random uncertainties in input data [127]. These challenges can arise from various sources, such as unpredictable or unquantifiable estimations of labelling or input data [128], adversarial or noisy samples [129], or datasets labelled through questionable or corrupted means [130]. The optimisation techniques used for these cases employed typically involve a blend of various algorithms, including gradient descent, simulated annealing, genetic algorithms, and others [131].

In the context of data processing, particularly in cases where class boundaries are unclear or a single data point can belong to multiple classes, the quality and quantity of training samples are crucial. Pre-processing and fine-tuning of data can greatly enhance the overall performance in these circumstances [132]. It has been observed that linearly separable datasets may not converge to any local minima when perturbations are introduced during training with gradient-based optimisation algorithms [90]. Therefore, the selection of a suitable optimiser is essential for the effective generalisation of a noisy training processes [133].

In order to safeguard against potential adversarial attacks, strategies are recommended for implementation during the training and / or deployment phase of neural networks. Various frameworks have been introduced to identify possible threats or disturbances in input data [97]. In addition, adversarial attack frameworks have been devised to showcase the resilience of a model and provide additional protective measures, generally [134].

An innovative technique known as "Parametric Noise Injection", which involves introducing Gaussian distributed noise to both parameters and activation function during the training process, has yielded favourable learning results when compared to alternative defence strategies using either clean or corrupted data [135]. Furthermore, a training approach rooted in information theory and noise injection has demonstrated the potential to significantly enhance noise tolerance, potentially doubling it in some cases [107].

An efficient refinement algorithm incorporating adversarial examples into a training dataset of non-robust models has been shown to increase the system's ability to withstand approximately four times as many adversarial attacks compared to a SOTA model [94]. Moreover, through statistical analysis, it is feasible to predict the pixel-wise sensitivity of an image classifier [136]. Furthermore, employing a dual approach of an adversarial generator and a defensive optimiser can offer general resilience against distorted or adversarial input data [137].

In accordance with the design of an alternative to traditional CNNs named local binary convolutional neural network (LBCNN) proposed by Xu [138], a method involving a blend of linear transformations and non-linear activation of perturbed inputs has exhibited promising results in the realm of image classification, avoiding the need for convolution layers [139]. Furthermore, the introduction of noise into gradient information during the optimisation process has been identified as a means of enhancing both learning efficiency and the likelihood of avoiding getting trapped within unwanted local minima [140]. Notably, this approach offers the added benefit of bolstering resilience against input variability throughout the training phase [141].

In the realm of graph neural networks (GNNs), connecting unlabelled samples with mislabeled ones during the training process has been found to be capable of enhancing general robustness against label flipping [142]. This study reveals the feasibility of introducing self-calibration mechanisms in neural networks.

By adapting each fixed parameter using samples from a specified distribution, it has been observed that a trainable network can be produced that exhibits comparable efficacy in contrast to networks derived from gradient-based optimisers [143]. Furthermore, it is also possible to assign the parameters of a neural network by another network named "Hyper-network". With the utilisation of

Hyper-network, the methodology is extended to intricate architectures such as residual neural network (ResNet) yielding competitive results across multiple defined tasks. This approach also offers a quantifiable degree of uncertainty assessment [144].

In contrast to optimising for high resolution or accuracy, training on a hardware platform must also demonstrate robustness against noise and errors in feedback signals and lower-precision presentations of parameters and functions. The flatness of the local loss surface takes precedence over achieving the global minimum loss value within the landscape once a certain level of accuracy has been reached accordingly to the specifications [145]. Binarized training algorithms are utilised for updating based on the difference between a given point and its perturbation, given the interval or binary-state nature of hardware neural network accelerators such as memristive neural network chips [145, 146].

Furthermore, there are several alternative optimisation frameworks that are compatible with hardware implementations [92]. These methods include perturbation of inputs [147] or parameters [148], local learning [149, 150], self-organising feature map [151], Boltzmann machine learning [152], and others.

However, there remains a significant gap in our comprehensive understanding of how individual parameters and the overall topology can impact both the resilience and efficiency of a given neural network [87]. Although existing research has focused primarily on the potential risks posed by noisy data or parameters on the performance of the model, there are alternative optimisation algorithms that capitalise on stochastic elements for improved outcomes.

The uncertainty also pertains to the correlation between the decision boundary and the utilised activation functions. It remains unclear whether there exist valid inputs that could be situated significantly far from the boundary based on the configuration [87]. The endeavour to achieve explainability in this context is anticipated to offer insights into logical or illogical decisions, thereby indicating potential vulnerabilities in the system [153, 154]. Various initiatives have been taken to visually represent the input-output transformation process, thus facilitating a more simplified network analysis and a deeper understanding of the issues and solutions derived from the model [155].

Another potential strategy to enhance performance in inaccurate settings involves the implementation of fuzzy interfaces, which can offer an efficient, resilient, and transparent interface to neural networks [156, 157]. Furthermore, the balance between the capacity for generalisation and the efficiency in specific tasks remains a significant focal point in these studies [127].

The aforementioned literature suggests that the implementation of HNNs within fuzzy systems, particularly those represented by analog circuit systems, may yield significant improvements in global robustness. Moreover, the optimisation algorithm delineated in these papers holds potential applicability for the training methodologies employed in these low-precision systems. Furthermore, leveraging insights gained from studies on loss landscapes and decision transparency, we can enhance the training process by integrating analytical assignment with back-propagation or other alternative optimisation algorithms, thereby increasing training efficiency.

2.4 Conclusion

The general characteristic neural networks system has shown promise in a variety of applications, albeit at the expense of energy consumption, response time, and storage space for computing devices. The advancement of HNNs aims to offer a practical solution to achieve a balance between decision-making accuracy and operation efficiency in these aspects, especially in fields where the budget for these aspects is limited. Analog systems may further enhance efficiency compared with digital implementations, but their inherent inaccuracies could hinder their full potential.

Issues such as failures in processing internal and external signals, as well as vulnerabilities that arise unintentionally while iterating and responding to biased or pseudo-labeled data during training and application, have underscored the importance of optimising system robustness.

Although efforts have been made to identify and address potential risks and failures at the mathematical modelling and implementation levels, there has been a lack of focus on assessing risks during the manufacturing phase or operational periods. Many works assume that the presence of incomplete information within the system contributes the majority of the failures while neglecting the possibility of failures or errors during operation, even in well-optimised systems. In contrast, there is a noticeable deficiency in the allocation of energy and resources towards these particular areas of study, potentially leading to unforeseen hazards during the implementation phase.

Chapter 3

Methodology

The research of hardware neural network (HNN) implementation has shown the potential the designs have to represent a specific neural network, which in return highlights the importance of analyzing of difference between a hardware implementation and its mathematical prototype of an ideal neural network. The value of the aforementioned analysis is commonly overlooked or confused with the ability of the given system to perform a certain task. In this part, we will propose a set of procedures from realising the design of a hardware implemented deep neural network (DNN) system with certain criteria and assigning, tuning the magnitude of components involved, to final analysis of performance with other works as reference.

3.1 Overview

It is recognised that the analog devices or the analog part in mixed-signal-based circuit designs are typically serving as analog computing systems. In order to deploy hardware neural networks (HNNs) with such a system, with the aforementioned recognition, we can use physical reactions and dynamic behaviours to represent mathematical operations in a manner similar to analytical solving. This involves a physical system evolving from an initial state, typically the inputs states, to designated states as its outputs through state space updates based on transfer functions of the components involved [85].

By designing circuits as modules to represent the operations required in the linear and non-linear sub-modules in neural network operations and ensuring standardised communication between modules, we can construct a system that meets performance requirements as defined by the corresponding mathematical model the hardware design is to represent. We have created an experimental simulation environment to simulate necessary operations, such as multiplication and accumulation applied in linear transformation, as well as non-polynomial activation of given incoming signal vectors. According to our specification, each sub-module was tested independently to evaluate their performance under ideal and non-ideal conditions, determining the system's capacity and failure tolerance levels for various tasks.

In light of considering HNNs as a viable implementation of neural networks — which are acknowledged for their theoretical Turing complete nature — conducting exhaustive testing on all possible input configurations for tasks

characterized by infinite variability is both impractical and unnecessary. Furthermore, the inherent lack of explainability associated with neural networks complicates the ability to forecast the performance of HNNs in relation to specific tasks. As we shall elaborate later, the main objective of our forthcoming discussion will not be to assess whether a HNN can competently execute a particular task. Rather, our emphasis will be on evaluating whether, within specified tolerances in data presentation and computation, this system can produce outputs that closely mimic those generated by a finely-tuned model that has been specifically trained for the same task.

The assertions further emphasise the comprehensive and case-independent meta-criteria described in Section 2.2, thereby underscoring the significance of the conceptual components pertaining to both learning and application performance. The hypothesis we kept can be concluded as follows:

- With infinite resolution and accuracy, a hardware neural network can fully represent a given neural network .
- Neural networks are Turing complete.
- A HNN can be Turing complete with infinite resolution and accuracy.

The analysis highlights the importance of assessing the capability of a proposed design of a HNN to execute a specific task, assuming it is mathematically solvable. This involves determining if a corresponding neural network design can be accurately represented by the implementation at hand. Alternatively, one can evaluate the effectiveness of a neural network in addressing a particular problem by considering any discrepancies caused by inaccuracies in the HNN implementation. In essence, the key consideration in evaluating performance is whether the system can faithfully reflect the intended design of the neural network or if it can produce comparable results within a limited framework. A systematic approach can also be employed to assess the efficacy of individual sub-modules in achieving their theoretical mathematical equivalents. Thus, we may conclude from the claims that the primary criteria for HNNs is the operational efficiency at both the system and element levels, on the condition that the anticipated outcomes related to decision-making accuracy are satisfactorily achieved.

Additionally, it is imperative that we explore and assess the operational efficiency of the implementation in relation to conventional methods as a central objective of this technological advancement. The assessment will be conducted based on the quantities of the elements involved and the methodology employed in the execution of the identical computational tasks across each sub-module's operation.

This section will primarily concentrate on the following aspects:

- The design and standardisation of a realisation and description of the hardware neural network system at a circuit level.
- Alignment between the proposed system and the corresponding mathematical representation at the systematical level.
- Overall performance of decision-making capability estimation of the system proposed and comparison against current implementations.
- Anticipating the potential impact of non-ideal factors on individual components or the system as a whole.

3.2 Nomenclature and Validation Procedure

In the exploration of implementing HNNs, there arises a notable challenge of reconciling the theoretical framework of the neural network models with the real-world functionality of individual components of the system. The necessity for aligning these two aspects is primarily hindered by discrepancies in terminology and standards, as well as the complexities introduced by varying levels of abstraction in practical application.

In order to streamline the conversation and establish a consistent terminology, it is useful to provide the definitions and limitations of certain terms in the realm of circuit design and implementation of artificial intelligence within this document.

When discussing the signal flow within a neural network, the term "input" will refer to the incoming signal at a neural network's level of abstraction only, typically originating from an external data stream. At this level of abstraction, we conceptualise the implemented system as a series of interconnected mathematical functional blocks, each of which functions as a discrete unit. These functional blocks facilitate the processing of data through various interfaces while intentionally excluding the underlying physical principles and realisations. Likewise, the signal obtained from the final layer of the network will be referred to as the "output". Signals entering the individual non-polynomial layers, or the inputs of the activation functions, of the network will be referred to as "pre-activation values", while the outputs of these layers will be known as "activated values". The term "target" will indicate the desired output corresponding to a given input, representing the objective of network optimisation.

The terms "accuracy" and "loss" only hold significance within the context of mathematical models of neural networks in this part of the research, helping to evaluate the alignment between the network's output and its target output. In this context, accuracy and loss should not be viewed as directly associated with the information present in the data from the output layer and its associated errors. Instead, they should be understood as functionals of the output data, the output target, and the relevant evaluation criteria. Consequently, these terms should not be equated with output, output error, or similar terminology.

It is important to clarify that accuracy is obtained by integrating the output of the neural network with particular evaluation criteria to quantify the proportion of correct predictions. The term loss, on the other hand, is determined by a designated loss function that evaluates the discrepancy between the actual output and the target result. Furthermore, although accuracy and loss often have a statistically negative correlation, the two terms are not strictly revealing an equivalent aspect of the given neural network. By using specific evaluation and loss functions, systems can show a high loss in conjunction with a high accuracy simultaneously or vice versa. Meanwhile, the term "precision" refers specifically to the inaccuracies that arise from the representation of parameters or the degree of operational accuracy within neural networks, which should not be confused with the former two terms.

The operation responsible for a non-polynomial transformation within the mathematical model will be referred to as the "activation function". In contrast, a "transfer function" in the field of circuit design only, will denote the mathe-

matical representation of the relationship between two physical variables: the independent input and the corresponding dependent output of a system characterised by linear properties, as observed in a physical element or device. In certain instances, the activation function may be symbolized by the transfer function of a specific circuit, although modifying the transfer function to align with the activation function's requirements is not always feasible nor necessary. This conclusion is derived from the conditional constraints placed on the selection of the activation functions in conjunction with the non-polynomial characteristics inherent to semiconductor devices.

The aforementioned terms, excluding the transfer function, are dimensionless. In contrast, the transfer function adheres to the standards set forth by the Système International d'Unités (SI) and can be scaled by altering the units applied based on the hardware configuration and mathematical model in use.

In the context of advanced research, contemporary studies suggest that the mathematical operations inherent in neural networks can be emulated by meticulously engineered circuit architectures. However, it is important to note that the alignment between the mathematical formulation and the analog circuit may not always be precise, as discrepancies may arise from inaccuracies in modelling the non-polynomial components of the circuit's transfer functions. Furthermore, due to the limitations of computing power, the widely used simulation program with integrated circuit emphasis (SPICE) software may struggle to efficiently solve the equilibrium state transfer functions of analog circuit systems with numerous non-polynomial and inter-connected components. This can hinder the ability to conduct simulations in a single iteration of solving. Additionally, existing SPICE software may not be equipped to, set-up for, or intended to handle the optimisation process based on vast datasets commonly used in software-based training, necessitating separate simulation and verification of each trained model. The execution of circuit testing and model verification is contingent upon the utilisation of pre-trained parameters and the incorporation of randomly selected data samples. Our research suggests that component selection for network development can be tailored based on tolerance and other specifications to ensure compatibility with the desired application.

All simulations carried out in this study were carried out using the LTspice platform, using the predictive technology model (PTM) for a low-power $45\ nm$ metal-gate field effect transistor (FET), as detailed in Appendix B.

3.2.1 Module design and communications

In this work, we outline the procedural steps for designing a HNN implementation as follows:

1. Establish the format for presenting and communicating information signals.
2. Determine the appropriate domain and range for presenting the signal flow including input, output and the layer-wise communications.
3. Map functionality of the given neural network within the aforementioned range.
4. Partition of the neural network system into functional components based on specified requirements.

5. Create circuit schematics for each functional component.

The first two steps will be discussed mainly in this section, with a more thorough analysis of circuit schematic design provided in subsequent chapters.

In the field of neural network's research, it is standard practice to separate the weighting and summation operations from the activation functions by organising them into linear and non-linear layers. This abstraction allows for a more straightforward analysis of system design in both of the mathematical model and analog circuit level, as will be seen, by breaking it down into specialised submodules, each taking specific inputs and producing desired outputs. Communication between these modules at a physical implementation level is represented in terms of physical variables, with voltage levels chosen as the preferred method for simplicity in both design and analysis.

The selection of criteria for electronic systems with inter-communications across multiple levels involves determining whether an electronic or magnetic signal is more suitable for generation and processing. Factors such as device size and unwanted interference between neighbouring components must also be taken into account to optimise performance. Thus, a magnet signal is not always preferable in this area unless technology advances. When considering signal types for the interface, they can be categorised according to their domain and form.

Although both of the two options within each categories are theoretically equivalent in their outcomes can be achieved with careful design, practical considerations such as operational efficiency, systematical complexity in implementation, and compatibility with external computing devices come into play. After evaluating the pros and cons of different options, we have decided to use a voltage-based signal format relative to a common ground as the standardised communication method. This decision was influenced by the need for straightforward communication with various input and output devices, as well as the cost of potential implications of converting between current and voltage formats.

In addition, we have chosen to present signals in the time domain for better time efficiency and a more continuous flow of information. Also, the analysis process can be simplified with this form of signal by eliminating the time sequence modifications and related translations. Also, the specification enables the system to operate with negative parameters or signals. Although this may result in some loss of accuracy due to noise, we believe that the benefits of this approach outweigh the drawbacks in the context of our project.

In general, it is important to recognise that the signal strength of each layer within neural networks can exhibit considerable variability in practical applications. While this variation does not necessarily result in overflows or present significant risks in mathematical models and software implementations, direct application in HNN systems — where analog circuit systems are primarily utilised — may not be advisable. Should the input or output signals of any layer be transmitted according to their original levels, there exists a substantial likelihood of exceeding the acceptable operational range for both linear and non-linear units. Therefore, to mitigate the risk of system failure, it is imperative to implement linear contraction at each layer.

The signal is initially limited to a practical range of the supply voltage, which is balanced to allow for the representation of negative signals commonly

found in practical applications of neural networks. In practical scenarios, when dealing with the incoming signals of linear modules, the objective of scaling is to preserve a significant amount of information throughout the process for as many cases as possible. If a signal s falls within the range of $[s_{min}, s_{max}]$, a simple linear mapping $f : S \rightarrow \frac{2S}{s_{max}-s_{min}} - \frac{s_{max}+s_{min}}{s_{max}-s_{min}}$ is sufficient. In cases where the true distribution range of a signal is unknown, label each case with z, where z_{max} and z_{min} represent the maximum and minimum values, respectively. If either of these values has an absolute value greater than or equal to two thirds of the given range, the mean or median can be used as the reference point $\langle z \rangle$. Assuming $\Delta z = |z_{min} - \langle z \rangle| \geq |z_{max} - \langle z \rangle|$, the scaling relationship can be expressed as $f : Z \rightarrow \frac{3}{2\Delta z} (Z - \langle z \rangle)$. This approach is particularly useful for inputs in pattern recognition or prediction tasks with unbounded input values, as well as for the output of neurons with unbounded upper or lower activation functions, such as the widely used *Rectified Linear Unit* (*ReLU*).

Conversely, in non-polynomial layers, the primary objective of scaling is to adjust the capacitance of the system accordingly. The purpose of this process is to reduce information loss in the output stage of the designated layer while minimising distortion. Instead of simply scaling the inputs, a more direct approach is taken for activation function with both upper and lower saturation regions. In such cases, the non-saturation region is scaled down to approximately $\frac{2}{3}$ of the supply limit.

For those activation functions without saturation limits, assuming the functions are monotonically increasing, two sets of linear transformations can be applied at the input and output ends to negate the impact of the aforementioned limitation. The parameters for these transformations can be estimated based on the observed input range or derived from the previous layer's output within a certain range $z \in [z_{min}, z_{max}]$. With the activation function denoted as $\alpha(\cdot)$, we may specify the loss function $\delta(\cdot)$ is defined to measure the discrepancy between the transformed output and the ideal target output, using mean square error (MSE) as the metric:

$$\delta(A_i, B_i, A_o, B_o) = \textit{MSE}(A_o \alpha(A_i Z + B_i) + B_o, \alpha(Z)), \tag{3.1}$$

where A_i, B_i represent the linear operations applied to the input set Z of the activation function while A_o, B_o denote the transformations on the outputs. The ultimate goal of this optimisation process is to minimise the disparity between the outputs of the transformed activation function and the original one. Consequently, we can ensure that the proposed HNN operates within a secure range, while striving to retain the performance characteristics of the neural network system to the greatest extent feasible.

It can be deduced that when subject to appropriate transformations, the implemented devices generating activation functions in the form of linear functions can achieve the pre-supposed result without disrupting at its final output signal's side. In the case of generating the function *ReLU*, it is possible to establish a set of transformation that can adjust the input of the non-polynomial layer in a way that preserves the desired output.

Likewise, it is possible to adjust the input of an exponential function when it serves as the activation function in a network while adhering to the condition

shown below that ensures the target output remains unchanged.

The logarithmic function, serving as the inverse of the exponential function, exhibits a comparable yet reversed correlation across matrices. Similarly, sine and cosine functions can undergo smooth transformations, enabling periodic functions to be shifted by any multiple cycles without the need for additional adjustments of the pre-activation values to meet the demand while avoiding distortions at the activated values.

The implementation of these linear transformations will facilitate the precise characterisation of a specific category of activation functions across various input ranges, utilising the transfer functions of practical devices supplied with finite sources. Nevertheless, it may prove challenging to attain the aforementioned realisation in a lossless manner solely through the application of linear transformations when employing more generalised activation functions.

In practical application, it should be noted that the transformation may not always maintain complete accuracy due to the non-polynomial nature of the activation function $\alpha(\cdot)$, thus necessitating meticulous optimisation and compensations. In such instances, a critical and viable strategy is to extend the activation functions through Taylor expansion or Fourier transform. Alternatively, a more risky method involves approximating the given activation function to a scalable function and regarding any resulting discrepancy as a systemic error, provided it is negligible compared to errors introduced by other components of the system. In this work, we will assume that all signals including the inputs and the pre-activation values are scaled to an appropriate range, and the activation functions applied either saturations on both ends or is scalable according to the method aforementioned, without involving distortions to the systematic performance.

All normalisation methods are based on fundamental linear operations, demonstrating their potential for convergence within the linear module proposed for a thoroughly trained model in the applications we specify in this work.

By establishing a correlation between the signal within a HNN system and the parameters within a mathematical neural network model, we can forecast the alterations in efficiency and robustness terms as a physical implementation of the complexity of a neural network. The systematic complexity of the suggested circuit as a network can then be categorised based on how the quantities of elements evolve as the size and depth of the network increase. Interestingly, we can partition the linear operation's summation component into one category, where complexity increases linearly with each layer's size, and the multiplication section into another, characterized by a bi-linear growth in complexity derived from the adjoining layers' sizes. This approach allows for any systematic errors arising in the summation part to directly translate into a representation of activation functions as an unidentifiable factor or distortion. Conversely, the linear transformation matrices mentioned above will be applied uniformly across all connections to neuron weighting junctions.

3.2.2 Shared benchmark of modular system

In order to assess performance in the presence of non-ideal conditions, it is important to understand the concept of "robustness" in both the field of circuit design and computer science. As shared within the two fields, the term

robustness refers to the system's ability to function properly despite disturbances or variations.

In circuit design, the major concern of robustness is to ensure that the system can still operate effectively even in the face of certain internal interference including production failure, random electron moving or noise generated during operation. It is essential to evaluate the performance of the system in various operational conditions to ensure that it meets its intended functionality, as well as to assess whether it incorporates inherent fault tolerance mechanisms during the design phase [158–161].

On the other hand, in the realm of computer science, particularly in the context of neural networks, robustness is related to the system's ability to handle external perturbations such as noisy inputs, signal variations, sample distribution, and failure optimisation [127, 162–164]. These factors can lead to unstable or unintended behaviour, highlighting the importance of considering system performance from a systematic and practical perspective.

With differing methodologies for integrating the mathematical neural network model with physical components, including the utilisation of digital or analog techniques to represent both linear and non-linear layers, there is a necessity for establishing a standardised benchmark for articulating and evaluating performance and efficiency across these approaches.

The proposed criteria, based on the meta-criteria outlined in Section 2.2 will primarily assess the capacity of implementations to attain equivalence with their corresponding theoretical models of neural network and evaluate the energy, time and space level cost-effectiveness of their operations.

In this section of the project, we will outline the process by which we established a series of benchmarks to standardise performance comparisons across various implementations and tasks accessible. The steps involved are as follows:

1. Develop a theoretical model of a neural network optimised using software-based techniques for a arbitrarily given decision making task.
2. Validate the consistency between design of hardware and the given mathematical model of neural network at multiple levels of abstraction through various methods, as will be discussed.
3. Conduct a performance analysis of the system using established benchmarks, comparing it against current state-of-the-arts.

An in-depth explanation outlining the process of converting parameters and activation functions will be presented in the forthcoming chapters. Our emphasis will primarily be on identifying common patterns in the discourse of this section.

In order to standardise comparisons and establish criteria, we use the ideal concept of a neural network as a primary reference point. This assumes the existence of an ideal system capable of performing operations without any potential distortion or noise, with near-infinite resolution for data storage, presentation, and calculations in a well-defined and well-trained model set at the state-of-the-art (SOTA) level. While the meta-criteria stipulates that the testing procedure must be case-insensitive, for the sake of facilitating our discussion, we may choose to utilise several widely recognised benchmark examples as our initial reference points. Subsequently, we will determine the appropriate hyperparameters based on these selections. This model is presumed to be acquired

using a designated network topology and optimiser during the training phase. However, it should be noted that it is not guaranteed that it will reach a global minimum. Nevertheless, it can be regarded to be an effective solution within the vicinity of the local minimum that it attains.

The achievement of an optimal network configuration presents significant challenges in practical implementation, primarily due to the constraints imposed by the inherent limitations in the precision and quantity of computing components, including neurons and connections. In order to address this issue, we implemented a well-constructed model within a traditional programming framework, utilising a 64-bit computing environment equipped with sophisticated optimisation tools. This approach allowed for the application of a computational method that achieves a level of representational accuracy sufficiently close to that of the ideal model. This model is rigorously trained, verified, and tested using a randomised dataset to ensure reliability. The model developed on that platform, when fully trained and fixed, is double-checked with all pieces of data in the dataset specified, inputted in a randomised sequence and size of batches, to make sure any further optimising will not alter the weights further (gradient is summed to be a zero for all batches) or will oscillate within a certain range (gradient is summed to be non-zero but the ideal-case parameter lies within an interval between values a floating-point-based system can represent). These checks ensure that the model parameters remain within a reasonable margin about the local minima found during optimisation.

With a well-established network, the approximation of the ideal-case network is considered a valid benchmark of the performance of an optimised system. It should be noted that while the test does not ensure that the network has reached a definitive global minimum of the loss function space, the ability to converge parameters to a specific set of values allows for the system to achieve adequate efficiency, albeit still facing challenges in being non-deterministic polynomial-hard (NP-Hard) due to the complexity of exploring all possible parameter combinations and verifying against all available data sets.

In this project, our attention is directed towards the limitations posed by the implementation of neural network by purely analog hardware systems. We will primarily examine a proficiently trained model with parameter inaccuracies that impact signal representation and operations. Our analysis will delve into the isolated and collective influences of various aspects, spanning individual stages, combined stages, and a comprehensive evaluation throughout the system.

Upon translating the ideal parameters into approximate circuit components values based on their precision and the system specification, and the connectivity of the components, we aim to streamline the multi-stage design and analysis process for performance and robustness. To achieve this, we will use a mixed-method approach that involves analytically estimating ideal-case performance. We will also introduce variance calculated based on the method we couple the circuit connectivity and the corresponding mathematical equivalency to predict worst-case corruptions in the event of potential reduced precision. This prediction is taken from the Monte-Carlo method simulations. Furthermore, we will analyse potential failures of physical components and noise generated through analytical transfer functions using an electronic-based simulation. These results of software solving of the neural network level and layer level will be verified

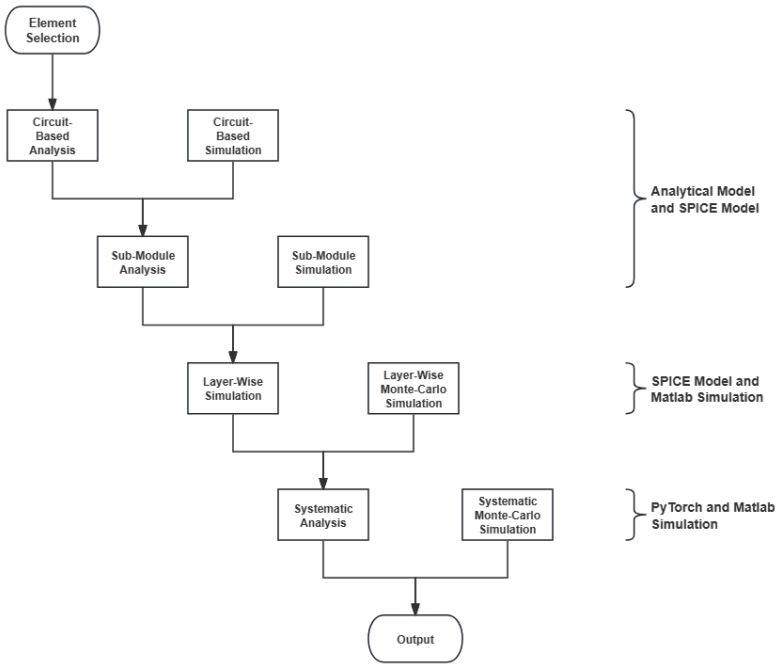


Figure 3.1: The workflow involves evaluating possible disruptions and performance at each level of abstraction based on the methodology as described in the text. Analysis at the circuit and small-scale sub-module levels is conducted using an analytical model and validated through simulation program with integrated circuit emphasis (SPICE) simulation. Layer-wise and systematic evaluations are carried out using SPICE simulation and PyTorch simulation, respectively, with verification completed in Matlab.

using simulated results as individual components and further validated through a systematic small-scale analysis. This ensures consistency between the mathematical model derived from transfer functions and the simulation outputs generated with SPICE software. The iterative approach is used at different levels of abstractions of transfer models when multiple components interact.

Once the consistency between circuit level and mathematical level simulation of both optimal output and sub-optimal distortion is confirmed, we will translate the physical model of all components involved back into a mathematical framework. We will condense the weighting, summation, or activation function of the entire circuit into a unified mathematical abstraction, incorporating the predicted disturbances according to the mathematical model. Inter-connections will be simplified through ideal mathematical operations, supplemented with additional randomised disturbances from a pre-determined distribution. This abstraction process will transform each circuit design into a mathematical representation within a matrix or an activation function, accounting for the uncertainties outlined in a study that emphasises the reliable optimisation of neural networks.

Based on the examination of perturbations observed at each element of a weighting matrix or activation function's output signals, namely their pre-activation and activated values, we will be able to accurately assess the operational impact that varying levels of element-wise variances have on the system's decision-making process. Subsequently, by determining the desired output

or tolerance level required for a system to function within an acceptable range, we can effectively ascertain the necessary specifications for component selection and manufacturing tolerances.

In order to evaluate effectiveness, we will analyse both the static and dynamic characteristics of each node within the circuit implementation. This assessment will include examining the weighting and summation devices in the linear-operations, and the circuits designed to function as an activation function generator for the non-polynomial part. Each of these sub-circuits will be modelled as a single operation of floating-point numbers in the mathematical model of neural network operation, with its energy consumption and responding time required to be examined under the case a given precision requirement is met. As the system operates in a fully parallel mode within layers, we will measure the time it takes for a peak-to-peak step-formed signal to be applied to a circuit's input and for a steady output to be observed at the output as the operation speed of the system. The energy consumed during the process and any potential reset operations, will be considered for overall energy consumption analysis.

With the convergence of the model reaching a stable local minimum, we can now conduct a detailed analysis of the performance of various implementations generated through computations [27] or LUT [165] on a field-programmable gate array (FPGA)or other potential devices such as standard chips or specialized neural chips [47, 166]. This analysis will focus on the ability of these implementations to generate similar outputs at both modular and system level, allowing comparison them to our own design. The results from each stage will be compared to those obtained from a full-resolution computation using data and parameters sourced from the specified references at each layer.

Training sessions for systems that use various configurations of activation functions are conducted using a standardised topology, learning rate, and optimiser. Limited diversity in optimisation algorithms shared during the training phase result in similar time and energy complexities of this part. Additionally, the multitude of hyper-parameter variables requiring testing and the sensitivity of the training process creates challenges in accurately estimating analytical factors [167–170]. Should the training algorithms under consideration undergo further refinement and be deemed ready for implementation, it may facilitate a thorough evaluation of training efficacy across various algorithms.

3.3 Performance and efficiency analysis

In consideration of the possibility of implementing a neural network using exclusively analog hardware and with the appropriate design parameters, it has been demonstrated numerous times to be entirely viable. This indicates that a hardware neural network (HNN) can achieve equivalent performance to its theoretical mathematical model equivalency under optimal conditions. The primary focus of this study is to determine whether the hardware implementation can deliver a performance comparable to other approaches, in terms of both efficiency and robustness, taking into account variability in the components involved and signal distortion caused by noise.

The current SOTA in the construction of HNNs varies greatly in terms of structures, ideologies and technologies, making it challenging to establish a

definitive standard that encompasses all the proposed designs. To address the concerns in the aspect of circuit design, we have chosen energy consumption, response time, and silicon footprint as the primary criteria for our implementation. We aim to demonstrate that our design has achieved a cutting-edge level in these aspects.

In light of the innovative activation function generator and the functions proposed, our analysis also encompassed a thorough examination of learning efficiency that related closer to the field of computer science. The notion that any non-polynomial function can serve as an activation function to accurately model a continuous function in the real domain [9, 10] prompts the question of whether the proposed function can rival the effectiveness of other widely utilised activation functions. Additionally, as a result of the comprehensive nature of systematic redundancy in quantities of layers, neurons and connections, and non-polynomial operations, we have established a set of criteria to assess the precision of each layer and transform it into a systematic analysis.

In the subsequent section, we will proceed under the assumption that our ideal model successfully describes the actual configuration of a specified neural network. We will then present our strategy for evaluating the performance for each relevant aspect.

3.3.1 Benchmark and method of analysis

In consideration of the modular and systematic approach to performance evaluation, as discussed previously and illustrated in Fig. 3.1, we can summarise our process for assessing performance and other relevant factors for different aspects of the system and alternative solutions suggested by existing works under both optimal and sub-optimal conditions as follows as is applied in standard system designs:

1. Categorise the system into neural network functional blocks according to their designated design objectives.
2. Establish the attainment of their designated objectives for each functional block as the performance criterion.
3. Segment each functional block into circuit groups based on mathematical operations.
4. Define the mean performance metrics for systematic operations pertaining to each circuit group as the efficiency criteria.

The activation function circuit (AFC) will be introduced in the following section, due to the novelty of the activation function, we will conduct a comparison of its compatibility with other commonly-used activation functions on commonly used datasets. This comparison will focus on analyzing the optimisation speed of networks utilising the transfer function of the AFC as an activation function g, aiming to achieve performance levels similar to other widely used activation functions. Time efficiency in this context will be measured by the training iterations required to reach a specific level of accuracy or loss, as well as the learning outcomes of the network at the same stage of optimisation with same configuration, compared with other widely used activation functions as baselines.

All experiments will utilise a shared network topology and optimiser on

a small scale to ensure consistency and efficiency in executing the tests. The objective of this phase is to establish a baseline system for comparison purposes, aiming for a discernible improvement over random guessing, rather than reach a SOTA level accuracy. Thus, we will primarily focus on small-scale networks with adequate iterations of optimisation to simplify the training and validation process. The ultimate goal is not to achieve SOTA performance in terms of mathematical modeling, but rather to assess the performance of the network with the proposed activation function in a controlled environment.

The subject of comparison is the SOTA technologies published within the last decade, which primarily prioritize stand-alone systems designed for a singular purpose. It is important to note that intricate systems encompassing functionalities unrelated to neural network processing, akin to traditional computers, will not be included in the comparison.

In analyzing the data sourced from available sources, considering the system's topology and tasks that are performed, it is noted that not all proposed designs provided a comprehensive discussion on all functional blocks involved. As such, our approach involves primarily extracting and organizing the data based on their significant contributions, if specified, while disregarding criteria that are not prominently highlighted. By assuming that in a neural network, the output magnitude and parameter visibility at each specified node are evenly distributed, we can approximate the average power and time consumption of each operation within a network by dividing the consumption within the target fields by the total number of operational functional blocks within that stage.

In order to assess efficiency, we will be analyzing the energy consumption in both static and dynamic states, as well as the response time for each potential operation and interaction between functional blocks. This analysis will also include the silicon footprint. It is important to note that an N-Channel metal oxide semiconductor (N-MOS) and a P-Channel metal oxide semiconductor (P-MOS) will each be counted as one transistor, without taking into account layout spacing or the size difference to make a matched pair, for simplicity of discussion.

The concept of operation is defined within the framework of mathematical representations outlined in our documentation, with equal consideration given to both linear and non-linear operations. To clarify, each instance of multiplication, addition, and non-polynomial transformation facilitated by the activation function for a specific input-output pair will be considered as a single operation. It is important to note that in today's advanced computing device architectures, particularly those featuring high levels of parallel processing capabilities, the time required to compute a set of inputs using a specific neural network will significantly vary across different implementations or architectures. As such, we rely on the average duration needed to complete each operation as a point of reference.

In this work, the analysis of the AFC will primarily focus on its capability to consistently generate desired outcomes and maintain a standard level of learning proficiency in terms of accuracy and loss at a comparable number of training iterations to commonly utilised activation functions. Additionally, the circuit will be assessed as a functional component in terms of its static and dynamic power consumption, as well as response time, without undergoing any optimisation procedures.

In the upcoming analysis of the AFC proposal, we will examine the efficiency of time and energy by testing the latency in response and power consumption during dynamic and static mode using a SPICE software. This will involve testing a standalone circuit with an ideal voltage source and outputs of no loading.

To assess performance, we will input a sweeping periodic signal of a triangle and sine waveform at various frequencies and analyse the phase shift and distortion to determine the limits of response. Power consumption will be calculated by measuring the root mean square (RMS) current level for each region the transistor structure is operating in and for a whole cycle.

The evaluation of the multiply accumulate circuit (MAC) will focus on its ability to execute accurately a specific linear transformation task. This assessment will involve testing the performance of the weighting circuit in applying the intended weight to a designated input and analysing the variance between the analog output and the theoretical output derived from mathematical computations. Furthermore, we will assess the overall capacity of the system to generate a linear transformation on a varied set of parameters and inputs within the designated operational range, by evaluating the deviation between its output and the desired target output.

In a MAC where external information on the parameters and data flow of its potential application is not available, our modular testing will prioritise the analysis of peak-to-peak power and time consumption for all possible combinations of inner connections within the proposed weighting system. In order to estimate space requirements, we will provide a formula for determining the minimum transistor count needed for each junction at every stage, with resolution as an independent variable of the model, considering the lack of resolution information for a specific application.

In the assessment and comparison of the SOTA works, particularly in the realm of digital systems, we will be conducting a thorough analysis of the clock cycles required for each operation individually. The required number of cycles will then be multiplied by a standard time interval of $0.2\ ns$, corresponding to a clock frequency of $5\ GHz$, which is commonly achieved by a typical central processing unit (CPU) may achieve. For the purposes of this analysis, we will disregard the read-and-write consumption of the computing process. Instead, we will focus on the Assembly Code instructions utilised and the number of cycles needed for each line of code in order to estimate and compare time requirements with greater simplicity. The performance evaluation of analog designs will primarily be based on the specified characteristics as stated in the respective works.

In regards to the spatial aspect, it is relatively straightforward to estimate the overall footprint based on the simplicity of the design and the required technology. By analysing the FPGA specifications, we can determine the minimum number of transistors needed for a computing device, as well as calculate the portion of gate arrays utilised for the task. However, in cases where cutting-edge technology, particularly in analog-based designs, is employed, the mapping of multi-stack and cross-layer components to the silicon surface can vary significantly depending on the specific circumstances. This circumstance allows for the assessment of the dimensions of these components solely on the basis of their classification and quantity.

Therefore, we have chosen to focus on discussing the expected number of transistors per functional block and developing a formula to predict the scalability and growth of transistors necessary for each circuit involved in this project.

3.3.2 System and modular tolerance

The evaluation of a potential circuit design revolves around its ability to accurately replicate the output of a mathematical model when provided with the same input. This analysis focusses on the alignment of the physical system output, as measured by the voltage measured at its output port, with the anticipated output of the corresponding mathematical representation. Dimensional considerations are disregarded in this assessment for simplicity of discussion.

In application-orientated industries, it may not be advisable to assess the performance of a system by evaluating each individual sub-system. The notable attributes of neural networks, such as non-linearity and opacity, present significant challenges when conducting a thorough system-level analysis, particularly concerning accuracy and loss. This complexity makes it impractical and ineffective to rely solely on information regarding the network's architecture and parameter configurations, as a comprehensive understanding necessitates detailed data samples. In neural network with multiple hidden layers trained using the gradient descent algorithm and an activation function with a gradient not exceeding unity within its range, it is challenging to establish solid prediction that the enhancement of the precision of each component results in an overall improvement in systematic accuracy or performance at the neural network's level.

In order to establish a dependable benchmark for assessing modular performance, it is our belief that the tool should accurately depict the tolerance of each component within the design. This can be achieved through quantifiable measurements aligned with a straightforward and universally accepted standard that can be applied to any potential input and output scenarios.

Using a uniform approach to replicate the alignment between mathematical models and analog implementations, the sequential steps are outlined as follows:

1. Establish models and conduct simulations to assess the variability of individual components.
2. Develop models and conduct simulations to assess the variability of each layer within a specified configuration based on component variability.
3. Create models and conduct simulations to evaluate multi-layer and systematic variability through layer-by-layer analysis.
4. Derive models and perform simulations to assess performance based on a combination of component and signal variability.

The fourth stage, distinguished from its predecessors, stands alone and centres primarily on the characteristics of neural network. This stage will highlight certain aspects related to its optimisation and configurations, serving as a reference and providing insight into the practical application's scope, rather than focusing solely on the functionality of combinations of non-ideal components as represented by the mathematical model. Furthermore, when examining the network's robustness, in addition to analysing its resilience to external disturbances, such as noise and temperature fluctuations, it is essential to consider its ability to withstand variations in elements. The examination of external perturbations will be

touched upon in the relevant chapters.

In summary, when considering notations X; P; $Y = PX$; $\alpha(\cdot)$; or $Z = \alpha(Y)$ represent the input vector, linear transformation matrix, pre-activation value, activation function and activated value in ideal cases, respectively, and using the same symbols with an additional hat notation to denote their perturbed versions, we can evaluate and assess performance using the following equations:

$$\Delta X = \hat{X} - X; \tag{3.2}$$

$$\Delta Y = \hat{P}X - PX; \tag{3.3}$$

$$\Delta Z = \hat{\alpha}(Y) - \alpha(Y). \tag{3.4}$$

These equations quantify the discrepancies observed in the input stage (ΔX), caused by the linear transformation (ΔY) and introduced by the linear transformation and the activation function (ΔZ). These terms serve to illustrate the shortcomings that may arise in the implementation of each functional block compared to ideal theoretical models.

In order to assess the overall effectiveness of a neural network , it is possible to establish a target value, denoted as T, for the single-layer neural network. The general function of the neural network can be labelled as $f(\cdot)$, and its loss function can be defined as $\delta(\cdot)$. The network's performance in a purely theoretical context can be measured by $\delta(f(X), T)$, without taking into account any specifics of the actual system.

For instance, if we consider a single-layer perceptron as our model, we can calculate a series of altered outputs:

$$\Delta Z|_{\hat{X}} = \alpha\left(P\hat{X}\right) - f(X) \tag{3.5}$$

$$\Delta Z|_{\hat{P}} = \alpha\left(\hat{P}X\right) - f(X) \tag{3.6}$$

$$\Delta Z|_{\hat{\alpha}} = \hat{\alpha}(PX) - f(X) \tag{3.7}$$

These equations demonstrate the difference in output compared to the model due to variations in individual components. By analyzing these functions, we can gain a better understanding of potential errors introduced by different elements of the network. While these insights can help identify how the output may be distorted systematically, they do not necessitate an actual decrease in overall performance in practical cases.

In adherence to this approach, it is deemed necessary to conduct additional analysis on the impact of $\delta\left(\hat{f}\left(\hat{X}\right), T\right)$ on the systematic performance, with $\hat{f}(\cdot)$ denoting a distorted system. This quantity is intended solely for illustrative purposes in showcasing the alterations in network performance, without indicating directions for any ensuing corrections.

With the use of the benchmarks, we can effectively assess the deviation introduced into a system in comparison to its desired outcomes. However, this method is not ideal for a systematic assessment in the context of neural network, due to the non-polynomial nature of activation functions. These functions may produce similar outputs for different inputs, making it difficult to identify potential failures or errors in the input data.

With the use of Taylor expansion, with the gradient of the activation function $\alpha(\cdot)$ at the input point x denoted as θ, while disregarding the second and higher order terms, it leads to the emergence of an error term as a result of input perturbation between $\alpha(x + \Delta x)$ and $\alpha(x)$ represented by the equation:

$$\alpha(x + \Delta x) - \alpha(x) = \theta \Delta x, \tag{3.8}$$

where the value of θ falls within the range of 0 to 1 by designs for the activation function, ensuring that it does not exceed the maximum magnitude of the gradient between x and $x + \Delta x$. This principle can further be supported by a more broad interpretation of Lagrange's mean value theorem, which asserts that:

> "For a given planar arc between two endpoints, there is at least one point at which the tangent to the arc is parallel to the secant through its endpoints."

Based on this concept, it can be inferred that for a function with a gradient that consistently remains below an absolute value of 1 within its applied range, the secant between any two points is either less than or equal to 1. This function demonstrates that, when introducing a perturbation of Δx to the input, any discrepancy in the output of the system in comparison to its original output should converge after the activation process.

The efficacy of the activation functions can be compromised in non-optimal scenarios. This discrepancy may come from two main sources: inconsistencies between the theoretical activation function and the practical circuit transfer function, as well as the noise generated during operation. These perturbations are quantified using a linear operation to minimise a loss function as shown in Eq. (3.1), along with an additional relative error ϵ_α encompassing all potential inputs and a constant noise with magnitude ϵ_n. In the case of the activation functions analysed in this study, where the deviations between the original and implemented functions are minimal, our estimation approach has been validated as effective as will be demonstrated in Chapter 6.

Although some researchers in the field of computer science focus on analysing errors in floating point arithmetic within digital systems [171–173], the prevailing approach leans towards prioritising accuracy and security over storage concerns. This is mainly due to the redundancy built into the system and the limited impact of errors on applied neural networks [174, 175]. In contrast, in analog systems where inaccuracies can vary significantly from digital systems, there is a growing need to evaluate the capacity correcting errors. To meet the demand for a comprehensive framework for assessing the performance of individual submodules, a thorough analysis involving multiple layers is necessary. This holistic approach will reveal the extent to which each functional block introduces unwanted distortions, ensuring the validity of performance analysis of the HNN at a general level.

This thesis will primarily focus on evaluating the reliability and effectiveness of the implementations of each layer on the functioning of a HNN, with the aim of establishing standards that will allow researchers and industry professionals to develop and implement a sophisticated neural network on a silicon platform. In this study, we will operate under the assumption that any combination and magnitude of parameters are feasible and that the capacity to execute

any real-world task is determined by the mathematical model produced in the structure of the neural network. The objective of this proposed design is to accurately represent each necessary operation for the mathematical model with minimal deviation from the optimal output scenario.

The analysis of the performance of the circuit can be achieved through a careful examination of a neural network, assuming that it has undergone extensive training and reached a state-of-the-art level. Performance can be defined as a function of all potential ideal inputs X and their respective ideal outputs Y. We can denote the transfer function of the circuit as $\varphi(\cdot)$, with the loss function assumed to be the RMS of the probability density function (PDF) of the relative error between $\varphi(\cdot)$ and Y

$$\delta(\varphi(\cdot)) = \textit{RMS}\left(\frac{\varphi(X) - Y}{Y}\right) \tag{3.9}$$

The decision to not use non-systematic analysis when neural networks with a gradient descent-based optimiser experience decreased performance can be better explained by the lack of transparency without modular pre-training and additional fine-tuning with human intervention. Research has shown that while optimising neural network can yield satisfactory results, it falls short in pinpointing and explaining issues [176], often described as a " black box" [177] or "Matryoshka Doll" scenario [178]. This complexity hampers our ability to fully understand the inner workings of the system, posing challenges in evaluating the performance and effectiveness of individual components.

There is significant concern within the artificial intelligence community about the uncertainties surrounding potential risks and approaches to assess the performance of feature matrices and data representations used in training and applications [177–179]. As a result, there is a shortage of precise tools to accurately gauge the performance and learning efficiency of specific layers, their inputs, and outputs, utilising metrics like "accuracy" and "loss" commonly employed in systematic analyses.

In addition, due to constraints in data accessibility and challenges related to batch testing in SPICE software, it is exceedingly difficult to thoroughly test every sub-module of a proposed system with all conceivable combinations of input and output data in practice. It is imperative that the benchmark for testing is raised to a higher standard in order to guarantee the performance is maintained at a consistent level across a wide range of potential applications, whether they are actively being tested or not.

3.4 Conclusion

In order to successfully implement the proposed circuit with the HNN framework and accurately measure its performance relative to other SOTA works in various formats and configurations across multiple dimensions using a variety of datasets, we have established specific criteria and methodologies for conducting these comparisons.

The methodology is grounded in our strategic objective of developing a modular system that incorporates a standardised framework for information trans-

mission and processing. This approach entails the translation of the conceptual model of the neural network into a corresponding hardware implementation. Furthermore, we evaluate the coherence between the proposed hardware design and the underlying mathematical model.

In this section, we have outlined the key criteria deemed essential for the implementation, along with detailing the methodology used for measurement and estimation from a modular foundation to a systematic approach. In order to address the clarity issue identified, we have clarified that performance evaluations are to be made relative to the neural network being represented by the system, with all comparative analyses conducted in a context-neutral environment devoid of practical application considerations.

The terminology and classifications of signals will serve as a framework for the work presented in Chapter 4 and 5. Additionally, the criteria established for assessing the performance of a specific HNN will be utilised in the subsequent three chapters. This approach will facilitate a more comprehensive comparison with other widely-used activation functions or SOTAstechniques, thereby ensuring a robust evaluation of the learning outcomes and resilience of systems employing these technologies.

Chapter 4

Activation Function Circuit

We are to introduce a hardware neuron that has the capability to generate a wide range of activation functions. Strict requirements for the explicit form of activation functions within neural networks are not necessary [9, 10]. Through the use of standard transistors, a variety of non-polynomial transfer functions can be easily achieved to be utilised as activation functions.

In our study, we discuss the similarities and differences between hardware versions and similar mathematical activation functions. Two hardware neural networks (HNNs) examples have been created based on the design of the hardware neuron. These networks have been thoroughly evaluated for regression and image classification tasks.

It has been determined that the accuracy and reliability of the HNNs are comparable to their software counterparts, with only slight performance degradation when considering input perturbations and component variability. This neuron, compatible with complementary metal oxide semiconductor (CMOS) technology, paves the way for easily manufacturable HNN . This provides a practical, low-power platform together with low power multiply accumulate circuits (MACs) for incorporating HNNs into applications such as the Internet of things (IoT) and systems on a chip (SoC).

4.1 Overview

The computational model formalised as artificial neural networks (ANNs) serve as the foundational framework for the majority of contemporary machine learning (ML) studies and applications [180]. These neural networks have found application in a diverse array of fields, including but not limited to computer vision, speech recognition, automatic control, finance, and robotics. The fundamental objective of utilising neural network in such contexts is to approximate an unknown function, a task facilitated by the theoretical understanding of the universal approximation properties inherent to neural networks. In a simplified explanation, the universal approximation theorem entails that for any function f within a certain class F and any desired level of precision ε there exists a neural network N capable of approximating the said function f to within the specified ε degree of accuracy. Various results of this kind are available in the literature that verify the universal approximation capabilities of neural networks across a wide range of practical functions, albeit with certain conditions often

imposed on the activation functions of the networks to ensure this universal behaviour [181–183].

4.1.1 Background: Hardware neural computing

Although not longer strictly following the cadence outlined by integrated circuit (IC) advancements that serve as the foundation for computing systems, ICs are consistently decreasing cycle time and energy consumption for a floating-point operation (FLOP) [184]. When it comes to the software execution of a neural network model, the associated computational overheads—such as memory usage, compute time, and power consumption—in relation to the network's performance can vary significantly across the wide variety of available hardware platforms. Consequently, evaluating the comparative energy efficiency of the software implementation of a neural network is not straightforward [185].

Software-based neural networks are typically implemented on machines with a von Neumann architecture. Such an implementation involves the compilation of the code from a higher-level programming language to low-level machine instructions suitable for the underlying platform. These instructions are then executed on a circuit that is designed to handle a much wider class of problems. Furthermore, the separation between processing and memory in a von Neumann model leads to a data latency and energy cost that can exceed the time and energy needed to perform a single operation by several orders of magnitude [186].

An alternative is to map the neural network—which requires no separation between processing and memory—directly to a circuit architecture. This approach naturally leads to an implementation using hardware neural network (HNN) systems, especially circuits with analog electronic realisations [187]. HNNs have been developed for many decades [47] with applications including signal processing [188–191] and control automation [192–194]. These works follow varying approaches to the design of the weighting, summation and activation function functional blocks. A key goal for the design of HNN is to boost the performance of a neural network relative to the equivalent software designs performed on conventional computing platforms. Depending on the application, improved performance can be achieved in terms of reduced computation time, lower power, higher reliability, reduced form factors for mobile and embedded applications, and enhanced security [47].

Many applications of neural networks do not require a large number of neurons per layer, thereby admitting circuit layouts compatible with conventional complementary metal oxide semiconductor (CMOS) circuits. HNNs fabricated with conventional CMOS solutions offer attractive features for integration with embedded systems and particularly for systems on a chip (SoC) applications. Many SoC applications—as for most electronics applications—require trade-offs between high frequency and power efficiency. Low-power neurons are especially appealing for applications that use autonomous distributed systems or the Internet of things (IoT). For IoT applications, the demand to perform complex tasks locally is at odds with the low energy requirements for each component in a network, whereas the speed of operation may not be a critical factor.

4.1.2 Motivation: Efficient Look-Up Table

In an analog HNN, circuit configurations mimic the architecture of the neural network. Similarly, as a combination logic circuit is implemented as interconnected **Gates**, the HNN architecture consists of interconnected neurons. Some notable advantages that are gained include:

1. reduction of energy costs for performing the operations required of a neuron;
2. elimination of circuit redundancy compared to, for example, a processing unit or a micro-controller;
3. limiting latency to the propagation delay through each neuron layer.

However, there are also some disadvantages associated with HNNs as well. First, precision is lost in analog computations compared to binary computations. Furthermore, the fan-out is limited by the interconnect technology used to manufacture a HNN, and hence scalability becomes an issue with an increasing number of neurons in a layer. As such, for the foreseeable future, software implementation will remain the preferred approach for applications requiring high precision and / or a larger number of neurons per layer.

The hardware design of the activation function may be approached from at least two angles. One approach is to first select an activation function as the target, then develop a circuit to mimic that function [195–197], and finally implement a neural network design for integrated circuits [64, 198, 199]. This is justified by the aforementioned universal approximation results in neural networks. A somewhat opposing approach is to take advantage of non-polynomial devices such as transistors to explore which non-polynomial functions are easily reproduced by a circuit, and to exploit non-polynomial functions that are intrinsic to the operation of a circuit. In this study, the latter approach is followed, which allows a large class of viable activation functions.

Against this background, a neuron is studied that is fully CMOS compatible, can be designed to work in the weak inversion or sub-threshold region for low power conditions, is simple from a circuit viewpoint, and can be adapted to provide tailored activation functions for different applications. The starting point of the study is the voltage-to-voltage transfer function for a simple two-transistor circuit. The transfer function produced by the circuit, while exhibiting non-polynomial characteristics, has the potential to function effectively as an activation function in practical neural network applications. More broadly speaking, the circuit is capable of offering a range of activation functions that can be examined as a circuit-based representation of a piece-wise linear (PWL) function. The circuit implementation of the activation function generator will then be demonstrated to have advantages for the HNN design. The variability in the electronic components providing the summation and weighting functions also influences the overall performance of a neuron, and as a consequence, that of the HNN. Hence, for representative HNNs applied to a simple regression problem and for a classification example using the Modified National Institute of Science and Technology (MNIST) dataset, robustness against input perturbations and stability against component variations are analysed.

4.2 Push-Pull Linear Follower

The foundation of a HNN is a neuron circuit that is integrated with an interconnect scheme to facilitate the construction of a network. A straightforward implementation of neurons can be achieved with two main functional blocks: an activation function circuit (AFC) and a MAC. These components work together to execute the non-linear and linear operations mandated by the various layers of a specified neural network.

In the face of the challenging task of analytically modelling the behaviour of transistors and other active elements in the field of electronic engineering, which often entails a large number of parameters [200–203], it could lead to difficulties in designing a circuit to replicate the behaviour of any widely used activation functions. Nevertheless, recognising that any non-polynomial functions can serve as activation functions, it is possible to create a custom circuit with a non-polynomial transfer function to serve as a look-up table (LUT) of an activation function, and construct neural networks based on this approach.

4.2.1 Circuit diagram

The inherent non-polynomial nature of the transfer function of active components has proven to be a suitable choice for serving as the generators of activation functions in a neural network [47].

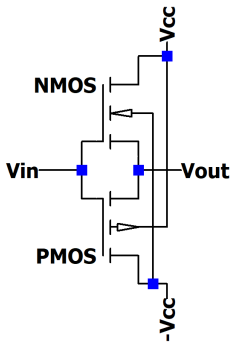


Figure 4.1: The basic diagram illustrating the complementary metal oxide semiconductor-based activation function circuit. The circuit is powered by two balanced voltage sources referred to as **Vcc** and **-Vcc**. The signal produced and analysed in the multiply accumulate circuit (MAC) is fed into the input port **Vin** which includes the common **Gate** of the transistor pair. The resultant output can be retrieved from the shared **Source**, denoted as **Vout**.

The proposed circuit for the AFC of the HNN system features a straightforward pair of symmetrically matched CMOS devices depicted in Fig. 4.1. This configuration serves as a means to deliver a range of activation functions. The CMOS pair-based design showcases a topology known as a “single stage CMOS buffer”, where the input signal is applied to the common **Gate** of the two transistors while the output signal is extracted from the **Source**. This setup results in a consistent relationship between the input voltage **Vin** and output voltage **Vout** , as evident from the circuit operation. The design, while yielding a comparable

transfer function to that presented by Khanday et al.in their research [204], features a significantly more streamlined architecture. This simplification not only reduces the silicon footprint but also facilitates easier wiring configurations.

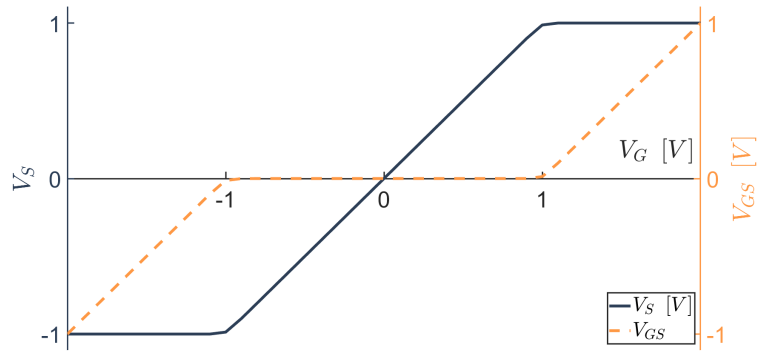
In the typical operation of a CMOS design, it can be anticipated that only one transistor will be in the active state, facilitating the flow of current. This transistor will demonstrate significantly higher conductance than the other transistor, thus exerting a strong influence on the output voltage, usually directed towards one terminal, specifically **Vcc** or **-Vcc**. As the **Gate** voltage aligns with or surpasses the level of the **Drain** , the transistor can be effectively represented as diode-connected. Subsequently, the output voltage stabilises at the **Drain** voltage, denoted as V_D.

An analysis of the relationship between the voltage applied to the **Gate** of the CMOS pair illustrated in Fig. 4.1 and the resulting voltage observed at the **Source** , as well as their discrepancy, is depicted in Fig. 4.2a based on the mathematical model obtained in the subsequent section and verified with results of simulation program with integrated circuit emphasis (SPICE) simulations utilising the predictive technology model (PTM) as shown in Appendix B. As has been illustrated in Fig. 4.2a, the voltage following nature of the circuit results in a zero **Gate**-**Source** voltage ($V_{GS} = 0$) and facilitating sub-threshold operation at the centre of the graph. As the input voltage V_i (denoted by V_G) approaches the **Drain** voltage of either the N-Channel metal oxide semiconductor or P-Channel metal oxide semiconductor transistor (V_D^n or V_D^p, respectively), one of the transistors will reach a zero **Drain**-voltage, shown by the orange curve, follows the relationship $V_{GS} = V_G - V_D^{p/n}$ in the saturated region when the complementary metal oxide semiconductor pair is locked, otherwise it remains close to zero. It is important to note that the N-Channel metal oxide semiconductor (N-MOS) and P-Channel metal oxide semiconductor (P-MOS) transistors cannot be operated concurrently above the threshold voltage with the design.

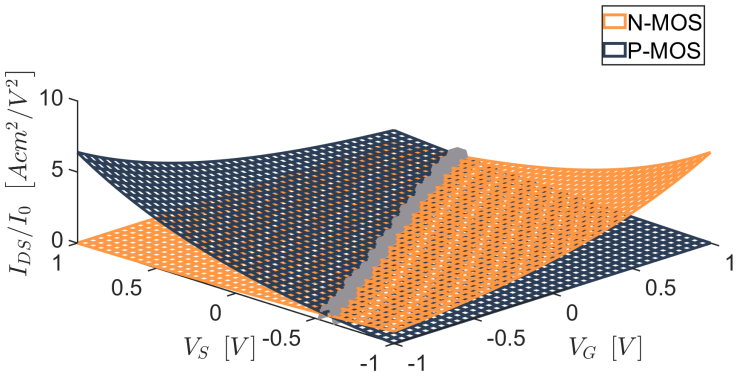
By understanding the transfer function in relation to the **Gate**-**Source** voltage V_{GS} applied to the field effect transistors (FETs) and its conductance, the presence of two saturation regions at the end of the curve of the **Source** The voltage V_S can be simply elucidated. One transistor operates in an above-threshold region where resistance is minimal, while the other operates in the depletion region with significantly higher resistance. In such a scenario, V_{GS} will be drawn towards the corresponding voltage supply level (**Vcc** or **-Vcc**).

In the operational range of the transistors, as indicated by the simulation results, both transistors are functioning in a sub-threshold state. In this sub-threshold state, the conduction effects can be described by the equation $I_D = I_0 \exp \frac{V_{GS}}{\xi V_T}$ [205]. Here, I_0 represents the current pre-factor, calculated as $I_0 = 2\mu C_{ox} V_T^2 \frac{W}{L}$, with μ_n denoting electron mobility and C_{ox} referring to **Gate** capacitance specific to the fabrication technology being utilised. The aspect ratio $\frac{W}{L}$ is determined by the dimensions of the transistor's **Channel** (width W and length L).

For a symmetrically matched pair of CMOS transistors in this study, the pre-factor remains constant, as long as the condition $\frac{W_p}{W_n} = \frac{\mu_n}{\mu_p}$ is met. The non-ideality factor is denoted as ξ and the thermal voltage V_T can be expressed by $k_B T/q$, where k_B represents the Boltzmann's constant, T is the temperature and q is the charge of an electron, which is approximately $26\ mV$ at room tempera-



(a)



(b)

Figure 4.2: The study presents an analytical model of the activation function circuit with a discrete matched pair, derived from the mathematical framework discussed in Section 4.3. (a) The transfer function illustrated by the dark curve demonstrates the correspondence between the output voltage V_o, denoted as V_S, and the input voltage V_i, denoted as V_G, for a matched transistor pair. The voltage difference between V_i and V_o remains at zero, except in instances where V_i approaches the **Drain** voltage of either transistor. (b) The relationship between the **Drain-Source** currents I_{DS} and the current pre-factor I_0 for the individual transistors is depicted for varying values of V_G and V_S (N-MOS represented by the orange surface and P-MOS by the dark surface), with fixed **Drain** voltages set at $V_D^n = -V_D^p = 1\ V$ for the N-MOS and P-MOS transistors, respectively. The intersection of the two surfaces (highlighted with grey meshes) at the centre of the graph indicates the condition $I_{DS}^n = -I_{DS}^p$ The solid grey mesh also represents the condition where $V_o \approx V_i$, aligning with the mathematical model of the AFC.

ture, as is applied in this work.

When the gate-source voltage V_{GS} is fixed and consistent for both transistors, the current flowing through the system remains steady, in line with Kirchoff's current law, which dictates that:

> "The algebraic sum of currents in a network of conductors meeting at a point is zero."

As a result of the circuit configuration and the analysis conducted, it is deter-

mined that the current flow through both transistors will be consistent and equal, based on the aforementioned analysis and circuit configuration.

4.2.2 Non-linear transfer function

The utilisation of a circuit to represent the activation function $\alpha(\cdot)$ can be achieved using the voltage-based transfer function $\varphi(\cdot)$ of the proposed circuit. This function produces an output that is saturated for inputs below a lower threshold, acts as a voltage follower within a specified range, and returns to a saturated state for inputs exceeding an upper threshold. The configuration depicted in Fig. 4.1, showcasing a two-transistor setup, will be showcased as meeting these criteria successfully.

The circuit arrangement comprises a single pair of CMOS transistors, designed with a topology similar with a conventional CMOS push-pull follower. However, a key distinguishing feature is that the transistors in the inverter have been exchanged. The AFC is structured in a manner ensuring that the **Drain** of the N-MOS transistor (**Vcc**) is maintained at a voltage V_D^n, while the **Drain** of the P-MOS transistor (**-Vcc**) is set at a voltage of V_D^p. The **Gates** of both transistors are connected to the input voltage V_i, while the output voltage V_o is derived from the shared node of the two **Sources**.

In the context of simulations discussed in Li et al.'s work [64], employing voltage supplies of $V_D^n = 1.8\ V$ and $V_D^p = 0$, with threshold voltages for both transistors to be $V_{th}^n \approx |V_{th}^p| \approx 0.5\ V$, the operation takes place in the weak inversion or sub-threshold region. The circuit is then tested within an input range of $-1.8\ V \leq V_i \leq +1.8\ V$, illustrating a resulting voltage transfer function closely approximating *ReLU* function.

The results of the simulation above suggest that the selection of the voltage supplies V_D^p and V_D^n corresponds to the allocation of values to x_- and x_+ in a PWL function as will be shown as Eq. (4.13). Next, an analytical model based on metal oxide semiconductor field-effect transistor (MOSFET) operation in weak inversion and incorporating the transition to a blocked state will be developed. The utilisation of the analytical model will elucidate the degree to which the voltage transfer function of the AFCs replicates the different functions outlined by the PWL model. Additionally, it will offer straightforward relationships to illustrate how the mismatch and temperature variation of the transistors affect the performance of the activation function circuit (AFC).

In the weak inversion region, the **Drain** current for a N-MOS transistor can be approximated by:

$$I_D^n = I_0^n \exp\left(\frac{V_G - V_{th}^n}{V_T}\right) \times \left(\exp\left(-\frac{V_S}{V_T}\right) - \exp\left(-\frac{V_D}{V_T}\right)\right). \tag{4.1}$$

The aforementioned equation showcases the relationship between a pre-factor denoted as I_0^n, the threshold voltage of a N-MOS transistor known as V_{th}^n, and the thermal voltage indicated as V_T. By utilising a specific pre-factor denoted as $\beta_n = I_0^n \exp(-V_{th}^n/V_T)$, the **Drain** current can then be reformulated accordingly as:

$$I_D^n = \beta_n \exp\left(\frac{V_G - V_S}{V_T}\right) \times \left(1 - \exp\left(-\frac{V_D^n - V_S}{V_T}\right)\right). \tag{4.2}$$

For the cases $V_{DS}^{n} \gg V_T$, Eq. (4.2) can be approximated as:

$$I_D^n = \beta_n \exp\left(\frac{V_{GS}}{V_T}\right). \tag{4.3}$$

A similar description for a P-MOS transistor in the sub-threshold region can be written as:

$$I_D^p = -\beta_p \exp\left(-\frac{V_{GS}}{V_T}\right). \tag{4.4}$$

The impacts of drain induced barrier lowering (DIBL) and the body effect may be considered [206] when examining the variation in the current pre-factor with **Drain-Source** biasing and **Source-Bulk** bias. These supplementary effects do not have a meaningful impact on the conclusions drawn from the basic analytical model. This conclusion will be further validated through circuit simulation using compact models that account for additional effects not addressed in the analytical model or the analysis of individual transistors to be presented in Section 4.3.

Upon observing that both transistors are functioning within the sub-threshold range, the current flow through a CMOS pair of transistors is adjusted to be equal by utilising equations Eq. (4.3) and Eq. (4.4). This analysis yields the conclusion that the stated condition is as follows:

$$\exp\left(\frac{V_o - V_i}{V_T}\right) = \exp\left(\frac{V_i - V_o}{V_T}\right) \tag{4.5}$$

must be met, where V_i and V_o refer to the input and output voltages applied and measured at the **Vin** and **Vout** ports respectively. This equation was referenced in a previous study by Li et al. [64] to explain the results of their circuit simulation. In order for the condition $V_o = V_i$ to be satisfied, the circuit's operation is limited to functioning as a voltage follower for values of V_i within the range of $V_D^p \lessapprox V_i \lessapprox V_D^n$. This means that the unity gain area of the AFC offers a straightforward, energy-efficient voltage follower with broader utility for sub-threshold circuit designs. However, it is important to acknowledge that Eq. (4.5) cannot accurately depict the output voltage as it approaches either V_D^n or V_D^p, or when the activation function reaches its saturation.

The circuit is then examined under the condition that the output satisfies $V_D^p \ll V_o \approx V_D^n$. In this case, having the same current through the two transistors implies that the following condition must be met for a matched complementary pair of transistors:

$$\exp\left(\frac{V_i - V_o}{V_T}\right) \times \left(1 - \exp\left(-\frac{V_D^n - V_o}{V_T}\right)\right) = \exp\left(\frac{V_o - V_i}{V_T}\right). \tag{4.6}$$

This equation may be arranged to give:

$$V_o = V_i - V_T \sinh^{-1}\left(\frac{1}{2}\exp\left(+\frac{V_i - V_D^n}{V_T}\right)\right), \tag{4.7}$$

where $\sinh^{-1}(\cdot)$ is the inverse hyperbolic sine. It is noted for large values of the

input V_i that

$$\exp\left(-\frac{V_i - V_D^n}{V_T}\right) \approx 0, \tag{4.8}$$

allowing Eq. (4.7) to be approximately written as:

$$V_o \approx V_i - V_T \sinh^{-1}\left(\sinh\left(\frac{V_i - V_D^n}{V_T}\right)\right) = V_D^n. \tag{4.9}$$

Hence, as the input exceeds V_D^n by several multiples of $k_B T/q$, the output saturates to V_D^n. A similar analysis can be performed for the condition $V_D^p \approx V_o \ll V_D^n$ that leads to:

$$V_o = V_i - V_T \sinh^{-1}\left(\frac{1}{2}\exp\left(-\frac{V_i - V_D^p}{V_T}\right)\right). \tag{4.10}$$

In this scenario, the output voltage reaches saturation at the lower value of the power supply, denoted V_D^p. Given that the second term in Eq. (4.7) significantly deviates from zero only for certain values of V_i that render the second term in Eq. (4.10) effectively zero, and vice versa, it is permissible to consolidate the two equations to derive the transfer function:

$$V_o = \varphi(V_i) = V_i - V_T\left(\sinh^{-1}\left(\frac{1}{2}\exp\left(+\frac{V_i - V_D^n}{V_T}\right)\right) - \sinh^{-1}\left(\frac{1}{2}\exp\left(-\frac{V_i - V_D^p}{V_T}\right)\right)\right), \tag{4.11}$$

The function $\varphi(\cdot)$ denotes the transfer function of the circuit under consideration. After thorough measurements, it has been determined that this function can be effectively approximated as a PWL function operating within the domain of $\mathbb{R}$, which be expressed in terms of an input variable v as follows:

$$\varphi(v) = \begin{cases} \textbf{Vcc}, & v < \dfrac{\textbf{-Vcc}}{\theta} \\ \theta v, & \dfrac{\textbf{-Vcc}}{\theta} \leq v \leq \dfrac{\textbf{Vcc}}{\theta} \\ \textbf{-Vcc}, & v > \dfrac{\textbf{Vcc}}{\theta} \end{cases} \tag{4.12}$$

The parameter θ represents the deviation of the actual slope of the transfer function, which satisfies $\theta \lessapprox 1$. The non-unity slope is a result of a **Bulk** leakage path from **Vcc** to **-Vcc**. Using ideal components with the silicon on insulator (SOI) substrate or by properly biasing of the leakage path, we can achieve a slope closer to unity in the linear region. Additionally, we can amplify the pre-activation input to optimise the performance as observed at the input side of the circuit.

The approximation demonstrates its accuracy in the full range of input voltages V_i when reasonable choices are made for the voltage supply and threshold voltages. This model elucidates how a matched complementary pair serves as an approximation for the set of PWL functions categorised by the activation

function $\alpha(\cdot)$ given in Eq. (4.13). Furthermore, it enables a description of how the transfer function may deviate from the PWL expression due to transistor mismatch and temperature effects. It is observed that the circuit transfer function and the PWL model for an ideal matched pair show notable differences only within a range extending a few multiples of the thermal voltage V_T about V_D^n and V_D^p.

The difference between the simulation output of the transfer function and the PWL model is most appreciable in the transition regions between the saturated and linear regions, but the simulation and model are nearly indistinguishable for this transistor pair and at the chosen scale if proper scaling is made according to the practical-case transfer function generated.

4.3 Activation Function Analysis

In the realm of neural networks, there is a range of non-polynomial activation functions that can ensure the universal approximation property. However, it has been observed that certain activation functions may be more efficient in learning efficiency or final outcome than others for specific applications.

It should be noted that incorporating functions such as the hyperbolic tangent through traditional active electronic components can lead to a significant increase in the complexity of designing a hardware neuron [200–203]. As an alternative solution, various non-polynomial transfer functions can be generated by a single CMOS circuit. This is made possible by making simple design choices for a two-transistor circuit, allowing it to produce transfer functions that encompass a variety of PWL functions that such as *ReLU*, *ReLU6* [207], and hard hyperbolic tangent ($\mathrm{tanh_H}$) [208]. A comprehensive list of representative functions related to this circuit can be found in Tab. 4.1.

Although the specific form of an activation function is not necessarily critical for a neural network, there are distinct advantages to using certain functions over others in practical applications. For instance, choosing *ReLU* over $\mathrm{tanh_H}$ or vice versa. can lead to improved performance in specific scenarios. Furthermore, it is important to note that a circuit does not need to perfectly replicate a desired activation function as long as its learning outcomes and some specific characteristics closely approximate the intended function.

4.3.1 Comparison with other activation functions

Prior to delving into the characteristics of the hardware neuron, a preliminary PWL function representing the activation function that the circuit aims to replicate is presented as Eq. (4.12), with the adjustment of the factor θ adjusted to be 1, is shown as follow:

$$\alpha(x; x_-, x_+) = \begin{cases} x_-, & x < x_- \\ x, & x_- \leq x \leq x_+ \\ x_+, & x > x_+ \end{cases} \tag{4.13}$$

with $x_- < x_+$. The mathematical activation function $\alpha(x)$ exhibits a linear relationship with a slope of one between the lower threshold x_- and the upper threshold x_+. It should be noted that the input range of x may exceed the spec-

ified range of $[x_-, x_+]$. When the argument $x < x_-$ or $x > x_+$, the function reaches a saturation point at x_- and x_+, respectively.

In the case of $x_+ = -x_- = 1$, the function transforms into a hard hyperbolic tangent function known as $\tanh_{\mathrm{H}}$. The function $\tanh_{\mathrm{H}}$ is an odd function that saturates at extreme positive and negative values of the input. It is monotonically increasing, differentiable in most of its domain of definition, and has a slope of one near the origin, similar to the function $\tanh$. The same reasoning applies to equations Eq. (4.13) leading to PWL functions for *ReLU* and *ReLU6* with $x_- = 0$ and $x_+ \to +\infty$ or $x_+ = 6$ respectively.

Generally, the activation functions generated by Eq. (4.13) share in common that they are PWL, monotonically increasing, and have a derivative that yields a slope of unity within the range of $x_- \leq x \leq x_+$. This property has been shown to provide an advantage in learning efficiency [209].

Table 4.1: Some representative activation functions, the functions discussed in the comparation of the transfer function-based activation function are highlighted.

Activation Function	Mathematical Expression	Range
Sigmoid	$\frac{1}{1+e^{-x}}$	$(0, 1)$
tanh	$\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$(-1, 1)$
$\mathbf{tanh_H}$	$\max(-1, \min(x, 1))$	$[-1, 1]$
ReLU	$\max(0, x)$	$[0, +\infty)$
ReLU6	$\max(0, \min(x, 6))$	$[0, 6]$
Softplus	$\log(1 + e^{x})$	$(0, +\infty)$
Softsign	$\frac{x}{1+\lvert x\rvert}$	$(-1, 1)$

The AFC is simulated utilising a generic $45\ nm$ low-power CMOS technology node described by compact models from the PTM repository [210]. The low-power $45\ nm$ compact models are optimised for a norminal power supply voltage of $V_{DD} = 1.1\ V$ in the model, offering valuable insights into the functionality of the AFC for contemporary technology nodes that operate at lower voltages. This includes addressing short-channel effects and substrate leakage issues.

The transistors within the AFCs are selected for optimal performance, with gate lengths set at $L = 45\ nm$ and widths configured at $W_n = 140\ nm$, and $W_p = 450\ nm$. The threshold voltages for the N-MOS and P-MOS are chosen to be equal with $V_{th}^n = |V_{th}^p| \approx 0.42$ V which together with the selection of the transistor widths approximates a matched integrated complimentary pair.

In Fig. 4.3, the voltage transfer function for the PTMs simulation and analytical model, as well as the scaled functions $\tanh_{\mathrm{H}}$ and $\tanh$ are presented. The

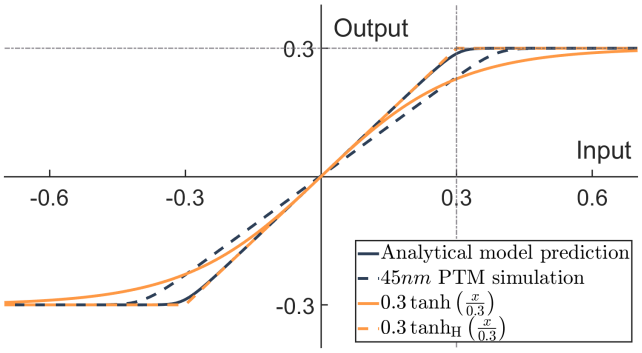


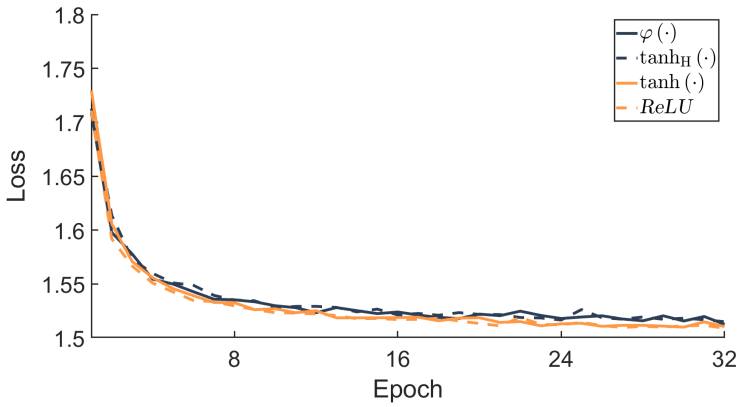
Figure 4.3: Analysis of the transfer characteristics of the activation function provided by the transfer function obtained through the $45\ nm$ predictive technology model (PTM) simulations with unit $[V]$ (dark solid line). This includes comparisons with activation functions derived from a simplistic analytical model by Eq. (4.11) (dark dashed line), a hyperbolic tangent function scaled by a factor of 0.3, formulated as $\tanh(x) = 0.3\tanh(x/0.3)$ (orange solid line) and a scaled hard hyperbolic tangent function with the same factor given by $V_o = 0.3\tanh_{\mathrm{H}}(x/0.3)$ (orange dashed line). The curves are examined in the context of activation functions, and therefore are dimensionless.

power supply voltages have been selected to be $V_D^n = |V_D^p| = 0.3\ V$, while the input voltage is systematically adjusted within the range between $\pm 0.7\ V$.

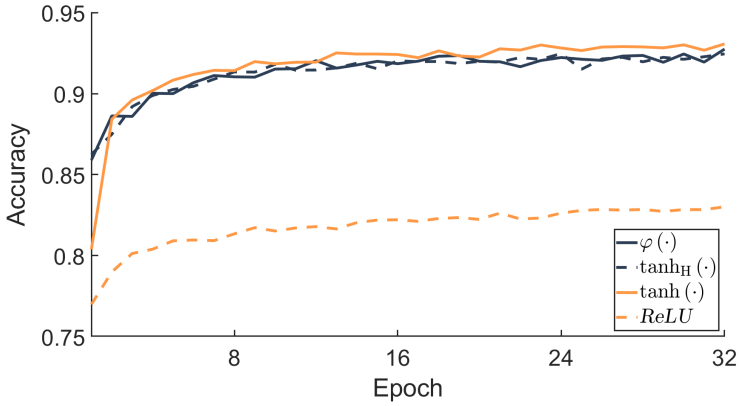
Upon initial examination of the analytical model at lower voltages, it is evident that the reduction in voltages leads to a smoothing of the transition to saturation due to a proportional rise in areas influenced by Boltzmann factors. However, the slope at $V_i = 0$ remains very close to one. In contrast, the PTM simulations exhibit a much more pronounced smoothing near the transition point, with a deviation from unity in the central region in the relationship between input and output. These deviations can be attributed to various factors such as DIBL, substrate leakage, and **Source / Drain** resistances that were considered in the integrated pair simulations but were not accounted for in the simplified version of the analytical model. Importantly, these factors do not have a significant impact in the simulations of matched discrete pairs.

In light of the impacts attributed to a smaller technology node, it is observed that the activation function generated by the integrated AFCs yields an approximation closely resembling a hyperbolic tangent function $\tanh(\cdot)$ in a crude manner. Therefore, it is reasonable to anticipate that both will demonstrate comparable learning efficiency when incorporated into neural networks.

The activation function can subsequently be integrated into a neural network. The findings of two separate experiments will be presented, one focused on a regression task and the other on image classification using the MNIST dataset. In addition, identical networks with the same structure and parameters were trained for both tasks using alternative activation functions, *i.e.*, $\tanh_{\mathrm{H}}$, $\tanh$, and *ReLU* for comparison purposes. All of these activation functions are increasing monotonically and have derivatives near the origin that resemble the identity function, leading to improved learning efficiency for the network [209].



(a)

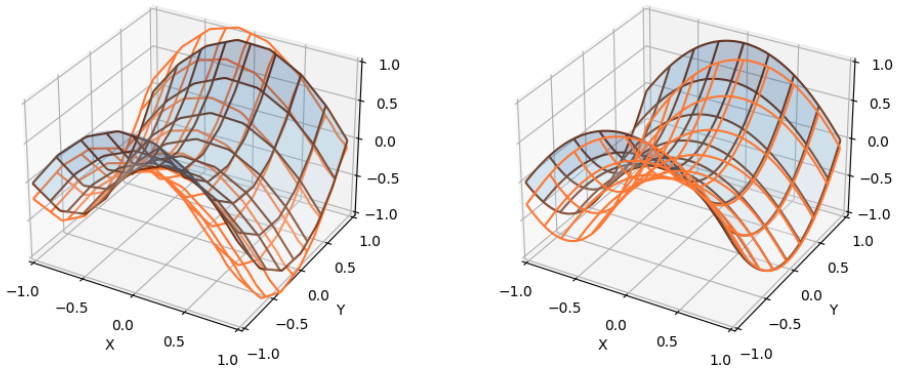


(b)

Figure 4.4: The assessment of loss and accuracy on validation sets using various activation functions. This includes the simulated transfer function $\varphi(\cdot)$ (black solid lines), $\tanh_H(\cdot)$ (black dashed lines), $\tanh(\cdot)$ (yellow solid lines) and *ReLU* (yellow dashed lines). (a) The loss obtained on validate sets for different activation functions implemented in the same neural network architecture with the same training hyper-parameter set. (b) The accuracy obtained on validate sets for different activation functions implemented in the same neural network architecture with the same training hyper-parameter set. It is evident that the *ReLU* based network exhibits lower validation accuracy compared to others, even the validation loss remains at a same level.

Utilising the parameters derived from the software model, the resulting HNN is constructed and implemented using LTspice. The output of each neuron within the HNN are validated against the trained software model. The training and validation outcomes for the MNIST dataset can be observed in Fig. 4.4. It is evident that the behaviour of the circuit transfer function closely resembles that of $\tanh_H$ and $\tanh$ functions.

In the scenario illustrated in Fig. 4.5, a regression task was carried out to approximate a hyperbolic paraboloid $f(x, y) = x^2 - y^2$ within the range of $-1 \leq x, y \leq 1$. This task was accomplished using a neural networkconsisting of a single hidden layer with 16 neurons. The network was fully connected with a two-neuron input and a one-neuron output. The activation function used in this network were the proposed circuit transfer function $\varphi(\cdot)$, $\tanh_H(\cdot)$, $\tanh(\cdot)$ and



(a) (b)

Figure 4.5: The learning outcome (dark brownish mesh) and output of a distorted network with a Gaussian noise of 3% the original values introduced (orange mesh) of (a) Circuit transfer function-based network and (b) $\tanh$-based network. The transparent blueish shaded surface indicates the target plane.

ReLU.

Table 4.2: The validate loss in L1 norm on simple regression task of $f(x, y) = x^2 - y^2$ with Gaussian noise added to the weighting parameters in a trained model when different activation functions are applied. For example, the row with $\varepsilon = 0.10$ corresponds to the noise tolerances with standard deviation of 10%.

	$\varphi(\cdot)$	$\tanh_{\mathrm{H}}(\cdot)$	$\tanh(\cdot)$	*ReLU*
$\varepsilon = 0$	0.0089	0.0089	0.0015	0.0069
$\varepsilon = 0.01$	0.0449	0.0276	0.0392	0.0178
$\boldsymbol{\varepsilon = 0.03}$	0.1654	0.1751	0.1704	0.1476
$\varepsilon = 0.10$	0.2375	0.2588	0.5411	0.2906

Through this approach, a series of well-trained models were obtained, with associated losses of 0.0089, 0.0089, 0.0015, 0.0069 in the L1 norm as shown in Tab. 4.2. It should be noted that the function $\tanh(\cdot)$ is found to yield the most favourable results due to its continuous nature. The other three functions perform similarly in terms of accuracy, showing results that are comparable to both the $\tanh(\cdot)$ function and the target surface.

Variations in parameters following a Gaussian distribution can have a significant impact on the output of a network. In the absence of such perturbations, the simulations shown in Tab. 4.2 indicates a consistent learning efficiency across different activation functions.

When subjected to perturbations by introducing a Gaussian noise whose root mean square (RMS) value equivalent to 3% of the original parameter values onto the weight components in the form will be shown in Eq. 4.19 and Sec-

tion 5.5, the simulated loss values shifted to 0.1654, 0.1751, 0.1704, 0.1476, respectively. This similarity in the results indicates that all four activation functions exhibit a similar level of robustness.

Analysis of the results presented in the Tab. 4.2 reveals that, as perturbations on the circuit components increase in magnitude, the $\tanh$ network experiences the highest increase in loss. Consequently, we can assert that the implementation of the transfer function generated by the proposed AFC as the activation function of a neural network in practical applications is unlikely to lead to a substantial reduction in robustness or learning efficiency.

It has been observed that the introduction of Gaussian noise directly to the components representing parameters of the network can have a significant impact on the output, particularly if the network is well-trained. Furthermore, the relationship between the additional loss experienced when a consistent level of noise is added and the specific activation function utilised is not easily discernible at this stage.

4.3.2 Special applications

Capitalizing on the strong linearity present in the non-saturation region of the circuit design, we are able to enhance efficiency when carrying out specific tasks such as recursive neural networks (RNNs) and residual neural networks (ResNets). This allows us to effectively maintain the input information within a specified range by executing linear operations on the relevant portions, or by isolating the desired segment while filtering out extraneous information through the saturation regions.

Using a hard hyperbolic tangent function $\tanh_{\mathrm{H}}$ proves to be the optimal choice for this process. The activation function produced by the transfer function of our proposed circuit is able to operate with a consistent efficiency as $\tanh_{\mathrm{H}}$ within the majority of non-saturation regions. There are isolated instances where the function may have limitations, specifically in areas near the transition points where neither $|V_{DS}| \gg V_T$ or $|V_G| \gg |V_D|$ conditions are satisfied, as well as in the two saturation regions.

In this section, we will demonstrate the capabilities of PWL functions using an experiment that converts an input within the range of $[-1, 1]$ into its Gray Code binary representation. By analytically assignment of the parameters, it is determined that the system must accurately process data points that meet specific criteria, allowing them to progress to the next layer without being altered. Any data points that do not meet these criteria will be disregarded and set to constant values.

In this particular scenario, as depicted in Fig. 4.6, the range of input values ranging from -1 to 1 is discretized into four segments, between ranges divided at the points -0.5, 0 and 0.5. Each segment contains the information on the output of the first hidden layer in a specific range and will be further transformed linearly within subsequent hidden layers using a matrix that employs a technique akin to "folding", inspired by the work of De Ponte et al. [211] and Pace et al. [212].

Based on our thorough analysis, we are able to calculate the analytical formula for determining the appropriate size and transformation matrices required

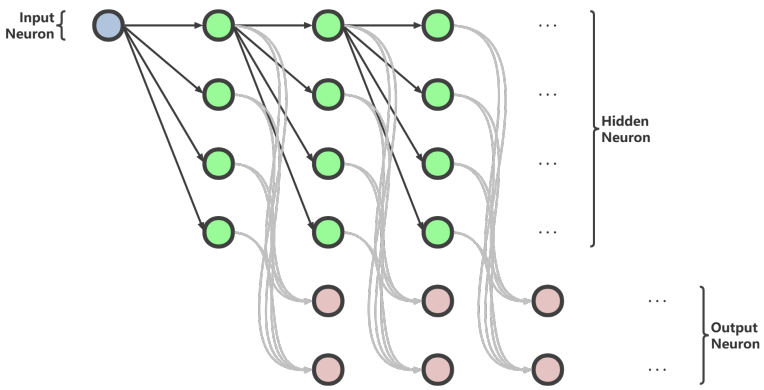


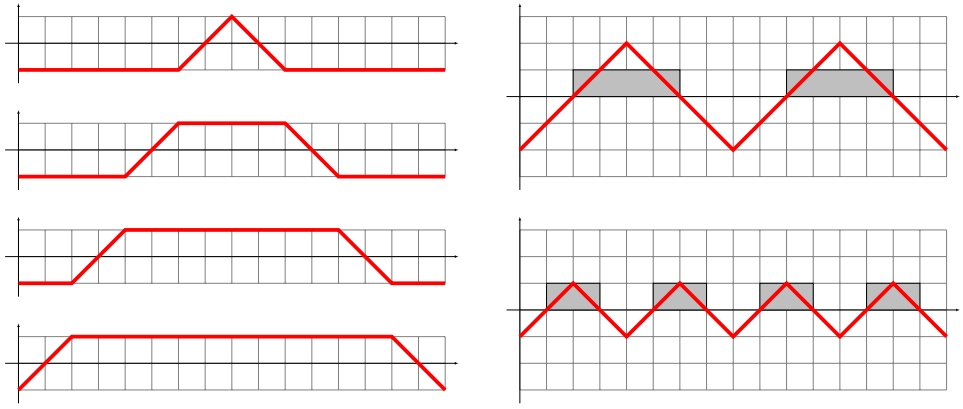
Figure 4.6: The configuration of the recursive neural network for Gray Code representation representation is depicted at a theoretically arbitrary bit resolution. Each layer has the capability to generate an additional two bits of output. The illustrated figure showcases the output neurons (coloured red) from the second to the seventh bit, while the input neuron is depicted in blue. The green nodes represent the recursive hidden layers. Bias nodes have not been specifically identified. The adjacency between consecutive hidden layers is depicted by full connectivity through black straight lines (only connections from the uppermost neuron are displayed for simplicity). The activation functions used in hidden neurons is a hard hyperbolic tangent function ($\tanh_{\mathrm{H}}$). The outputs are interconnected through their respective hidden layers by silver curved lines, using a *Binary Step* function as their activation functions.

for each layer to meet various specifications regarding the resolution of bits to be produced, whether through a recursive or non-recursive approach. The matrices necessary for facilitating communication between the hidden layers in the illustrated architecture depicted in Fig. 4.6 are as follows:

$$weight = \begin{bmatrix} -4 & 4 & -4 & 4 \\ -4 & 4 & -4 & 4 \\ -4 & 4 & -4 & 4 \\ -4 & 4 & -4 & 4 \end{bmatrix}, bias = \begin{bmatrix} -7 & -5 & -3 & -1 \end{bmatrix} \tag{4.14}$$

The results obtained from the matrices are illustrated in Fig. 4.7, with an activation function of the **CMOS** transfer function and subsequent adjustments based on the deviation from the $\tanh_{\mathrm{H}}$ function. It is evident that effective utilisation of the matrix for enhanced resolution hinges on the strength of transfer function's identity. In cases where the transfer function is not sufficiently similar to the PWL function, or the linearity in the non-saturation region is not strong enough, additional processing is necessary to align inputs with the linear range of the activation function.

The performance of the network has been evaluated using an 8-bit resolution, demonstrating a minimal bit error rate of less than 1% on all outputs. This design distinguishes itself from prior research by minimizing the need for extensive amplification in comparison to softer activation functions when utilised in recursive operations [211]. As a result, the system exhibits enhanced robustness to potential parameter variations and relative errors.



(a) (b)

Figure 4.7: The system depicts the principle of a piece of signal processing in Figure 4.6 serves as a solution for the Gray Code analog-digital converter classifier task. The virtual waveform stands for the input dataset arranged by the magnitude of corresponding input seen at the input neuron. The grids are set to illustrate the relative magnitude of the signal at each port of the corresponding layer (one hidden layer and the output layer it is fully connected to). (a) The outputs of the hidden layer neurons are generated through a well-designed utilisation of input signals in the form of a period of a triangular waveform consisting of four segments produced by the previous layer. (b) The pre-activated input generated by a linear transformation of the outputs of the corresponding hidden neurons with biases for balancing (red lines). The frequency of the virtual waveform is doubled once or twice seen by the two output neurons. To produce a Gray Code presentation of the input, Logic **1** is outputted when pre-activation values are greater than 0, represented by shaded gray boxes, and **0** otherwise.

With the utilisation of selectively retaining information without compromise, we can implement this approach in ResNets by formulating matrices to convert the necessary data generated at one layer into a linear region and preserving it for subsequent use. In this aspect, the circuit and activation function plays a crucial role in isolating and ensuring the layer-wise robustness of hardware implementations of such designs.

4.4 Circuit Performance

In the context of hardware design for the AFC, one must carefully consider its potential functionality as an analog LUT and its ability to generate the expected output in an analog manner. Additionally, a crucial aspect to address is how the design may impact a well-established system that relies on digital computation. The key factors to prioritise, aside from simplicity and minimal space utilisation, include energy efficiency in both active and dynamic states, as well as the speed at which it can consistently produce an output that aligns with the input.

In order to successfully implement large-scale neural networks, it is crucial to take into account the overall scalability with respect to energy consumption and fan-out. This refers primarily to the increasing size of each layer within the

model and the cumulative time consumed throughout its depth.

When constructing a system with ultra-high dimension input and output to handle a significant number of computations, it is important to consider the possibility of building the system using a fully parallel approach without the need for external control or data read and write operations. In doing so, we can calculate the amount of power required for each operation individually and then estimate the peak power consumption needed for the system to function normally in practical cases. Additionally, we should also estimate the operating time needed based on the duration of a single non-polynomial operation.

In accordance with the specified topology of the neural network to be emulated, the number of load devices driven by the device is largely contingent upon the final configuration of the system. This, in turn, necessitates the establishment of a robust fan-out capability to effectively manage the flow of data within the circuit. The term fan-out is a concept commonly employed in digital circuit design to delineate the maximum number of inputs that can be provided from a single device without impeding the functionality of the overall circuitry.

In this section, we will discuss the effectiveness of the proposed AFC, with a specific focus on identifying any potential issues related to its ability to drive multiple loads.

4.4.1 Energy consumption and responding time

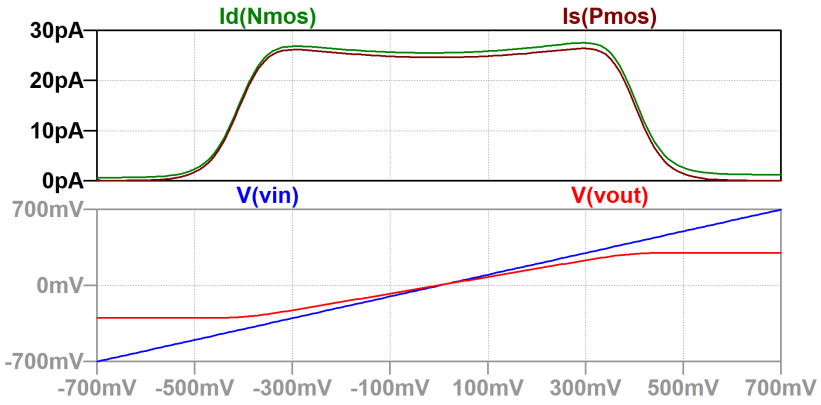
For every operation conducted in the stage of activation function process with the AFC, an estimation will be made regarding the time and energy consumption. This estimation will be based on the duration taken for the input signal to reach the input port of the device, denoted as **Vin** until a stable signal is detected at its output port, denoted as **Vout**. In addition, the analysis will also consider the current flow during this specified timeframe.

With a designated array of voltage supplies allocated for each component, it can be inferred that the total power consumption correlates directly with the current flow. Within this section of the system, there exists the condition of the device functioning in a state where the input remains relatively constant throughout the operation and maintains a consistent magnitude until the subsequent operation is ready to be executed. Therefore, when estimating the total power consumption, it is feasible to overlook the distinction between dynamic and steady stages and instead consider the average power consumption for a specific level of input as the overall consumption, which can be calculated by

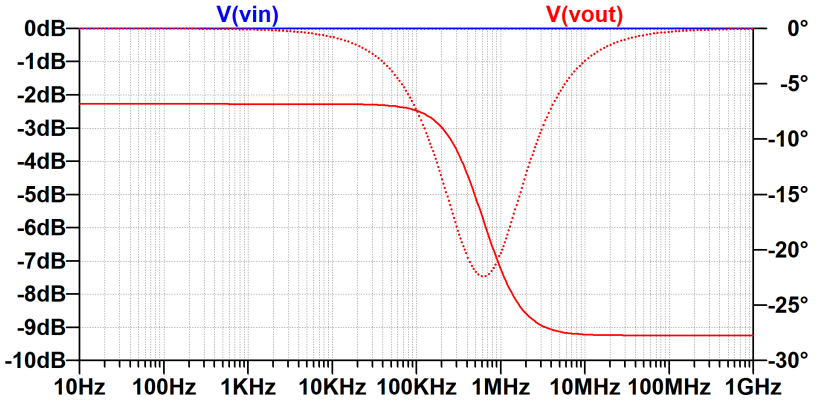
$$P = I_D V_{DS}^n + I_D V_{SD}^p. \tag{4.15}$$

In the given scenario, where I_D represents the current flowing through the system when a specified input voltage is applied to the common **Gate**, the potential differences between the **Source** and **Drain** of the CMOS pair are denoted as V_{DS}^n and V_{SD}^p, respectively. In this portion of the conversation, we will be setting aside the consideration of the polarity of these parameters according to the symmetry of the system to simplify the discussion.

As previously discussed, when the curves of transfer function reach saturation, there is no current flowing through the circuit as one of the transistors is in a blocked state. If the N-MOS transistor is operating above the threshold,



(a)



(b)

Figure 4.8: The simulation results produced by the predictive technology model using LTspice software. (a) The DC sweep from $-0.7\ V$ to $0.7\ V$ at the input port labelled as **Vin** (blue line) along with the corresponding output response (red line). Additionally, the current flow through the two transistors is illustrated in the upper subplot. The horizontal axis represents the voltage level of the input signal, while the vertical axis illustrates both the current flow through the device and the voltage level of the output signal, respectively. (b) The small signal analysis conducted within a frequency range of $10\ Hz$ to $10\ GHz$. The input signal is displayed as a reference in blue curves while the output response is represented in red. Solid lines indicate the amplification ratio and dotted curves showcase the phase shift. For input frequencies exceeding $100\ kHz$, it is observed that the system experiences a phase shift that increases to approximately $22°$ before returning to $0°$ at $100\ MHz$. Additionally, there is a noted amplitude attenuation of $7\ dB$ for $1\ MHz$ and higher input frequency. This phenomenon arises from current leakage due to parasitic capacitance between the **Gate** and **Source**. The horizontal axis represents the frequency of input signal in small signal analysis.

its channel resistance will be significantly lower than that of the P-MOS transistor in the sub-threshold region. This results in the majority of the voltage drop occurring across the P-MOS transistor, driving the **Source** voltage towards **Vcc** and causing the N-MOS transistor to be in a blocked state, leading to no current flow within the circuit, and vice versa.

With the analysis above, , it can be inferred that the two transistors must

both be functioning within the sub-threshold region concurrently or alternatively, one of the transistors may be experiencing a state of obstruction. This deductive reasoning, as demonstrated in the simulation depicted in Fig. 4.8a , aligns with the findings presented in the simulations showcased in Fig. 4.2.

As is typical for these low-power models, the current passing through the two MOSFETs is significantly lower compared to the discrete matched pair. The maximum current recorded is approximately $25.9\ pA$ at an input voltage of $V_i = 300\ mV$, with a peak power consumption of around $15.56\ pW$ for the pair of MOSFETs at standard room temperature conditions. In scenarios where the circuit operates in a saturated state, where current flow is blocked, a minimal current of approximately $1\ pA$ may be observed, resulting in an overall power consumption of about $1\ pW$.

In the absence of any external techniques artificially driving unusedAFCs into saturation states, it can be argued that for an input range of $\pm 0.7\ V$, the RMS power consumption is approximately $11.35\ pW$ for one single activation stage of neurons in a HNN at room temperature condition. This value can be considered the average power consumption when the input signals in certain neural networks are uniformly distributed within or exceed the specified range.

It is evident that in conventional neural networks, the number of AFCs required for implementation is directly related to the quantity of neurons utilised in the model. This ensures that the system can be easily scaled with a focus on energy efficiency, particularly when considering the architecture solely as the physical LUT or generator of activation functions.

In Fig. 4.8b, it is anticipated that there will be a nearly consistent amplification of the input signal within the frequency range of approximately $10\ Hz$ to $100\ kHz$. The overall decrease in amplification is expected to be less than $5\ dB$, resulting in a decrease of approximately 0.3 time of the original input signal. This indicates that the system is quite resilient and reliable for use as an activation function generator for signals with lower frequencies in the specified range of examination.

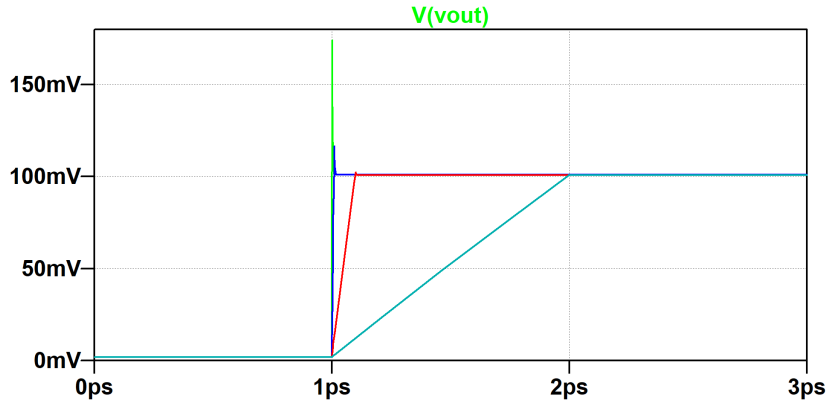


Figure 4.9: The reaction of a zero-load activation function circuit to a step input. The horizontal axis represents the duration over which the data has been collected. The raising time is selected to be $1\ fs$, $10\ fs$, $100\ fs$ and $1\ ps$ respectively. It has been observed that the peak overshoot is significantly increased by approximately 80% when the $1\ fs$ raising time is utilised compared to the steady-state output.

In the simulations, when a step function input ranging from zero to 0 to $0.3\ V$ is applied to the signal with its rising time within $0.1\ ps$, no overshoot or wave distortion is observed, as depicted in Fig. 4.9. This ensures that the performance of the AFC remains effective as a time-efficient LUT in real-world scenarios. Specifically, for both slightly loaded and unloaded AFCs, we can anticipate a relatively immediate response to the input signals provided.

The response time is directly related to the capacitance of the load connected to the device. Furthermore, it is important to consider that, as the magnitude of the net load increases, a higher level of charge flow is required. This places a constraint on the physical dimensions of each layer within the neural network where the system is to be suitable to be deployed without further adjustments involved.

4.4.2 Fan-out

In the implementation of HNNs, the activated signal generated by a single neuron will be sent to multiple separate devices for additional linear operations before being passed on to the next layer. The specifications of the AFC and the emphasis on energy efficiency have prompted concerns regarding the capacity of a circuit to drive multiple devices.

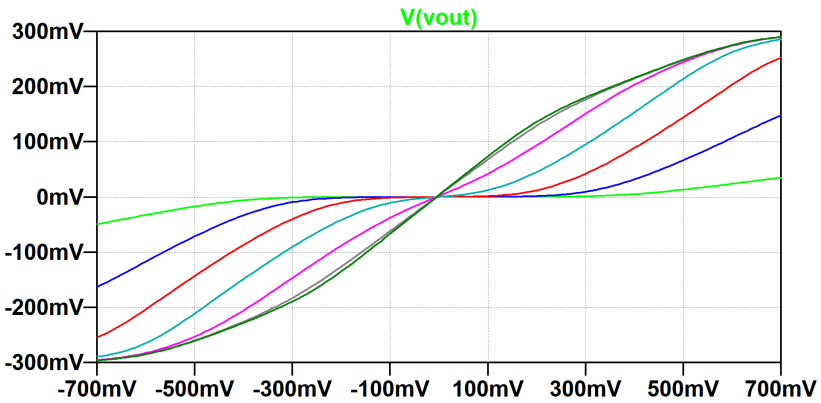


Figure 4.10: The transfer function of the proposed activation function circuit with a resistive load. The colour gradient from green, blue, and red to dark green corresponds to load resistance values ranging from $1\ k\Omega$ to $1\ G\Omega$. Each line on the graph represents a decade of incremental growth in load resistance. The horizontal axis represents the voltage level of the input signal, while the vertical axis illustrates the voltage level of the output signal.

If the potential load of the AFC being proposed is resistive, as is commonly discussed in the literature, it is crucial to consider the possibility that the resistance of the CMOS pair in weak inversion mode may not have a significantly lower output impedance than that of the load resistor. This could result in current leakage through the load to the ground if there are no additional isolation stages in place, ultimately causing distortion to the transfer function to the neutral point, as exemplified in Fig. 4.10. Although this characteristic may be advantageous in certain neural network where adjustment of the activation functions is preferred, as suggested by Liu et al. [213], the character may not be suitable for other de-

signs. Therefore, it is imperative to develop a weighting system that effectively eliminates any potential leakage paths.

In our proposed approach involving capacitive MAC , each junction of linear operation can be represented by a capacitor with a fixed value. This design is expected to reduce power consumption in active mode and significantly diminish dynamic power consumption associated with potential current paths. Nevertheless, the system's functionality can be compromised, particularly when executing a substantial neural network model and demanding a heightened operational frequency.

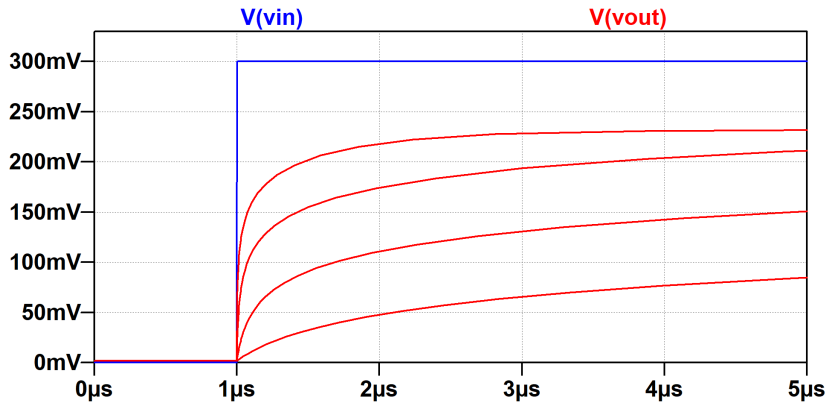


Figure 4.11: The response to a step function with transition from 0 to 0.3 V as the input of the activation function circuit when capacitive loads of different magnitude is applied at the output port **Vout**. The horizontal axis represents the duration over which the data has been collected. For the case a fan-out ratio of 1 is applied, the system is capable to reach the saturation region within 2 μs, corresponding to an over 500 kHz operation condition. The red curves stand for conditions where the equivalent load is set to demonstrate conditions that 1, 10, 100 and 1000 sets of loads are applied, each has a magnitude of 0.67 fF, while the blue curves are the signal observed at the input for reference.

As depicted in Fig. 4.11, the time-domain response of an AFC with a capacitive load to a step input signal can be analysed. The equivalent capacitance of the various loads, based on the topology of a given neural network, plays a crucial role in determining the response. In a commonly used neural network, the layer size can vary significantly from a single neuron to a few thousand neurons. Consequently, the implementation of numerous junctions at the output port, labelled **Vout**, can lead to an unstable condition, as evidenced by the data presented in the figures.

Based on the observed response to raising edge of voltage output **Vout** with capacitive loads introduced, it can be asserted that in order to achieve a response time of less than 2 μs to fit an operating speed of 500 kHz, a fan-out ratio can merely be set to approximately 1. When higher fan-out ratios are used, it is evident that the response time will increase in a linear fashion against the charging time. Furthermore, from the data presented in Fig. 4.12, it is apparent that the signal strength decreases linearly as the number of outputs or net capacitance of the load increases when exposed to higher frequency signals. In conclusion, in order for the system to operate reliably at a frequency of 500 kHz or higher, additional sub-circuits are required to prevent high-frequency signal

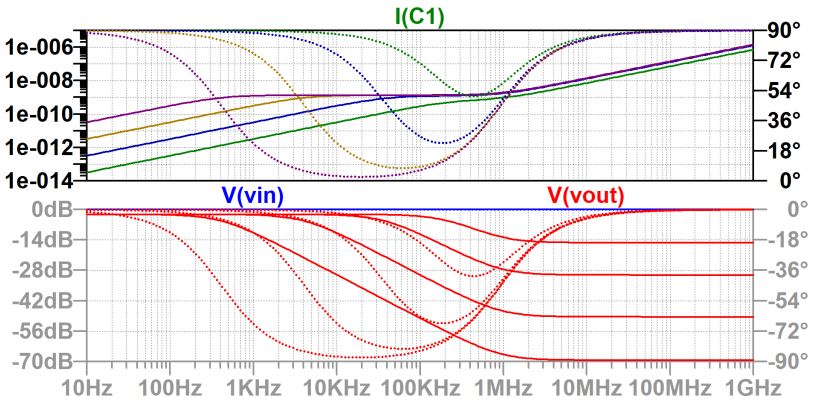


Figure 4.12: The small signal response of the activation function circuit when capacitive loads of different magnitude is applied at the output port **Vout**. The horizontal axis represents the frequency of input signal in small signal analysis. With the purely capacitive load applied on the **Vout** port, the high-frequency conditions are not able to normally perform as a clamping voltage follower and shorted to a nearly **Ground** level. For cases the fan-out requirement is smaller than 10 elements, we may have a nearly linear output for input signals with 10 kHz or slower. The upper sub-plot illustrates the current flow through the load capacitor corresponding to different sizes increasing linearly from the case fan-out is 1 (green curve) to 1000 (purple curve), the phase corresponds to the voltage phase plot respectively. In the lower subplot, the red curves stand for conditions where the equivalent load is set to demonstrate conditions that 1, 10, 100 and 1000 sets of loads are applied, each has a magnitude of 0.67 fF, while the blue curves are the signal observed at the input for reference.

interference.

It can be confidently asserted that in instances where low-frequency applications are the sole consideration and current leakage through the capacitor-based MAC can be deemed negligible for the AFC, a substantial fan-out ratio can be attained. However, in cases where additional isolation is necessary due to higher current levels, the use of resistive components in the subsequent MACs of the circuit will be imperative for achieving a heightened responding speed while maintaining fan-out ratio for the circuit.

As depicted in a modified rendition in Fig. 4.13 of the initial AFC, incorporating a supplementary cascade of several (specifically 10 in the illustrated instance) fan-out circuits mirroring the design of the aforementioned AFC, it is possible to enhance the overall fan-out ratio in a proportional manner without notable time delay.

From the analysis based on Fig. 4.14 , it is evident that the initial output consistently maintains a uniform response time in comparison to scenarios where loads are directly linked. This behaviour closely resembles the reaction time observed when a single load is introduced. The overall response time of the system at the output end, starting from the moment the incoming signal at the shared fan-out bus **FO** (referenced in Fig. 4.13) stabilizes, to when the output achieves a stable state, mirrors the response time illustrated in Fig. 4.11. This indicates that the linear follower structure of the AFC serves as an effective isolator. From the viewpoint of the original AFC, it can be assumed that it is connected to an

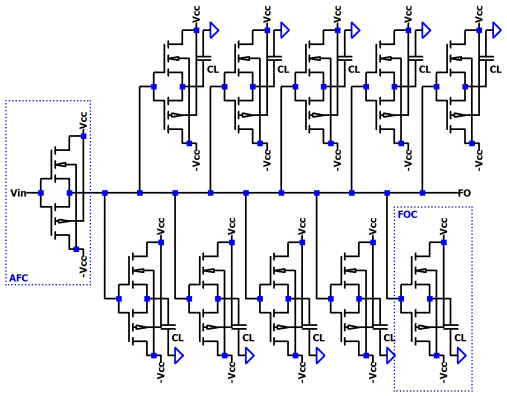


Figure 4.13: The diagram of the layout of the activation function circuit with an additional cascade to enhance the fan-out performance. By connecting to multiple fan-out circuits (designated by a box labelled **FOC**) through the shared port **FO**, the AFC (highlighted by a box labelled **AFC**) can accommodate a growing number of outputs (represented by load capacitors labelled **CL**). Each set of fan-out circuits may exhibit a similar fan-out characteristic as the original AFC without causing significant distortion.

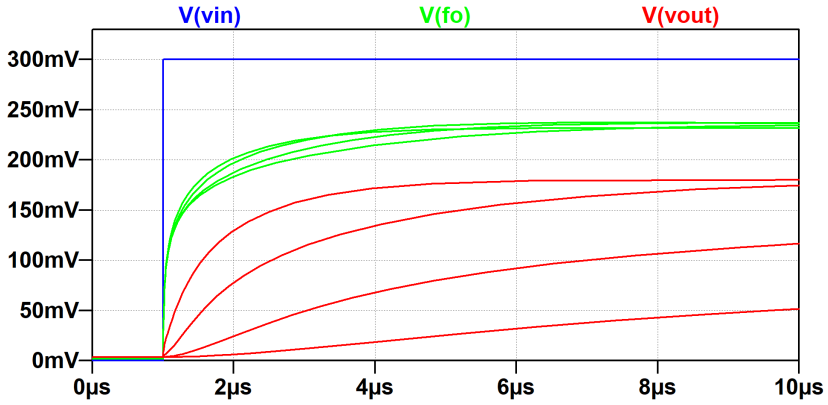


Figure 4.14: The response of the step function when applied to the activation function circuit (AFC) with fan-out circuits as shown in Fig. 4.13. The signal produced at the output port of the original AFC (highlighted in red as **TO**) is distributed to ten AFCs of the same type, each connected to an output load equivalent to a range of junctions' capacitance from 1 (the leftmost curve) to 1000 (the rightmost curve). The output observed at the load (highlighted in green as **Vout**) demonstrates a similar response time as depicted in Fig. 4.11, with a proportional decrease back to the initial value. The horizontal axis represents the duration over which the data has been collected.

output comprising ten devices, each with a constant capacitance value. If we designate the equivalent capacitance of each AFC-based isolating circuit as C_α, and the load as C_L, the total capacitance C_Σ perceived by the original component can be expressed as:

$$C_\Sigma = \frac{C_\alpha C_L}{C_\alpha + C_L}. \tag{4.16}$$

In the event aforementioned that when $C_\alpha \ll C_L$, the equivalent capacitance of

each isolating circuit C_α will predominate and ensure optimal performance of the initial stage of the system, as C_α exhibits a considerably smaller magnitude in comparison to the load.

Furthermore, in order to enhance the robust performance of large-scale implementations, it may be prudent to explore the implementation of amplifier-based systems on the output side of the AFC. Connecting resistors of a specified level can ensure stability in high-frequency environments by providing a consistent current supply and frequency-independent weighting systems. It should be noted that this adaptation may result in increased dynamic power consumption, but it is tailored to meet the specific requirements of certain task specifications.

4.5 Robustness Analysis

In the realm of this current article, it is stated that a neural network demonstrates robustness in relation to variations in a set of parameters if minor alterations to those parameters do not lead to substantial variations in the network's output. When utilised in software applications, the robustness of a neural network is commonly evaluated in terms of how it handles changes in input, model parameters such as weights and biases, and activation functions employed. Nonetheless, there has been a notable focus has been on researching input distortion, with little attention paid to other forms of distortion stemming from configuration issues raised from the precision in representing the functions involved.

In the upcoming section, we will be focusing on the analysis of the application of AFC. Specifically, our discussion will centre around the system's capability to consistently generate a reliable analog LUT of the function, irrespective of external factors such as the well-discussed and case-sensitive factor incoming signals. In addition, we will delve into the intricacies of electron interactions, which can lead to various types of disturbance and impact the efficiency of the system. Furthermore, we plan to assess the overall inaccuracies stemming from these aforementioned factors, as well as the variance present in the transistors utilised in the system.

As per the protocol, we will conduct simulations to illustrate the impact of noise injection on the system throughout the training and validation phases individually. This will allow us to anticipate the circuit's ability to withstand potential distortions from its expected ideal outputs in practical scenarios.

4.5.1 Thermal and noise tolerance

At this level of approximation for the derivation of Eq. (4.5) in Section 4.2, it is important to highlight that the voltage output remains nearly unaffected by temperature variations when using a perfectly matched complimentary pair, as previously discussed.

It is important to highlight that there is a minor leakage to the substrate for each transistor, measuring approximately $1.4\ pA$. This issue could potentially be addressed by utilising SOI substrates or optimising the manufacturing process.

In Fig. 4.15, a comparison of the temperature sensitivity between the PTM and analytical models is presented for the temperature range of 250 to $350\ K$ (approximately -23 to $77\ ^\circ\mathrm{C}$). The smooth transition of the transfer function of

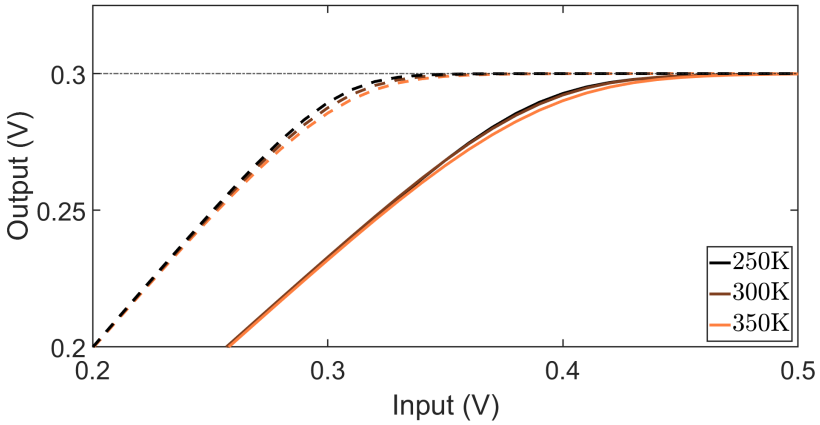


Figure 4.15: Thermal sensitivity of the activation function circuit analysed based on the $45\ nm$ predictive technology model simulations (solid lines) and analytical model (dashed lines) for the transition region between the linear and upper saturated region at temperatures ranging from 250 to $350\ K$. Only minor thermal dependence can be noticed at the transition region in both mathematical prediction and simulation outcome. The labels in the figures indicate the temperatures corresponding to the curves.

the circuit between the linear and saturated regions is clearly depicted in the figure. Due to the Boltzmann terms in the transfer function, the transition between these regions is more distinct at lower temperatures and smoother at higher temperatures.

As predicted by the analytical model, the significant temperature variation occurs only in the regions where the transition to saturation takes place, as shown in Fig. 4.15. On the other hand, the regions with a non-zero slope and the saturated limits themselves are essentially temperature-insensitive for practical purposes.

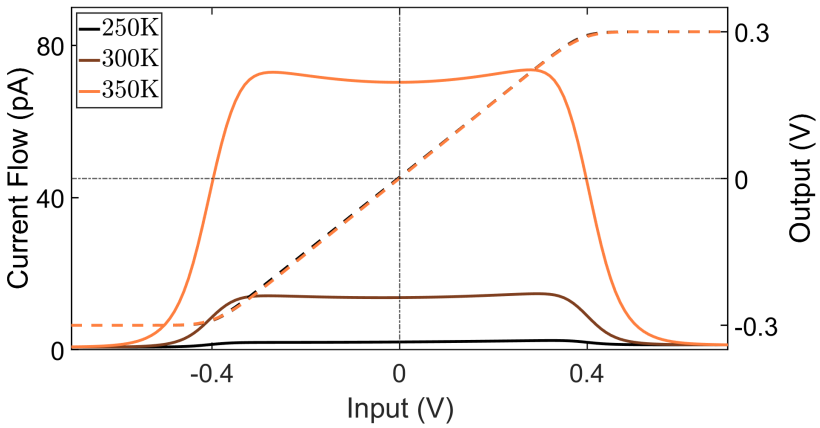


Figure 4.16: Evaluation of the current flow through the pair of transistors in the AFC (solid lines on the left vertical axis) and the voltage transfer function obtained using the $45\ nm$ PTM simulations (dashed lines on the right vertical axis) against varying temperature. The labels in the figures indicate the temperatures corresponding to the curves.

Fig. 4.16 illustrates that while variations in the transfer function with re-

spect to temperature within the defined domain may be subtle, the fluctuations in current within the linear region are substantially pronounced. Consequently, utilising the voltage-current transfer function of this circuit may result in significant calculation inaccuracies. Hence, as articulated in Chapter 3, we will adopt the pure voltage signal transmission methodology for our analysis.

The integrated AFCs can be interpreted as generating a transfer function that lies between a $\tanh_{\mathrm{H}}$ and $\tanh$ with a linear ramp. Based on the analysis, it can be deduced that the circuit is capable of approximating various functions similar to those produced by the PWL function discussed in the work. This offers a range of suitable activation functions that can be selected for specific applications within the hardware implementation.

In the event that the transistors are not appropriately matched, it is imperative to retain the pre-factors β and consequently, the transfer function within the linear region shown in (5) will be affected and becomes:

$$V_o = V_i + \frac{V_T}{2} \ln\left(\frac{\beta_n}{\beta_p}\right). \tag{4.17}$$

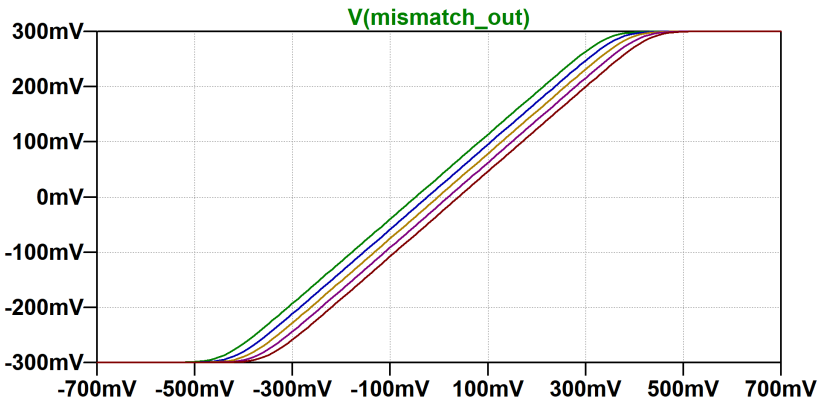


Figure 4.17: The results of a simulation involving a set of mismatched transistors with varying ratio $\frac{\beta_n}{\beta_p}$, ranging from 0.1 (rightmost) to 10 (leftmost) , displayed in a logarithmic sweep pattern. A linear trend is noticeable when plotted against the logarithm of the ratio $\ln \frac{\beta_n}{\beta_p}$. The horizontal axis represents the voltage level of the input signal, while the vertical axis illustrates the voltage level of the output signal.

Up to this point, it has been assumed that N-MOS and P-MOS transistors have threshold voltages that are ideally matched. Moving forward, we will now consider the scenario where the transistors are matched in all aspects except for a mismatch in their threshold voltages. This threshold mismatch will result in a small deviation in the output voltage, either positive or negative, compared to the ideal scenario. This deviation will be approximately equal to the difference between the two threshold voltages.

Consider the situation where the threshold voltages have a difference, which can be either positive or negative, denoted by $V_{th}^n = -V_{th}^p - \Delta V$. If we assume that the pre-factors β are equal and set ΔV to be non-zero, the impact of this

difference will be as follows:

$$V_o = V_i + \frac{\Delta V}{2}. \tag{4.18}$$

The discrepancy at this particular stage of the model assessment is anticipated to be unaffected by temperature for output values of the activation circuit that are not nearing or have not reached the saturation thresholds.

In order to introduce perturbations in the training and validation process of the network, Gaussian noise with varying standard deviations σ will be applied to both the input and the pertinent model parameters of the network. To further elaborate, define $\varepsilon(\sigma)$ as a value generated randomly from a Gaussian distribution with a mean of 0 and a standard deviation of σ. When considering a specific parameter, if we denote the ideal and physical (or perturbed) values as p and $\hat{p}$ respectively, the relationship between the physical and ideal values is defined as follows:

$$\hat{p} = (1 + \varepsilon(\sigma))p \tag{4.19}$$

The results of physical (or non-ideal) activation functions may involve multiple factors that require further estimation. The interference that arises during operations can be a mix of various sources, with amplitudes that may be proportional to the original signal, or dependent on factors such as frequency or temperature, but independent of the signal itself. It is also important to acknowledge that deviations in the activation function of AFCs can occur due to issues such as component mismatch, thermal effects on electronic systems, and manufacturing defects.

In such scenarios, these deviations can be viewed as a combination of a relative error in the output compared to the expected value and an additional fixed or voltage-dependent error linked to the bias signal's power supply. This bias signal is a constant source set at a nominal value of $1\,V$, present at each layer and subjected to amplification by a parameter denoted as b_k in the k^{th} layer. All these parameters are incorporated into a model represented by Eq. (4.19). Consequently, the behaviour of the actual activation function can be realistically depicted as a composite of the aforementioned disturbances, as will be shown:

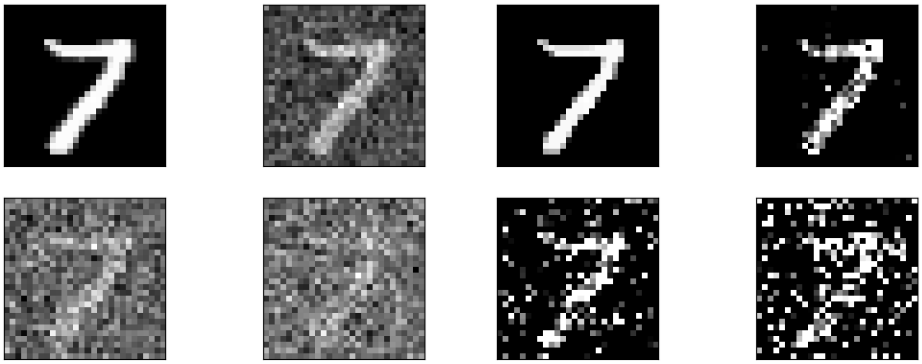
$$\begin{aligned} \hat{\alpha}(\cdot) &= (1 + \varepsilon_\alpha(\sigma_\alpha))\,\alpha(\cdot) + \varepsilon_n(\sigma_n) \\ &= (1 + \varepsilon_\alpha(\sigma_\alpha))\,\alpha(\cdot) + \varepsilon_\beta(\sigma_\beta)\,\beta. \end{aligned} \tag{4.20}$$

In this context, the terms ε_α, ε_n and ε_β refer to the scaling distribution of noise that is applied to the activation function, white noise with constant magnitude and noise modelled by the bias signal of the neural network denoted as β, respectively.

4.5.2 Tolerance in neural network system

The robustness of a neural network is often evaluated by examining how it responds to changes in input data and adjustments in its weights and biases. Furthermore, in this analysis, the network's ability to withstand computation imperfections and generation of activation function due to component tolerances in the hardware implementation is considered. To simulate variations in the hard-

ware elements of the HNNs during the network's training and validation process, random perturbations generated from Gaussian distributions with varying levels of standard deviation σ applied around the optimal parameter values.



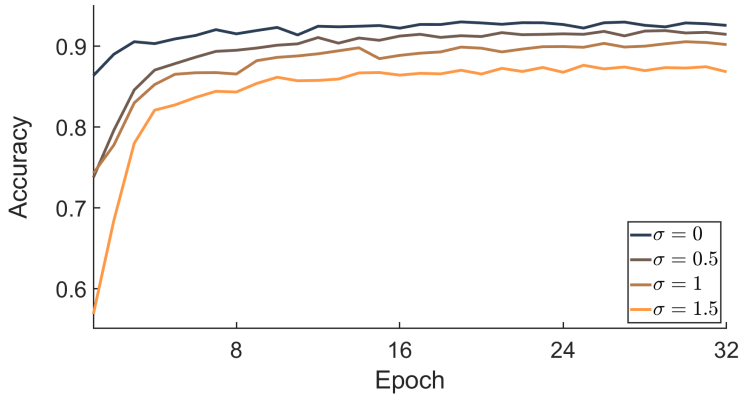
(a) (b)

Figure 4.18: One particular instance of corrupted input data, denoted as number **7** within a dataset known as the Modified National Institute of Science and Technology dataset, is to be entered into the neural network given. The four subsections of each case depict the input that the neural network will observe with a proportional distortion of 0, 0.5, 1 and 1.5 times of the original element. (a) The distorted inputs, without further pre-processing, as will be applied in the experiments. (b) The distorted inputs, after being clamped back to the initial range of input, attached as a reference.

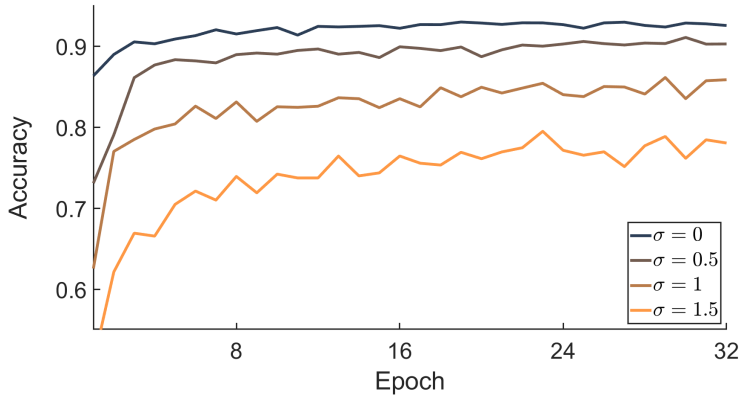
According to former practice, the perturbation on the input of the activation function can be considered as a scaled value with a known relative error, as depicted in Fig. 4.18. This relative perturbation follows a certain magnitude or distribution as outlined in Eq. (4.19). The output of the physical generated activation function can then be accurately modelled using the model defined in Eq. (4.20).

The physical device known as the hardware generator of the activation function is then integrated into a neural network for classification purposes. This neural network is of a topology of $[784 \times 28 \times 14 \times 10]$, *i.e.*, 784 input neurons, featuring two hidden layers with 28 and 14 neurons in the first and second hidden layers, respectively, and ten output neurons. The activation function and utilised after the calculation of the **logits** layer is a *Softmax* function. The neural network underwent training using the well-known MNIST dataset using a cross-entropy (CE) loss function for a total of 32 epochs with a batch size of 256. The results of the training process, including ideal scenarios, are depicted in Fig. 4.4 for reference.

Fig. 4.19 shows the impact of noise introduced to the input of the neural network on the network's accuracy. The perturbation affects only the input signal to be transformed by the input neurons. When the training data is contaminated with noise, a relative error of 50% may result in a mere decrease of 2% in precision when tested against an ideal validation set. Even with a higher error rate, there is not a significant decline in accuracy.



(a)



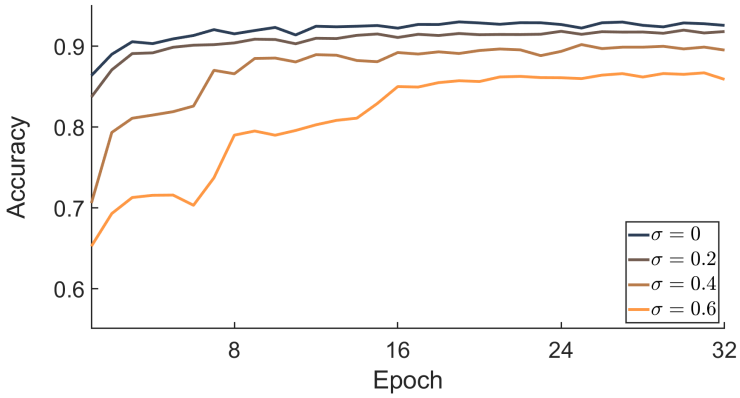
(b)

Figure 4.19: The impact of noise introduced at the inputs of the system during both validation and training procedures which can be represented by the equation $\hat{X} = X\left(1 + \varepsilon_X\left(\sigma\right)\right)$ (a) Noise added to the inputs during training period. (b) Noise added to the inputs during validation period. The dark blue, grey, orange, and yellow curves illustrate the varying levels of relative perturbation $\varepsilon_X\left(\sigma\right)$, depicted with standard deviations σ ranging from 0, 0.5, 1 to 1.5.

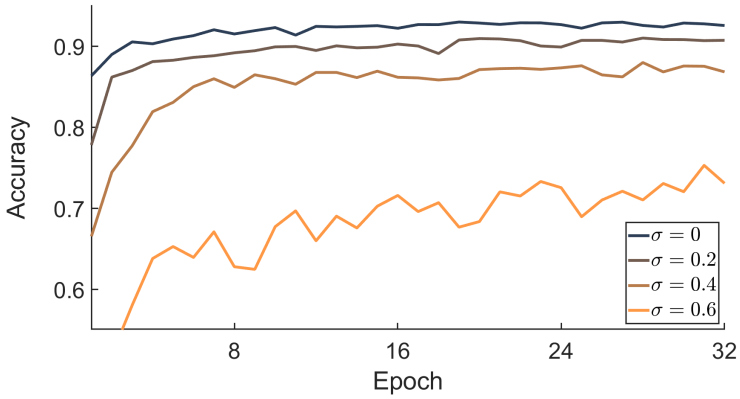
On the other hand, training the network with clean data can still yield a 90% accuracy rate even with data that has been slightly perturbed by 50%. However, with higher levels of distortion, the inaccuracies in the model increase significantly.

The impact of noise added to the output of activation functions in each layer on the accuracy of the network is depicted in Fig. 4.20. When a minor (20%) distortion is applied during both the training and validation phases, there is little observable decrease in accuracy. However, for all scenarios where perturbations are introduced to activation functions during validation, there is a consistent linear drop in overall accuracy relative to the intensity of the perturbations. It is worth noting that when a significant perturbation is implemented during the training phase, the system may struggle to reach a stable local minimum as effectively compared to other scenarios.

It is also possible to represent the distortions at the activation functions as a displacement along the curve and an additional random variation. This can be



(a)



(b)

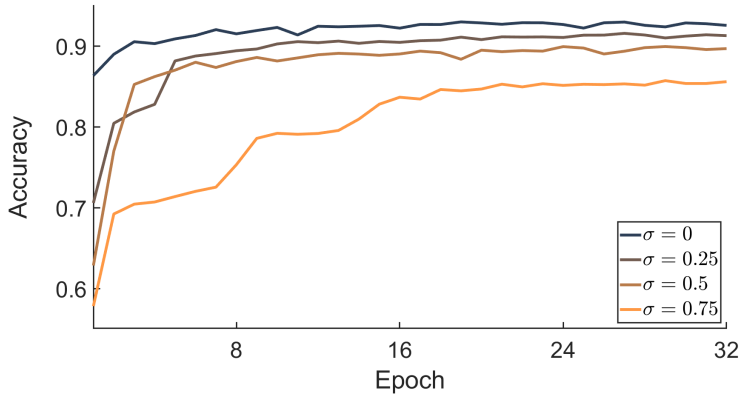
Figure 4.20: The impact of noise introduced to the outputs of each activation functions during both validation and training procedures which can be represented by the equation $\hat{\alpha}(\cdot) = (1 + \varepsilon_\alpha(\sigma))\,\alpha(\cdot) + \varepsilon_n(\sigma)$, where ε_α and ε_n shares a same standard deviation of σ. (a) Noise added to the activated signals during training period. (b) Noise added to the activated signals during validation period. The dark blue, grey, orange, and yellow curves illustrate the varying levels of relative perturbation $\varepsilon_\alpha(\sigma)$, depicted with standard deviations σ ranging from 0, 0.2, 0.4 to 0.6.

expressed as:

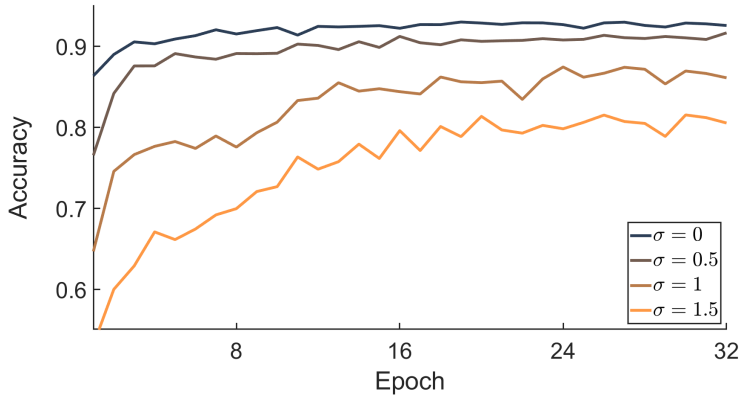
$$\hat{\alpha}(Y) = \alpha(Y(1 + \varepsilon_I(\sigma_I))) \times (1 + \varepsilon_O(\sigma_O)). \tag{4.21}$$

In this case, Y is the nominal pre-activation value of the activation function, ε_I and ε_O represent the perturbations on the pre-activation and activated sides of the activation function. This approach combines linear and non-linear operations, making it challenging to differentiate or analyse the impact of each individual sub-module. The output of Monte-Carlo method simulation with this mean of model is presented in Fig. 4.21

In conclusion, the HNN developed in this study demonstrates a satisfactory level of robustness and accuracy for classification tasks compared to the ideal-case software configurations according to simulations with noisy signal channels. When perturbations are applied to the neural networks when they do not exceed



(a)



(b)

Figure 4.21: The impact of noise introduced to the pre-activation and activated value of each activation functions during both validation and training procedures which can be represented by the equation $\hat{\alpha}(Y) = \alpha(Y + \varepsilon_I(\sigma)) + \varepsilon_O(\sigma)$, where ε_I and ε_O shares a same standard deviation of σ. (a) Perturbation introduced to the pre-activation and activated signals during training period. (b) Perturbation introduced to the pre-activation and activated signals during validation period. The dark blue, grey, orange, and yellow curves illustrate the varying levels of relative perturbation $\varepsilon_I(\sigma)$ and $\varepsilon_O(\sigma)$, depicted with standard deviations σ ranging from 0, 0.25, 0.5 to 0.75.

50% of the input or 25% of the activated output of the activation function, during either training or validation, the system can be trained with approximately 90% accuracy when validated on a separate dataset, which is only a slight decrease of approximately 2% compared to a system without perturbations.

In practical applications, as illustrated in Fig. 4.19b and Fig. 4.20b, we can confidently utilise a model trained under ideal conditions while implemented with non-ideal elements to achieve satisfying accuracy in this specified task.

In accordance with the findings presented in Eq. (3.8) in Section 3.3, it is observed that activation functions with a gradient not exceeding 1 throughout its domain have the potential to mitigate distortion or noise through their inherent properties. This phenomenon allows for deeper and more streamlined neural networks architectures to effectively maintain internal resilience by rectifying noise during the activation process.

4.6 Conclusion

Developed within the framework of hardware neural networks, a simple schema has been introduced for low-power implementation of activation functions. The design is fully CMOS compatible and can be readily manufactured with commercial technologies available.

The AFC demonstrates favourable characteristics with regard to the stability of generating the proposed transfer function, even when subjected to changes in temperature and parameters. As a result, this simplifies the requirements for the overall neuron design. This assertion has been demonstrated through the development and training of a multi-layer perceptron (MLP) for image classification task defined with the MNIST dataset. The activation function utilised in the sample networks can be seen as a variation of the hard hyperbolic tangent ($\mathrm{tanh_H}$) outlined in Tab. 4.1, incorporating a softer transition feature that enhances the efficiency of gradient-based optimisation techniques.

Compared with digital-based implementations of neural networks, HNNs are subject to less precision in their construction and less accuracy in their operation. Yet, under practical assumptions, the accuracy and robustness of the example hardware network are comparable to their software counterparts, with respect to noise added during both the training and validation phases. In particular, the robustness of the network is acceptable regarding the perturbations of both the input and also model parameters, within the ranges encountered in practice.

Moreover, the activation function with similarities to the $\mathrm{tanh_H}$ function demonstrates improved performance when compared to the widely used *ReLU*. It achieves higher accuracy in a shorter number of training epochs. The low power consumption, the resilience to input noise and component variability, and the ease of integration of the hardware neuron suggest a promising direction for compact HNNs for embedded and distributed applications. Compared with the function tanh suitable for embedded and distributed systems. Additionally, the derived function exhibits advantages over the hyperbolic tangent function in tasks that necessitate recursive behaviour and linear transformations.

Chapter 5

Multiply Accumulate Circuit

In order to develop a multiply accumulate circuit (MAC) with minimal current demand suitable for being driven by the activation function circuit (AFC), we have devised an innovative structure using capacitive components and pass-gates based on complementary metal oxide semiconductor (CMOS) technology. This novel MAC has been designed to accommodate signed weighting factors, while conforming to a defined range of specifications in terms of precision and range, and utilising technology that meets industry standards.

Furthermore, we conducted an analysis of the performance of the structure both as a standalone module and as a functional block within a specified neural network , taking into account both ideal and non-ideal components. Based on our calculations and simulations, we have demonstrated the ability of this design to achieve comparable levels of time and energy efficiency as the current state-of-the-art (SOTA) hardware neural network (HNN) design. Additionally, this structure maintains a high level of simplicity both in device design and scalability.

An examination of scalability with regard to equivalent bit resolution within the context of imprecise components has been conducted. It has been observed that there is a systematic error with respect to the desired output, which can be effectively controlled by employing appropriate elements as will be specified, and proper training algorithms.

5.1 Overview

In the realm of neural networks and associated disciplines, the commonly used linear operation $y = Ax + b$ plays a significant role in facilitating communication between layers, presenting pertinent data, and presenting its potential impact on classification and regression tasks. In the context of hardware implementation, the mechanism used to transform multiple inputs into outputs in a linear fashion is known as a multiply accumulate circuit (MAC).

The utilisation of an intricate network of multiple buses for each input and output in order to achieve a linear operation within a hardware-implemented system, often referred to as a crossbar or synaptic network, is widely utilised in the field of circuit design for neural network acceleration research. In this type of design, the horizontal and vertical wires correspond to the output port of the previouos layer delivering the incoming signals and the input port of neurons

in the subsequent layer, respectively. The connections between each horizontal (input) wire and vertical (output) wire typically are constructed with resistive elements, symbolising the weights that are being assigned. A detailed schematic will be demonstrated by Fig. 5.10 in Section 5.4.

5.1.1 Background: Crossbar circuit for linear transformations

The use of crossbar structures in parallel computing as a switching system has demonstrated its ability to efficiently distribute signals among memories and processors [39, 40]. Recent studies have explored the use of programmable resistive devices to modify the switching components, resulting in the successful development of a bio-inspired computing system on a chip in a naturally parallel fashion. This innovative approach enables a time-efficient communication and processing method that can directly interact with analog signals from external devices, such as sensors, without the need for time-consuming conversion stages [214]. Additionally, this method offers enhanced time and space efficiency when compared to traditional digitalized binary computing elements [215].

In the context of neural networks, particularly in deep neural network (DNN), convolutional neural network (CNN), or other time-independent fully connected structures, memristors are frequently selected as programmable resistors [216, 217]. Interestingly, this concept shares a topological similarity with conventional summing amplifier networks. In both cases, incoming signals for a specific neuron are converted into voltage levels and passed through weighting resistive components to a common wire or port, where the voltage reference remains constant with respect to a designated ground. These signals are then summed by the amplifier in the summing or converting circuit [218].

One drawback of the current system lies in the fact that the resistivity of a linear resistor is constrained to be positive, which impedes the ability to display input weights with identical signs for a specific neuron. One possible resolution could involve the incorporation of replicated input wires linked to the negative terminal of operational amplifier (Op-Amp) within the summation or conversion phases accompanied by extra junctions [219].

An additional aspect of linear circuit technology involves the utilisation of capacitive junctions, as discussed in the literature [215]. This particular application is typically focused on time-dependent systems, such as spiking neural networks (SNNs) [45], because of their ability to generate charge by current pulses during operation. However, these capacitive junctions can also be used effectively in more static and time-independent systems [220]. In these applications, the technology is known to effectively eliminate direct current paths and static energy consumption.

Various methodologies can be employed to optimise the efficiency of capacitive junctions within these systems. This entails meticulous adjustment of the charging duration, leveraging the first-order linearity in the exponential growth for a small signal, and ensures that charge generation on the capacitor is linearly proportional to time. Furthermore, the use of various capacitances of varying magnitudes can be explored to scale signals by different ratios [43,44,78]. These versatile techniques can be implemented individually or in combination to effectively address specific operational requirements [221]. Moreover, a method has

been devised for managing current sources through the process of signal weighting, leading to the conversion of these weighted signals into stored charges on capacitors [222]. In addition, additional junction designs incorporating quantum dots [223] as well as a hybrid of capacitors and memristors, known as memcapacitors [46, 224] are present but will not be elaborated upon in this research.

The criteria for evaluating such tools primarily centre around the operational frequency measured in floating-point operation per second (FLOPS), or operation per second (OPS), and the power consumption relative to this speed, expressed as operation per second per Watt (OPS/W). In recent years, typical designs have showcased a range of *Mega-* to *Giga-FLOPS* level, along with energy efficiency levels ranging from *Terra-* to *Pita-OPS/W*. This is consistant with the innovative capacitive designs that our research is currently developing.

5.1.2 Motivation: Low-current Multiply Accumulate Circuit

With the implementation of an energy efficient activation function circuit (AFC) that we have developed, it has come to our attention that, without additional amplification, the current output drawn from the circuit is markedly insufficient. This has resulted in a fan-out issue often encountered when incorporating traditional resistive arrays. In our pursuit of an energy-saving MAC design, we have instead opted for a lesser known capacitor-based system to navigate around current limitations at the expense of maximum operation frequency. Furthermore, to achieve an optimal design without the need for complex and time-consuming converters to bridge the gap between analog and digital signals, we have treated all digital signals as analog voltage levels and standardised the interface to facilitate seamless compatibility.

In the course of investigating this particular subject, it has come to our attention that in many MACs with capacitor-based functional components, the capacitance values vary from one another. This variance complicates the manufacturing process, increases the complexity of parameter selection, and requires extensive tuning. A potential interim measure involves quantizing parameters into binary format and implementing a ladder structure composed of just two types of capacitor within the system, with transistors serving solely as pass-gates. This approach allows for scalability in the design, with a minimal number of components involved, resulting in a highly compact and therefore space-efficient system.

Following an extensive review of the literature, it has come to our attention that there exists a conspicuous absence in the discourse pertaining to inaccuracies associated with control elements, current leakage paths, and non-ideal characteristics in capacitive components. These significant themes will be carefully examined and deliberated upon in the pursuit of our research endeavours.

The primary objectives of this chapter will encompass the following areas of focus:

- Development of a multiply accumulate circuit that incorporates a reduced number and variety of components compared to previous circuit designs.
- Evaluation of system performance in both ideal and non-ideal scenarios.
- Estimation of maximum energy and power consumption in worst-case scenarios, juxtaposed with state-of-the-art technology.

- Comparison of various implementation approaches involving software and hardware for the MAC stage, culminating in a detailed analysis of the proposed design.

5.2 Multiply Circuit

The key focal point in the development of the multiplicative segment of the MAC lies in analysing the energy usage per operation in various operational states, including dynamic and static modes. The proposed AFC raises a particular concern as it may not be optimal for driving multiple outputs due to the limited current supply of a sub-threshold mode transistor pair. This limitation renders it susceptible to noise interference and may result in increased processing time when interfacing without being isolated by summing amplifier arrays as commonly employed in the literature. Fig. 4.12 also indicates that the output impedance of activation function circuits (AFCs) may not be sufficiently small. Consequently, this could potentially hinder the MACs from drawing the necessary current to achieve the desired voltage level at the subsequent layer.

As an alternative solution, we suggest implementing a static capacitor-based system instead of a resistive array. The proposed design minimises the charge supply required for linear operations on inputs driven from the output ports of AFCs in preceding neurons or external sources. This system can be scaled efficiently using a recursive approach tailored to the specific task at hand, exhibiting a response speed and power efficiency comparable to current state-of-the-art (SOTA) using industry-compatible technologies. It excels in terms of space, energy, and response time efficiency, operates without the need for external commands, and can be easily integrated into existing memory systems. The details of the design and the analysis on its performance will be discussed in the following sections.

5.2.1 A scaleable quantized capacitive weighting system

The multiplying part of the MAC has been designed with a streamlined methodology influenced by principles of recursion. It features a network of interconnected capacitors that allow for seamless integration of input from either a predefined analogue voltage source derived from an activation circuit or the ground, made possible by the use of multiplexers (MUXs).

In an optimal scenario, as depicted in Fig. 5.1, parts surrounded by boxes that share a same line style contributes an equivalent capacitance as a whole across the three circuits. Consider the structures on the left and centre of the illustration as an example, upon analysis from the designated node labelled **o1**, assign **C** as the standard value of the capacitor as visually indicated in the diagram. Thus, the capacitance of the dotted part in the middle, seen from node **o2** can be computed as:

$$\begin{aligned} \mathbf{C}_{\Sigma}^{-1} &= 2\mathbf{C}^{-1} + (\mathbf{C}+\mathbf{C})^{-1} \\ &= \mathbf{C}^{-1}. \end{aligned} \tag{5.1}$$

Therefore, it can be postulated in theory that the two aforementioned seg-

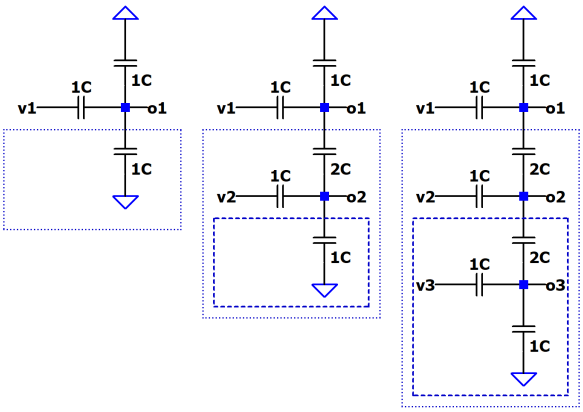


Figure 5.1: The diagram illustrating the fundamental principle of a capacitive weighting system. The selection of component parameters is made in a subjective manner for the purpose of illustrating their interrelationship. The signal input **vX** and its corresponding connection represent a selected input source, whether from a signal source or grounded, facilitated by an analog multiplexer. The dotted and dashed box outlines a systematic approach for integrating further input signals.

ments bear equivalence to the node being examined. Moreover, the expanded section demarcated by dashed outlines within the dotted rectangle serves as an illustration of how the system could be extended to enhance resolution.

Utilising the identical approach, we may deduce that for every node denoted as **oX** (where **X** represents the index of the node), the upper and lower components it perceives are equivalent in magnitude. By employing capacitor with a capacitance value of either one or two times of the standard value **C** within the system, and seeing it from the input nodes designated as **vX**, we can derive a correlation between the impact that input **vX** exerts on output **oX** as outlined below:

$$\begin{aligned} \mathbf{oX} &= \frac{(\mathbf{C}+\mathbf{C})^{-1}}{\mathbf{C}^{-1}+(\mathbf{C}+\mathbf{C})^{-1}}\mathbf{vX} \\ &= \frac{1}{3}\mathbf{vX}. \end{aligned} \tag{5.2}$$

In accordance with the superposition principle, when determining the ultimate output voltage observed at any given node within the network, it is advisable to individually assess the impact of each voltage supply and subsequently combine these results to obtain the final output. Referring back to the notations detailed in Fig. 5.1, it is possible to ascertain that the potential **o2** introduced onto **o1** is

$$\begin{aligned} \mathbf{o1}|_{\mathbf{o2}} &= \frac{(\mathbf{C}+\mathbf{C})^{-1}}{2\mathbf{C}^{-1}+(\mathbf{C}+\mathbf{C})^{-1}}\mathbf{o2} \\ &= \frac{1}{2}\mathbf{o2}, \end{aligned} \tag{5.3}$$

Following the methodology employed in the preceding analysis, it can be

inferred that $\mathbf{o2}|_{\mathbf{o3}} = \frac{1}{2}\mathbf{o3}$ and $\mathbf{o1}|_{\mathbf{o3}} = \frac{1}{2^2}\mathbf{o3}$. This principle can be extended with more intricate recursive expansion of the system. Furthermore, the ultimate result observed at node **o1** can be succinctly encapsulated as follows:

$$\begin{aligned}\mathbf{o1} &= \frac{1}{3}\mathbf{v1} + \sum_{i=2}^{n} \frac{1}{2^{i-1}}\mathbf{oi} \\ &= \frac{2}{3}\sum_{i=1}^{n} 2^{-i}\mathbf{vi}.\end{aligned} \tag{5.4}$$

Based on this element, we can proceed to execute the desired multiplication $p \cdot x$ as described in the linear operation of the neural network. This involves representing the input x by a voltage-based signal v_x seen at a specific input bus while representing p by $\frac{2}{3}\sum_{i=1}^{n} 2^{-i}s_i$, where s_i denotes the connectivity of individual input nodes **vX** to the bus v_x. This connectivity is indicated by $s \in \{0, 1\}$ and corresponds to an on / off switch in the circuit design. In this particular configuration, the maximum output that can be achieved from such a module with n-bit resolution will be equal to $p_{max} = \frac{2\left(1-2^{-n}\right)}{3}$.

In consideration of the level of accuracy necessary for a specific task, the design will facilitate potential scalability with binary-based bit-wise resolution. In an optimal scenario with n cascades of the suggested capacitor-MUX group, we can achieve a weight within the span of $\left[-\frac{2^n-1}{2^n}\alpha, \frac{2^n-1}{2^n}\alpha\right]$, with a resolution of $\frac{1}{2^n}\alpha$. It is important to note that in this context the term α equals $\frac{2}{3}$.

In order to achieve a weighting parameter p greater than 1, it is advisable to utilise a collection of m modules known as synapses within the weighting circuit. The quantity of synapses needed can be calculated by $m = \left\lceil \frac{p}{p_{max}} \right\rceil$, which represents the minimum quantity required. Each synapse will be assigned a weight of either $p_{lower} = \frac{\left\lfloor 2^n \frac{p}{m} \right\rfloor}{2^n}$ or $p_{upper} = \frac{\left\lceil 2^n \frac{p}{m} \right\rceil}{2^n}$.

In the examination of artificial neural networks (ANNs), utilising a securely trained and stable parameter configuration, the weight values are maintained in a static state in the memory for the regulation of MUXs. The designation of a parameter applied to each input value can be managed through a setup of a MUX for every bit. Consequently, in this particular framework, it is viable to assert that the system can be constructed utilising conventional static random-access memory (SRAM) without causing any disturbance to the primary functional components.

5.2.2 Pass-gate as the multiplexer

In the previous section, it was proposed that the weighting information could be stored across multiple synapses, with each synapse having a maximum magnitude of $\frac{2}{3}$. The weighting information for each synapse is calculated using the formula shown previously. It can be inferred that the parameter is governed by a series of distinct selections from a nominal set consisting of either 0 or 1. Consequently, it is reasonable to assume that the system operates in a binary manner and presents itself in a manner akin to fixed-point numeric representation. Given the linear relationship of each selection's contribution, we may designate this as a "bit", analogous to terminology utilised in digital systems.

To effectively control each bit of the weight, one common approach is to use transistors as switches. However, in real-world situations, the incoming signals may not always maintain a consistent positive or negative magnitude within the same synapse and task under varying conditions. The design including a pair of parallely connected N-Channel metal oxide semiconductor (N-MOS) and P-Channel metal oxide semiconductor (P-MOS) for pass-gates is a conventional design of many advantages. As a result, it may become essential to employ a pair of CMOS to create a practical pass-gate mechanism.

In alignment with the established design principles and the overarching goal of optimal functionality, it is imperative to address the potential issue of distorted signals due to the accumulated charge on input capacitors in our capacitor-based system. To mitigate this risk, a connection to the ground bus **GND** will be implemented for the purpose of discharging any residual charge. Moreover, in order to ensure proper operation and prevent both pass gates from being of a same state simultaneously and result in undefined behaviours, two additional transistors will be incorporated to create an inverter. This modification will transform the structure into a MUX capable of accommodating a range of incoming voltage levels.

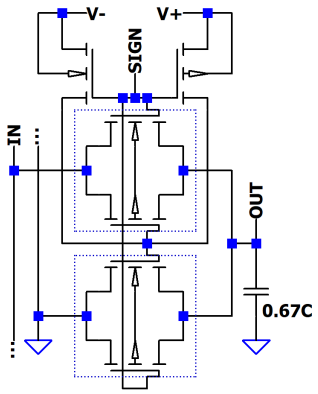


Figure 5.2: The schematic of the multiplexer system. The load capacitor of $0.67\mathbf{C}$ denotes the equivalent capacitor of the synapse seen from the input port. **IN** denotes the real input passed onto the synapse and **OUT** denotes the output port of the MUX, and the input of the proposed weighting array. **SIGN** denotes the sign of the given bit, fed to a pair of inverters to drive the two inverted pass gates (highlighted by dotted boxes).

An advantage of this particular design is that the control of the assigning module is effectively regulated by the **Gate** potential of the inverter and pass gates, as illustrated in Fig. 5.2. This ensures that no current may inadvertently leak through the components in the static stage. Essentially, apart from the necessary current path from the input **IN**, across two pass gates in opposing states, to **GND**, and the path within the inverter, minimal static energy consumption is anticipated when weights are assigned to the input for each bit. This setup allows for the input to be powered by charges stored in a CMOS-based memory or equivalent alternatives with a controlled level of energy consumption.

Furthermore, in applications where parameters remain constant or are only occasionally modified during linear operations as is the case for very low fre-

quency operation, it is likely that the module will rarely transition into a dynamic state. This observation aligns with the prevailing trends in the market regarding digitally designed units, suggesting that it is feasible for large-scale systems to attain an adequate energy budget for each operation with the design.

In certain unique configurations of neural networks where the parameters can be adjusted in real time, our studies, conducted using the circuit illustrated in Fig. 5.2, involved a capacitor with a capacity of $0.67\ pF$ to represent scenarios where the capacitors depicted in Fig. 5.1 have a nominal values of $\mathbf{C} = 1\ pF$. The selected value has been determined to align with commercially available levels for ceramic patch capacitors, thereby facilitating the subsequent circuit design and verification processes. In the domain of semiconductor manufacturing, specifically referring to the $45\ nm$ fabrication process, it is generally anticipated that the capacitance values will fall within the *femto-farad* or *atto-farad* range.

It was observed that the transient peak of current during saturation-region switching of pass gates was limited to a minimal level of half a *milli-ampere* (specifically peak currents of $240\ \mu A$ for N-MOS and $120\ \mu A$ for P-MOS with a switching time in a *micro-second* level) according to simulation performed with a $45\ nm$ technology predictive technology model (PTM) in a simulation program with integrated circuit emphasis (SPICE) software, as shown in Fig. 5.3. The primary contributor to the peak current pulse is the capacitor, which will exhibit a proportional decrease in relation to both the supply voltage and capacitance. Furthermore, by extending the transition duration, it is possible to achieve a reduction in the peak current. Under steady-state conditions, it is anticipated that the device draws less than $1\ nA$ from the source, depending on the selected channel widths of the N-MOS and P-MOS of $140\ nm$ and $450\ nm$, as designated for the AFC. As will be demonstrated in Section 5.4, the consumption will be negligible compared to the consumption in the capacitor array. Moreover, in the event that the system is permanently hard-wired, the energy consumption associated with the pass-gates can be effectively minimised. This dynamic mode, taking into account the charging and discharging impact of the capacitor, can thus be perceived as demonstrating commendable energy efficiency.

The potential ramifications of this design offer the opportunity for enhanced memory processing efficiency by allowing the potential seen at the SRAMs to directly interface with the assigning of parameters. Also, the design decreased silicon utilisation in real-world scenarios. In the aspect of weighting, the addition of each bit resolution only necessitates the use of one pair of inverter, two pass-gates, and two extra capacitors, culminating in a total allocation of six transistors and two capacitors within the framework of the current application and topology. Furthermore, for a hard-wired implementation of this system, it is also feasible to simplify the two pass-gates and construct the circuit using just three transistors.

5.3 Accumulate Circuit

Based on the proposed design of the multiply part of the system, it will be relatively easy to demonstrate a linear operation of a singular input. In order to linearly transform a one-dimensional array of multiple inputs simultaneously, as is typically seen in neural network models, a summation operation will be

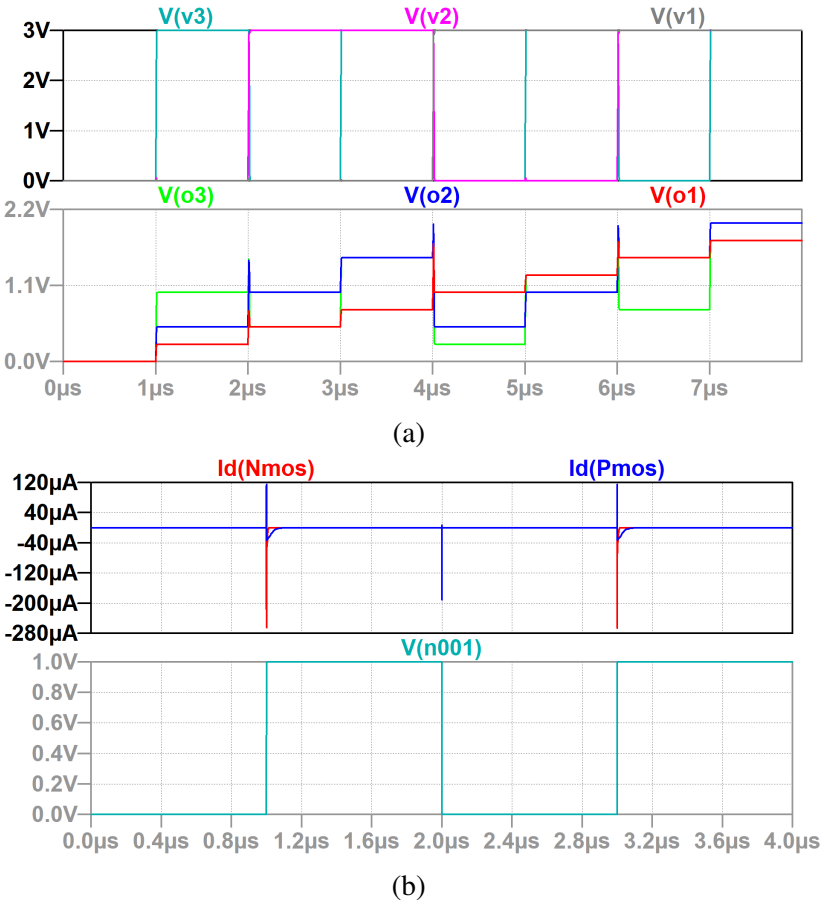


Figure 5.3: The response to inputs and energy consumption of a three-input multiply circuit in simulations. The horizontal axises represents the duration over which the data has been collected for both of the sub-figures. (a) The waveform of the conceptual diagram of a three-input multiply circuit, with the three inputs connected to the same voltage source of a magnitude of $2\ V$, controlled by separate ideal switches. We may see a clear linear combination relationship between the outputs labelled as oX against input combinations vX. (b) The current flow through two transistors of the pass-gate specified (Upper sub-plot) and voltage seen at the input port (Lower sub-plot) of a one-input multiply circuit, with input voltage to be $1\ V$. In both cases, the multiplying circuit is a three-input multiply circuit as shown in Fig. 5.1, with the magnitude of capacitors to be $1\ pF$ and $2\ pF$ respectively.

required.

Furthermore, it is important to note that in parameter matrices, the sign of each element may not consistently be non-negative. In many instances, even when dealing with binary parameters, the system may exhibit a balance of positive and negative parameters within the same layer. Unfortunately, achieving this balance solely through the architecture we have proposed is currently not feasible.

While it may seem logical to include the sign bit within the weighting system, we have opted to simplify the signing process by converging the sign bit-related systems at the summation stage for the sake of general ease and efficiency. This approach not only streamlines the process but also enhances the overall

robustness of the system, making it easier to control and train effectively.

5.3.1 Linear follower and Operational Amplifier-based summing circuit

The utilisation of signed parameters will enable the neural network to effectively exclude invalid information and to identify potential negative correlations between distinct patterns and categories throughout the training process. With respect to the quantized parameters, the differentiation between sign bits and significant digits is crucial for enhancing data storage efficiency. This methodology not only enables an increased capacity for information within the same number of blocks but also facilitates the seamless adjustment of weights from positive to negative values across zero. This process is achieved without introducing the counter-intuitive aspects as typically observed in digital systems where the inverse of a number is represented in the form of its complement. The utilisation of sign bits and significant digits aligns more closely with conventional writing practices, thereby providing a more streamlined and efficient methodology for the ongoing enhancement of weighting tuning and assignment systems.

An efficient method for building summation circuits in analog systems involves utilising Op-Amps. Through the utilisation of negative feedback, the Op-Amp is able to automatically stabilize the voltage difference between its two input terminals. By adjusting the ratio of impedance values in the components connected across the incoming signal, one input terminal, and the output terminal or a ground node, we can easily control the linear transformation of multiple sources. However, it is important to take into account the potential for leakage through resistively connected junctions, as well as the overall static energy consumption and current requirements necessary to operate the device efficiently. These challenges must be addressed in order to optimise the functionality and effectiveness of the system.

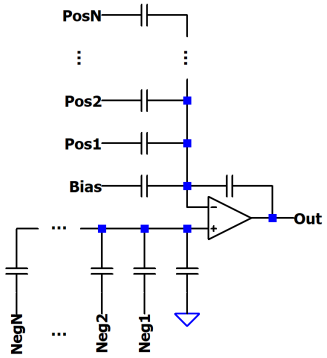


Figure 5.4: The diagram illustrating the revised operational amplifier configuration with capacitive elements. The signal observed at the output terminal denoted as **OUT** remains a combination of all input signals along with a bias signal labelled as **Bias**. It will be perferable to apply low power Op-Amps to further reduce the power consumption

In order to prevent signal leakage among different input signals and the

output node, we have replaced the resistive devices commonly utilised in the design by incorporating capacitors, as illustrated in Fig. 5.4. The relationship between output and input signals aligns with that of resistive Op-Amps, with the amplifiying ratio to be $-\frac{C_{Pos}-C_{Neg}}{C_{OA}}$, where C_{Pos} and C_{Neg} denotes the pair of capacitors connecting **PosX** or **NegX** and the Op-Amp respectively, and C_{OA} is the capacitance between the negative port and the output port of the Op-Amp. Thus, the output is determined by the difference between each pair of input signals.

Through our experiments involving various Op-Amps, including the Precision Low-Cost HS BiFET Dual Op-Amp **AD549**, Low Power Low Noise Precision field effect transistor (FET) Op-Amp **AD795** and Dual Single / Dual Supply Rail to Rail output Low Power FET Input Op-Amp **AD822**, as integrated in the LTspice library, we have observed that the output is significantly influenced by the bias current of these Op-Amps. This influence results in a bias shift in the output while maintaining an absence of any noticeable distortion. Therefore, when integrating non-ideal Op-Amps into the system, it is advisable to implement an additional bias voltage at the input port **Bias** illustrated in Fig. 5.4.

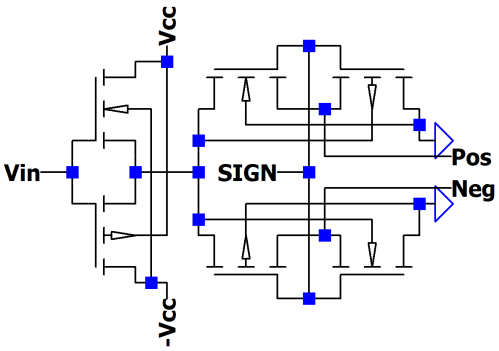


Figure 5.5: The diagram depicting the systematic layout of a signal cell that links the output port of the weighting capacitor array to the two input nodes of the summing amplifier. The linear follower positioned on the left serves to isolate between the capacitors and transmits the voltage to the two push-pull followers acting as the power source. The followers on the right side exhibit an inverted connection to the input **Vin** as well as a ground node **GND**, both being regulated by the same signal. This particular design ensures that only one of the input ports may be connected to a signal with a given magnitude of **Vin**, and **GND** otherwise.

In order to create a signed signal based on this design, the relationship between a duplicate of the suggested input and each combination of inputs for **PosX** and **NegX** as depicted in Fig. 5.4 is regulated by an external signal labelled **SIGN**. To elaborate further, we have implemented three sets of linear followers to serve as AFCs for the regulation. As illustrated in the schematic in Fig. 5.5, the signal **SIGN** can activate either the N-MOS or the P-MOS in a follower pair to reach the saturation region, thereby connecting the input port of either **Pos** or **Neg** in the amplifier to either the incoming signal **Vin** or a common ground **GND**. Furthermore, the opposing connections of the two followers ensure that the connections are in reverse orientation.

The primary limitation of this particular design stems from the extensive usage of the linear follower, which typically operates in a sub-threshold linear region, resulting in less-than-optimal response frequency. Although this approach may be suitable for tasks characterised by nearly constant input or gradual, low-frequency changes, particularly when parameters are maintained consistently, it may not yield a prompt response in high-frequency tasks. Additionally, the utilisation of an amplifier-based network may lead to an escalation in static power consumption. This design may not be the most suitable choice for energy-sensitive or time-sensitive systems. In such instances, it may be advisable to consider the alternative design outlined in the subsequent section.

5.3.2 H-bridge and charge pump-based summing circuit

A potential alternative design for the summation circuit of a single synapse can be conceptualised by considering the output capacitor of the weighting capacitor array, depicted as the uppermost capacitor shown in one schematic in Fig. 5.1, if we designate the port **o1** as the output port of the synapse.

In order to facilitate the integration of the voltage-based signal with outputs from various synapses, we explored the concept of a charge pump. This design involves connecting a charged capacitor in series with an external voltage source to generate a voltage level higher than the supply voltage. This application is in alignment with the principles of Kirchoff's voltage law, which could be described as

> "The directed total electromotive force (EMF) around any closed loop is zero."

In this proposed design, through the interconnection of multiple fully charged capacitors, each serving as the external voltage source for the others, it is possible to achieve an aggregate of these voltages or in mathematical terms, the summation of each weighted input.

In order to accomplish the objective, a recommended method involves connecting a set of double Pole double throw (DPDT) switches to the two plates of the output capacitor to facilitate switching between the charging and summing modes. Additionally, to invert the phase of the signal, an H-bridge was integrated at the output of the summing mode connection to alter the polarity of the connection to the respective capacitors. The basic design and a circuit that employs pass-gates as controlled switches are shown in Fig. 5.6.

The module shall represent an amalgamation of two distinct sub-modules: the H-bridge featuring four switches along with a controlling signal identified as **Sign**, as well as a sample and apply circuit encompassing the capacitor in question and the DPDT switches regulated by two mutually shared external signals known as **Sample** and **Apply**.

The sign indicator **Sign** is akin to the weighting mechanism depicted in Fig. 5.2 and likewise can be considered as being predetermined and either stored in the memory or hard-wired. The representation is generally similar to that of fixed or floating point numbers stored in a typical modern computer system but varies in the method in which a negative value is stored. Despite both systems containing a sign bit, the fundamental principles of the two systems differ. Specifically, in the system we have proposed, the mantissa consistently reflects

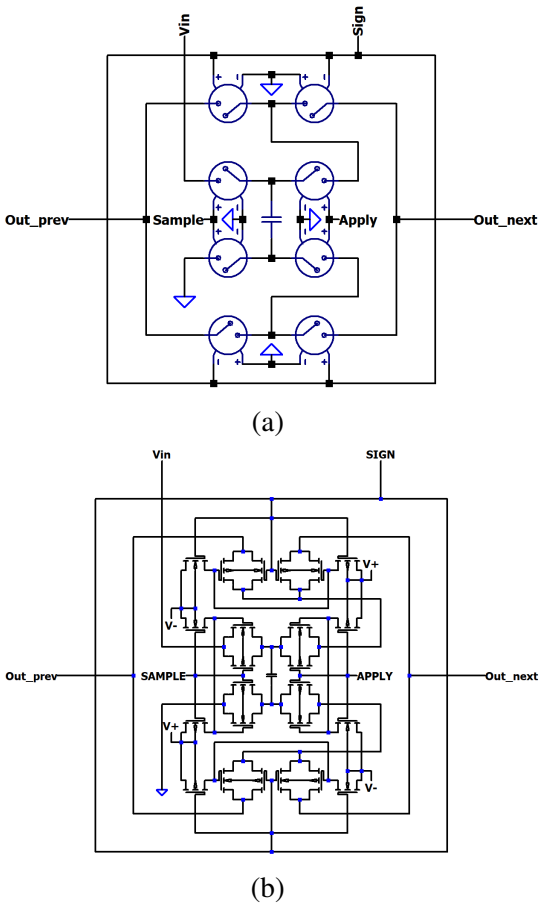


Figure 5.6: The basic concept of a summing cell in a sub-circuit **SUM** as shown in Fig. 5.10, and a diagram of how signal flows in the two phases of sampling and applying. (a) The summing cell, having an input **Vin**, a pair of outputs connecting to either the previous cell (or ground) **Out_prev** or the latter cell (or output) **Out_next**. The cell is controlled by one static signal **Sign** and two dynamic phases of operation **Sample** and **Apply**. (b) The cell translated to a transistor-based sketch, with the same notations employed.

the absolute value of the stored parameter, and any overflow will consistently shift the data to either a positive or negative zero. Considering the fact that in computer systems, negative data are often represented by complement of its inverse, it may be advantageous to incorporate additional **XOR** gates in the MUX of the multiplier if a more conventional interface is desired with a computer system.

The **Sample** and **Apply** signals are critical for the proper functioning of all synapses within a layer. It is imperative that these signals are managed properly to prevent logic conflicts. To ensure data integrity and safety, both signals should not be set to logic high simultaneously. In order to prevent the migration of stored information in the form of charges, it is recommended to have both signals set to logic low and enter the **Hold** stage in steady state to minimise steady-state power consumption. To streamline the operational process, a full functional cycle can be broken down into four distinct stages: the **Sampling** stage, the **Applying** stage and two **Hold** stage in between.

The **Sampling** stage initiates at the rising edge of the **SAMPLE** signal and concludes at its falling edge. During this stage, the system, in conjunction with the weighting capacitor array, establishes a capacitor ladder to generate a weighted output voltage across the summing capacitor as shown in Fig. 5.1. The **Sampling** stage typically lasts at least six time constants, where each time constant is given by $\tau = R_{on}C_{total}$, with the potential for an extended duration during the initial setup. In the expression of time constant, R_{on} stand for the net resistance of the pass-gates seen from the weighting capacitor, while C_{total} is the net capacitance seen from the input node.

Following the deactivation of the **SAMPLE** signal and prior to the activation of the **APPLY** signal, or vice versa, there are two brief **Hold** phases aimed at maintaining stability. The **Hold** stages should not exceed a single time constant to ensure efficiency and minimise the potential charge leakage through the pass-gates.

During the **Applying** stage, the **APPLY** signal is exclusively active. The capacitor is disconnected from the signal source and is connected to the H-bridge for the summation operation. The direction of the summation is determined by the **SIGN** bit stored, ensuring accurate processing of data.

In the system illustrated above, the potential is received through a **Vin** port, and then captured and maintained by the capacitor. Subsequently, it is modified by the H-bridge, which in turn determines the direction in which the potential is ultimately computed.

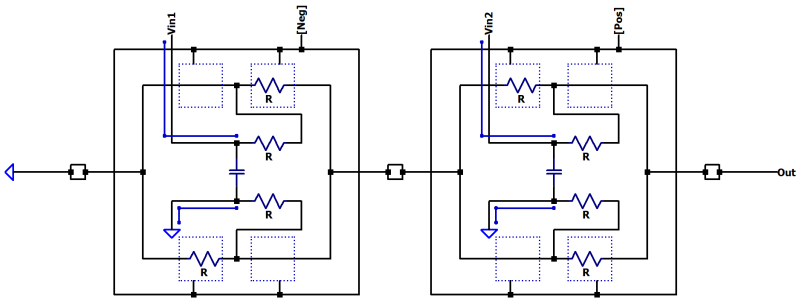
As evidenced in Fig. 5.7b, the manner in which the potential across the capacitors is observed varies according to their respective connections. For the sake of clarity and simplification, we shall designate the upper plate in the aforementioned figures as the positive plate of capacitors (noted as **C+**), and the corresponding lower plate as the negative plate (noted as **C-**). The potential written onto the capacitors in a state of equilibrium shall be derived as follows:

$$U_{\mathbf{C}} = U_{\mathbf{C+}} - U_{\mathbf{C\text{-}}} = V_{\mathbf{Vin}}. \tag{5.5}$$

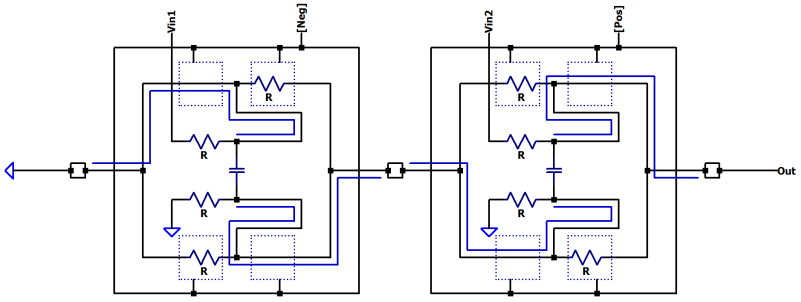
Therefore, in the event that we neglect the leakage, in the scenario depicted in Fig. 5.7b shows, $U_{\mathbf{Cl+}} = \mathbf{GND} = 0V$, $U_{\mathbf{Cr\text{-}}} = U_{\mathbf{Cl\text{-}}} = U_{\mathbf{Cl+}} - V_{\mathbf{Vin1}} = -V_{\mathbf{Vin1}}$, $V_{\mathbf{OUT}} = U_{\mathbf{Cr+}} = V_{Vin2} + U_{\mathbf{Cr\text{-}}} = V_{\mathbf{Vin2}} + (-)\,V_{\mathbf{Vin1}}$. Outputs under different circumstances can be examined in a similar manner.

In practical applications with a transistor-based implementation as shown in Fig. 5.6b, it is possible for the stored signal to leak through the imperfectly opened switches. We can make calculations under the assumption that the signal begins at each plate of a capacitor and ends at the ground nodes. It is important to consider that the resistance of each off-state switch is sufficiently high compared to on-state ones, so we only need to account for the shortest path through the pass-gate arrays between capacitors and the ground.

In this scenario, the path taken from each plate to the **GND** will pass through either one or two switches, resulting in a net resistance of approximately $1\ G\Omega$ for the cases we modify switches using pass-gates as demonstrated earlier. When using a capacitor of $1\ pF$, we can determine that the time constant τ_{leak} falls within the range of about $1\ ms$. The percentage of leakage is estimated to be around 2% of the original value at a low frequency of $10\ kHz$, which makes it



(a)



(b)

Figure 5.7: A two-cell **SUM** system cascaded with reversed **Sign** signal applied at different stages. The blue lines show the signal flow in the given phase. The dotted boxes show the states of the switches. On-state switches have significantly smaller inner resistance than off-state ones and are represented by wires in boxes, while off-state ones are represented by resistors labeled as **R** enveloped in boxes. The resistances of off-state pass-gates are finite yet large enough, so the leakage through these devices are negligible. (a) The system at a **Sample** stage. The signals provided by the input **Vin1** and **Vin2** flow in and charge the capacitors. (b) The system at an **Apply** phase. The signal stored on capacitors flows out and is read as a whole from the port **Out**

considerably longer and readily neglected in comparison. It is strongly advised to deploy devices with increased resistance in the off-state to enhance performance under low-frequency conditions.

5.4 Crossbar Designed MAC

In the context of mathematical modelling neural networks, it is paramount to ensure proper connectivity of neurons based on linear layer matrix guidelines. However, when transitioning to real-world applications on silicon, it is also imperative to establish a systematic approach for organising each component in a sequential manner while minimising overlap and conflict in the layer-wise design of wiring and other elements in addition to the mathematical criteria. In practical

terms, the "crossbar" layout will be used to transmit input to synapses aligned with matrices through parallel wires denoted as **BUS**, with the aggregated outputs will be transferred to a separate set of parallel **BUSes** for subsequent use in the vertical signal lines.

When considering each weighting block as a resistive component, and due to the fact that the signal, particularly in the form of voltage level, is often supplied to high impedance systems for subsequent summation and amplification [224, 225], or is transformed into alternative presentations. It can be inferred that the system remains a modification of a conventional Op-Amp-based summation network with multiple cascades.

5.4.1 MAC in functional blocks

The aforementioned sub-systems have the capability to be integrated and interconnected at various levels, with an inherent component for data storage. This allows the data-storage components interfacing with the weighting subsystem to function as a controlling system, utilising a network of interconnected wires to transmit and scale the analog signal generated during the previous operations. In addition, the system features capacitive components to linearly manipulate the analog data **Va** according to the stored data. Furthermore, the system includes a circuit for sampling the output, aggregating inputs, and transmitting them for additional processing.

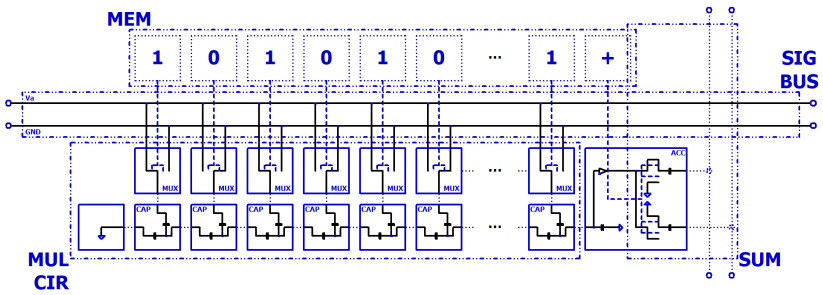


Figure 5.8: The block diagram illustrating the integration of the proposed linear follower and operational amplifier-based multiply accumulate circuit within a system, with a central memory unit as the controlling component. Each encapsulated section denoted by a solid boundary represents a subsystem with tangible hardware implementation, such as the capacitors **CAP**, multiplexer **MUX**, accumulating circuit **ACC** and incoming signal bus **SIG BUS**. The interconnections within the system are regulated by data stored in a dedicated memory unit, indicated by numerical values enclosed within a dotted boundary.

In the case as illustrated in Fig. 5.8, when the system utilises external Op-Amps as summation components, it can be efficiently integrated with the proposed multiply circuit and a memory-stored signal interface for convergence. At the system level, numerous circuits will be arranged in parallel and interconnected through vertical signal lines featuring rounded interfaces, as depicted on

the right-hand side of the diagram. This cascading configuration represents the inputs to an individual neuron in the subsequent layer, before which the wire will be connected to an operational amplifier, as illustrated in Fig. 5.4. Once weight-related data are inputted into the designated memories and accessible for the MUXs, the system can operate in a fully static manner without the need for additional external signals or controlling factors. The weight is managed by externally stored data (enclosed by dotted boxes), as will be identified as **MEM** in Fig. 5.10.

In the diagram, the leftmost bit of the weight signal represents the least significant bit (LSB), while the bit near the sign bit denotes the most significant bit (MSB) of the weight. The weighting data remains in a static state and influences behaviour through surface potential on pass-gate-based switches, which will not consume static-state power. The input signal and its grounding reference are depicted by two wires within the horizontal box, labelled **Va** and **GND** respectively.

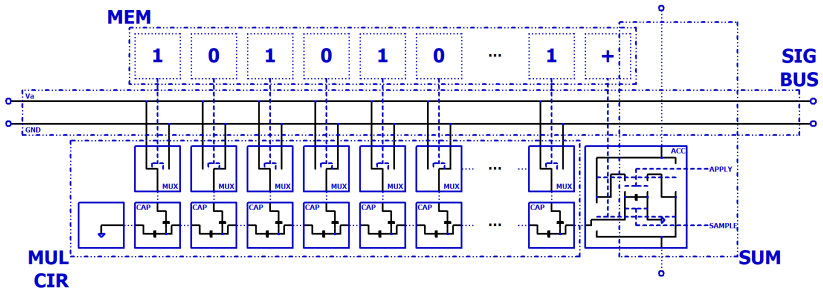


Figure 5.9: The block diagram illustrating the integration of the proposed H-bridge and charge pump-based multiply accumulate circuit within a system, with a central memory unit as the controlling component. Each encapsulated section denoted by a solid boundary represents a subsystem with tangible hardware implementation, such as the capacitors **CAP**, multiplexer **MUX**, accumulating circuit **ACC** and incoming signal bus **SIG BUS**. The interconnections within the system are regulated by data stored in a dedicated memory unit, indicated by numerical values enclosed within a dotted boundary.

The operational block diagram of the multiplying circuit with charge pumps as summation components, along with its interface with the incoming signal flow and the summation system, can be observed in Fig. 5.9. Additionally, the diagram displaying a series of capacitor-MUX pairs as have been presented in Fig. 5.1 and Fig. 5.2. The functioning control of the summation component is overseen by two signals denoted as **SAMPLE** and **APPLY**, as indicated in Fig. 5.6. The summation of the outputs will be transmitted through the interface denoted by the vertical line.

In Fig. 5.10, there is a depiction of a cascaded system that creates a crossbar design for the MAC unit. The memory unit (**MEM**), if not required to be adjustable, can be permanently hard-wired. Although the access to this part is not illustrated, the design flexibility allows for the generation of different charge

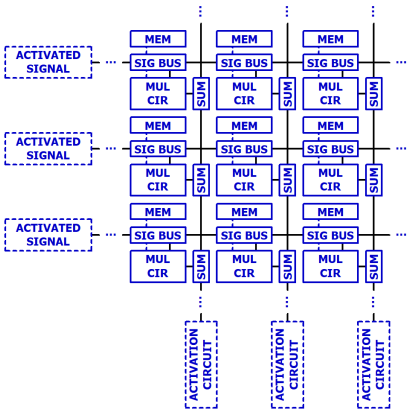


Figure 5.10: The graphic representation of how the suggested multiply accumulate circuit could potentially be linked together in a crossbar configuration and utilised as a MAC for linear operations. The **SUM** circuit will capture, maintain, and aggregate data from all components that are directly linked to it. The incoming signal is received from the side identified as **SIGNAL**while the output signal is produced and triggered by the elements marked as **ACTIVE**. The process of retrieving data from memories is not depicted in the diagram. For the purpose of this project, we can assume that the data is pre-configured and remains unchanged during operation.

levels in the MUXs.

The input signal, assumed to originate from a linear follower AFC, is connected to the weighting circuit labelled as **MUL CIR** based on data stored in memories. The processed signal is then sent to the summation component **SUM** and further transmitted to the **ACTIVE** circuit as input. The **ACTIVE** circuit can be a single AFC of the subsequent layer, or a sample-and-hold (S/H) device as will be introduced in Chapter 6. With proper isolation and maintenance of signal magnitude, the crossbar mesh can be cascaded at various levels to represent different layer sizes in a neural network in a theoretical scenario. Furthermore, it is feasible to efficiently integrate MACs utilising identical weights to amplify parameters exceeding one within the network, as previously addressed.

In the system, when utilising the charge pump-based MAC in a cascaded serial configuration to create a bus, the signals for each separate synapse in a given layer **SAMPLE** and **APPLY** are jointly utilised within the layer. When analysing various layers and factoring in response and charging times, it may be advantageous to ensure that these two signals are in opposite phases in adjacent layers to maintain a seamless data processing flow. Additionally, considering the system's characteristics, it is desirable for the input to remain relatively stable observed from the sampling capacitor at a frequency below a *mega-Hertz* level.

Additionally, bias can be implemented by modifying the **GND** port illustrated in Fig. 5.7 through the incorporation of an additional charged capacitor. This capacitor can be constructed by introducing an extra set of MAC connected appropriately to the common ground and powered by a constant voltage input of 1 to streamline the design and updating processes.

The primary benefit of the proposed design with H-bridge-connected charge pump in comparison to the SOTA is the elimination of amplifier-based circuits

in the summation portion of linear algebra, ensuring minimal static power consumption in low-current settings. Additionally, this design allows for a more versatile range of weighting parameters including non-positive values without the need for an additional connection to the negative port of the amplifier.

One significant issue of the charge pump-based MAC regarding the AFC or the S/H circuit identified as **ACTIVE** is its potential limitations in effectively responding to a consistently fluctuating signal presented in the form of voltage levels, especially when found at the common **Gate** of the two transistor pull-push structure of the AFC. The presence of parasitic capacitance within the AFC could further impact the accuracy of the combined signal's behaviour. Additionally, the possible leakage of charges from the resistive pathways illustrated in Fig. 5.7 could result in a gradual decrease in signal strength relative to its source.

While our analysis allows for an estimation of this decline in conjunction with the AFC's imprecision, a practical application may necessitate the introduction of the S/H circuit as mentioned between the input of AFCs and the output of MACs to maintain overall response stability. This proposed solution does involve a trade-off in terms of static power consumption and the introduction of a certain degree of distortion in the signal represented. The decision to implement such a system is discretionary and relies heavily on the scale of the network and the design of potential AFC under consideration, as well as whether the system can meet the desired outputs and specified application tolerances without the added S/H stage. A detailed discussion will be put forward in Chapter 6.

5.4.2 Space, time and energy efficiency

As previously mentioned, the synapses will predominantly be comprised of a sequence of capacitors, along with various CMOS-based switches that are linked to the **Va** bus and **GND** as shown in Fig. 5.8 and Fig. 5.9. This system will facilitate the utilisation of quantized weights represented by the formula $-1^b \times \sum 2^i \times s_i$, where b denotes the sign signal and s indicates the connection status to the signal source for the capacitor array.

As illustrated in Fig. 5.1, the system comprises a collection of capacitors with values of either **C** or 2**C**, with the option to select a nominal **C** $= 1pF$ for practical demonstrations, and even smaller values for real-world applications. This choice is arbitrary and primarily for ease of calculation, simulation, and demonstration purposes to elucidate the relationship between systematic time, energy consumption, and component magnitude. Additionally, the transistor can be chosen based on different on-state resistance R_{on}. The capacitance at each port is calculated to be $\frac{2}{3}C$, resulting in a system time constant of $\tau = \frac{3}{2}RC$.

In order to achieve a stable state in the system, a response time of $T = 6\tau = 9RC$ can be chosen for an analog steady state, with a maximum frequency of $f = T^{-1} = \frac{1}{9RC}$. The energy consumption for a single capacitor fully charged peak to peak by a single source is given by CV^2, leading to an average maximum dynamic power consumption of $\frac{V^2}{9R}$, where the voltage V corresponds to the signal level **Va**.

Since the majority of power is utilised during the initial time constant while charging the series of capacitors to $(1 - e^{-1})\,V$, the energy consumption amounts to approximately 40% of the upper limit. The peak power consumption

is calculated to be $\frac{4}{15}\frac{V^2}{R}$, which is approximately $0.3\frac{V^2}{R}$.

Furthermore, when considering a single multiplication operation and the necessity for a system to have a resolution equivalent to the 2^{-n} time of its origin, the rate of OPS can be calculated as $\left\lfloor \frac{2}{3RC} \right\rfloor$, with a power consumption of $n\frac{V^2}{9R}$. In addition, the metric of OPS/W can be determined to be $\frac{6}{nCV^2}$.

In a hypothetical scenario involving a *45nm* technology-based design, assuming a 4-bit resolution in the MAC. All capacitors are constructed using metal oxide semiconductor (MOS) capacitors with a **Gate** size of roughly 1000 unit squares and an oxide capacitance factor chosen to be $3 \times 10^{-3} Fm^{-2}$. In this particular case, any parasitic resistance, capacitance, and leakage path are ignored for the sake of simplicity of discussion. In the context of practical applications, it is advisable to utilise devices characterised by reduced leakage levels, or to incorporate the effects of parasitic capacitance and resistance into the calculations. This approach will yield a more accurate and reliable design. The **Gate** size is determined to be $2 \times 10^{-12} m^2$ (each edge length is *1.4*μm, which is significantly larger than the technology restriction to ensure precision in the design of the capacitor). Consequently, the capacitance is calculated to be $6 \times 10^{-15} F = 6fF$, and the resistance of the MUX at its **On** at its On state is simulated to be $2k\Omega$. The voltage supply is set at a nominal $1V$. The aforementioned dynamic mode parameters are as follows:

$$\begin{cases} e & = nCV^2 & = 24fJ \\ \hat{p} & = n\frac{V^2}{9R} & = 222\mu W \\ OPS & = \frac{2}{3RC} & = 55.6MHz \\ OPS/W & = \frac{6}{nCV^2} & = 0.25PHzW^{-1} \end{cases} \tag{5.6}$$

In the preceding examination of the weighting component of the circuit, it has been determined that with fully realisable technologies, it is feasible to achieve a linear operational functional block that operates consistently at its maximum operating frequency with peak-to-peak charging operations and consumes power at the *milli-Watt* level, while attaining computational speeds in the *mega-Hertz* range.

Additionally, if the system remains in a stable state where the sole current pathway for high-frequency noise produced within the system is directed to the ground via the leakage path outlined in the summation section utilising less than ideal pass-gate switches, the magnitude of this pathway still remains relatively low.

When analysing the performance of a system utilising the advanced design and a consistent configuration, it is evident that there is potential for scalability in both frequency and energy efficiency based on certain parameters. By evaluating the magnitude of the voltage signal range peak-to-peak and the capacitance, which is directly proportional to the square of the length of the **Gate** of the transistor-designed capacitors, we can optimise the energy consumption per junction or speed. Furthermore, maintaining a higher resistance level in the leakage path can allow for the utilisation of transistors with shorter **Channel** and lower conducting impedance, thereby enhancing the circuit speed and minimising the power requirements in a linear fashion.

Furthermore, adjusting the peak voltage, in conjunction with the techniques described above, has the potential to significantly decrease the power and

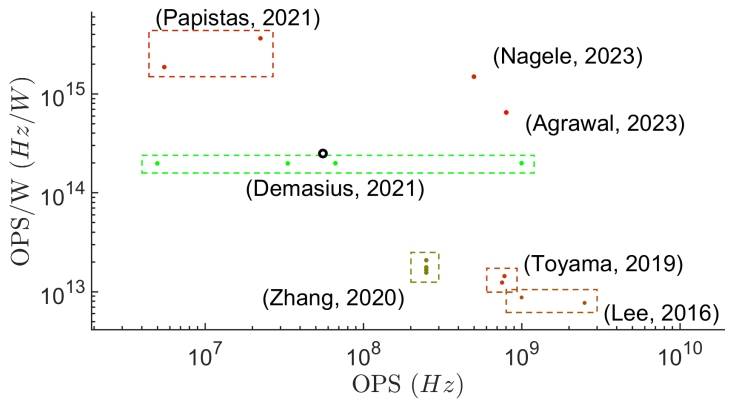


Figure 5.11: The latest state-of-the-art of capacitor-based and field-programmable gate array-based multiply accumulate circuits is outlined based on recent research articles, focusing on floating-point operations metrics [21, 32, 43, 44, 46, 55, 78]. This summary specifically examines time and energy efficiency. The central black circle represents the theoretical performance of the hardware neural network proposed in this work utilising *45nm* technology. A detailed introduction on the examples are illustrated in Tab. 5.1

energy consumption per operation or frequency. For example, by implementing technology similar to previous methods as specified in Eq. (5.6) and selecting a channel length of $450\ nm$ (resulting in a ten-fold reduction in capacitance), operational speeds of up to $556\ MHz$ and power efficiency of $2.5\ PHzW^{-1}$ can be achieved. This approach demonstrates performance levels comparable to leading works in the field, as illustrated in Fig. 5.11.

Table 5.1: The detailed description and rated expectation of time and energy efficiency of the state-of-the-art of multiply accumulate circuits shown in Fig 5.11. The efficiency is estimated according to constant voltage scaling method to a nominally $45\ nm$ scale for each system, assuming its validity.

Reference	Process	Technology	Rated OPS	Rated OPS/W	Demostration Examples
Lee et al. [43]	$40\ nm$	Passive switches and $300aF$ unit capacitors	$1.25\ G$	$6.50\ T$	CIFAR-10
Toyama et al. [55]	$28\ nm$	Gated-ring oscillator	$296.7\ M$	$5.17\ T$	N/A
Zhang et al. [44]	$65\ nm$	Successive approximation register analog to digital converter	$521.6\ M$	$36.7\ T$	Edge detection, MNIST based classification and speech denoising
Demasius et al. [46]	$90\ nm$	**Memcapacitive device**	$230.9\ M$	$795.8\ T$	letter classification
Papistas et al. [32]	$22\ nm$	SRAM-based compute cell	$2.66\ M$	$624\ T$	N/A
Agrawal et al. [78]	$5\ nm$	Population counter circuit	$9.88\ M$	$8.02\ T$	N/A
Nägele et al. [21]	$22\ nm$	Pulse width modulated input analog charge-based accumulation	$119.5\ M$	$341.9\ T$	N/A
This work	$45\ nm$	Capacitive weighting array and charge pump accumulator	$55.6\ M$	$250\ T$	Regression and MNIST based classification

In accordance with the SOTAs of MAC designs presented in Tab. 5.1, and assuming the validity of the scaling rule in this context, we have calculated the rated efficiency concerning operating frequency and power consumption through a single operational procedure. It is noteworthy that the study conducted by Demasius et al.a unique set of non-standard components referred to as “memca-

pacitor", as highlighted in the table. This technology is not compatible with the current manufacturing processes prevalent in the industry. Conversely, the designs developed by Lee et al., Toyama et al., Zhang et al.and Nägele et al.appear capable of achieving higher operating frequencies than the design proposed in this research. Additionally, the work led by Nägele et al.demonstrates the ability to sustain a higher operational power efficiency compared to our study. However, a significant consideration is that their approach entails a more complex design structure and does not allow for a negative weighting factor.

In this work, for a theoretical working environment with the technology specified, ideally, we may also drive the summation part circuit by the theoretical working frequency specified. However, the estimation only looks at the performance of the given weighting system, without real collaboration with the summation part and the activation function part. In situations where the design works, the overall design may not necessarily reach the proposed working frequency and may not always experience power consumption as in the theoretical case. For an even more stable system working with lower frequency requirements and higher energy consumption concerns, it is also necessary to lower the operation frequencies according to the practical requirements and allow for more charging time for the activation function circuit to respond to the signal accumulated, to ensure that the system can always meet a steady state even when the incoming signal is at the linear region of the activation function circuit.

From the design, the footprints can also be estimated by the number of connections in the crossbar network, or connections that For each weighting and summation junction with an n-bit resolution, it is necessary to incorporate $2n$ capacitors and $2n$ pass-gates with inverters. Among these, $n-1$ capacitors should be of a size that is double that of the remaining capacitors. Each pass-gate will utilise a two-transistor CMOS pair. Consequently, this arrangement results in a footprint that totals fewer than $6n$ transistor and $3n-2$ capacitors. Specially, in this part the capacitors used in the footprint counting are assumed to be of the same size and other configuration, while the capacitor of twice the capacitance is assumed to be two identical capacitors connected in parallel.

In the summation component of the design, an H-bridge-based configuration requires 24 transistors and one capacitor, while a linear follower configuration necessitates 6 transistors and 3 capacitors. Therefore, when considering the footprint of a capacitor to be significantly larger than that of a single transistor, the overall anticipated footprint is estimated to not exceed $6n+6$ transistors and $3n+2$ capacitors for each multiply-accumulate junction.

If the parameters are pre-set and hard-wired, it is feasible to exclude all pass-gates within the weighting component, as well as the H-bridge along with the inverters associated with the **Sign** signal, or the amplifying capacitors along with their corresponding pass-gates. Regarding the MUX, for each bit of resolution, the design will conserve one pair of inverters and two pairs of pass-gates, resulting in an overall footprint of $2n$ transistors and $3n-1$ capacitors for this segment. With respect to the summation component of the H-bridge configuration, simplification allows for the omission of four pairs of pass-gates, which leads to a footprint of 12 transistors and one capacitor for this design. In the case of the linear follower-based architecture, it is possible to eliminate two pairs of linear followers and one capacitor. Consequently, for the aforementioned junc-

tion, the footprint will consist of two transistors and two capacitors.

In conclusion, provided that the requisite spacing is implemented in accordance with the technology utilised, each junction exhibiting a resolution of n bits, with memory and optional Op-Amps configured otherwise, one MAC may necessitate a circuit footprint comparable to either $2n + 12$ transistors and $3n$ capacitors, or alternatively, $2n + 2$ transistors and $3n + 1$ capacitors. Furthermore, should further adjustments and fine-tuning be required, an additional $4n + 12$ transistors and one capacitor will be necessary.

5.5 Robustness Analysis

In the realm of pure mathematics and its practical applications through software implementations, careful attention is paid to ensuring precision for optimal use of limited memory resources. Despite these considerations, the accuracy is maintained at a floating point level, adhering to a resolution of 23 bits for single precision as outlined in the IEEE 754 standard [226, 227], and 10 bits for half-precision equivalents. Furthermore, in these scenarios, it is tacitly presumed that all computations, both linear and non-linear, are executed accurately and without any uncertainties.

In hardware implementations where the systems function as look-up tables (LUTs), we have noted discrepancies arising from noise in inter-layer communications, the sub-optimal representation of parameters, and deviations between our theoretical activation function and the actual transfer function observed in practical scenarios. The undisclosed risks present a significant likelihood of compromising precision, thereby calling into question the merit of relentlessly pursuing heightened resolution in the proposed design.

In the subsequent sections, we will examine the impact of parameter distortion resulting from non-ideal conditions on training and validation performance.

5.5.1 Component tolerance in multiply circuit

Using the topology depicted in Fig. 5.1, an output port labelled as **o1** and capacitors denoted as **C1** to **C7** (arranged from top to bottom) are designated. Regardless of the resistors displayed in the diagram, as in a stable condition, all capacitors are considered to be open loop and possess infinite resistance. It is assumed that there is a uniform level of inaccuracy denoted as ε shared among all capacitors, which is introduced to the components through a relative variation expressed as $\frac{\Delta C}{C} = \varepsilon$.

The resultant voltage levels observed at **o1** to **o3** when subjected to voltages **v1** to **v3** respectively can be succinctly summarized as follows:

$$\begin{cases} \mathbf{o1}|_{\mathbf{v1}} &= \frac{1+\varepsilon_2}{3+\varepsilon_1+\varepsilon_2+\varepsilon_{3+4\|(5+6\|7)}}\mathbf{v1} \\ \mathbf{o2}|_{\mathbf{v2}} &= \frac{1+\varepsilon_4}{3+\varepsilon_{1\|2+3}+\varepsilon_4+\varepsilon_{5+6\|7}}\mathbf{v2} \\ \mathbf{o3}|_{\mathbf{v3}} &= \frac{1+\varepsilon_6}{3+\varepsilon_{(1\|2+3)\|4+5}+\varepsilon_6+\varepsilon_7}\mathbf{v3} \end{cases} \tag{5.7}$$

Taking the net relative error of the equivalent capacitor denoted as $C_{1\|2+3}$ as an example, which is caused by the combined effect of multiple capacitors

(specifically identified as $\mathbf{C1}$, $\mathbf{C2}$ and $\mathbf{C3}$, where $\mathbf{C1} = \mathbf{C2} = \frac{1}{2}\mathbf{C3} = \mathrm{C}$) can be further ascertained through additional calculations shown below:

$$\begin{aligned}\varepsilon_\Sigma &= \frac{C_{\textbf{real}}}{C_{\textbf{ideal}}} - 1 \\ &= \frac{1}{\frac{1}{(1+\varepsilon_1)+(1+\varepsilon_2)} + \frac{1}{2(1+\varepsilon_3)}} - 1 \\ &= \frac{\varepsilon_1 + \varepsilon_2 + 2\varepsilon_3}{4 + \varepsilon_1 + \varepsilon_2 + 2\varepsilon_3}\end{aligned} \tag{5.8}$$

In this analysis, the symbol ε represents the relative error observed in each component, with the omission of terms beyond the second order. Subsequently, we have undertaken the conversion of the relative error expression into absolute form in order to facilitate subsequent calculations.

As noted previously, the distribution of ε is evenly distributed among the capacitors, allowing us to simplify the equation to $\varepsilon_\Sigma = \frac{\varepsilon}{1+\varepsilon} \leq \varepsilon$. Subsequently, the summation of two sets of capacitors can be reorganised as $\varepsilon_{\Sigma^2} = \frac{3\varepsilon + \frac{\varepsilon}{1+\varepsilon}}{4+3\varepsilon+\frac{\varepsilon}{1+\varepsilon}} = \frac{\varepsilon}{1+2\varepsilon} \leq \varepsilon_\Sigma$.

Therefore, it is possible to formulate the error of the voltage signal in a manner consistent with the discussion that

$$\begin{cases} \varepsilon_{\textbf{o1}} & = & \frac{2\varepsilon}{3+10\varepsilon} & \lessapprox & \frac{2\varepsilon}{3} \\ \varepsilon_{\textbf{o2}} & = & \frac{2\varepsilon}{3+7\varepsilon} & \approx & \frac{2\varepsilon}{3} \\ \varepsilon_{\textbf{o3}} & = & \frac{2\varepsilon}{3+10\varepsilon} & \gtrapprox & \frac{2\varepsilon}{3} \end{cases}. \tag{5.9}$$

For the sake of efficiency, we may assume that the net relative errors are of a distribution whose root mean square (RMS) value is no greater than $\frac{2}{3}\varepsilon$. Additionally, the impact of factors **o2** and **o3** have on **o1** can be systematically deduced as follows:

$$\varepsilon_{\textbf{o1}|_{\textbf{o2}}} = \varepsilon_{\textbf{o2}|_{\textbf{o3}}} = \frac{\frac{2+2\varepsilon}{4+4\varepsilon}}{\frac{1}{2}} - 1 = \varepsilon \tag{5.10}$$

$$\varepsilon_{\textbf{o1}|_{\textbf{o3}}} = (1+\varepsilon)^2 - 1 = 2\varepsilon \tag{5.11}$$

$$\varepsilon_{\textbf{o1}|_{\textbf{v3}}} = (1+\varepsilon)\left(1 + \frac{2}{3}\varepsilon\right) - 1 = \frac{5}{3}\varepsilon \tag{5.12}$$

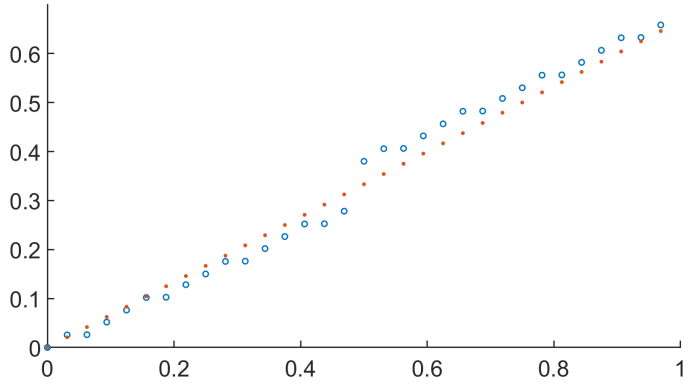
$$\varepsilon_{\textbf{o1}|_{\textbf{v3}}} = (1+2\varepsilon)\left(1 + \frac{2}{3}\varepsilon\right) - 1 = \frac{8}{3}\varepsilon \tag{5.13}$$

An elementary calculation suggests that, for a connected n^{th}, system (excluding the **SIGN** bit), the output adds $\frac{2}{3}2^{-n}$ when connected to a signal source. The noise is approximately $\left(n - \frac{1}{3}\right)\varepsilon$. To ensure effective operation, maintain $\sum_{i=1}^{n} 2\left(i - \frac{1}{3}\right)2^{-i}\varepsilon < 2^{-n}$, leading to $\varepsilon < \frac{3}{2^{n+2}-2}$, where n represents the number of bits integrated into the system.

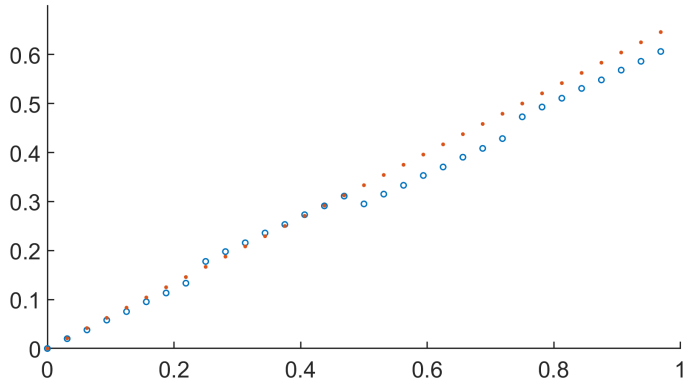
In a scenario where the standard deviation of the component tolerance is $\sigma = 0.1$, the bit resolution is approximately 3. For a 5-bit system, it is advisable to aim for component precision of $\sigma = 0.02$. When utilising readily available devices, achieving a resolution greater than 7 bits becomes challenging in this specific design, especially if the accuracy of the components is lower than $\sigma =$

0.01 according to the worst-case estimation.

For every weight utilised in a specific circuit, the potential values the weighting circuit may display consist of a set of fixed 2^{n+1} numbers. These numbers are evenly distributed within the range of $\left[0, \frac{2}{3}\frac{2^n-1}{2^n}\right]$ in absolute value, with an additional error that is represented by a Gaussian distribution applied to each element.



(a)



(b)

Figure 5.12: Simulated set of weight presented by a non-ideal weighting circuit, each has an element variance ε. The magnitude of each capacitor is given by $\hat{C} = (1+\varepsilon)\,C$. (a) Weighting parameters obtained with Gaussian distributed element variance ε with a mean $\mu = 0$ and standard deviation $\sigma = 0.25$. (b) Weighting parameters obtained with uniformed distribution ε ranged in $[-0.25, 0.25]$. Blue circles indicate the simulated weights presented, and red dots are the proposed weights with the combination of circuit elements. The negative parameters presented is symmetrical to its inverse.

In Fig. 5.12 , we present two potential weight sets that a proposed circuit could generate when dealing with non-ideal components, resulting from a 25% relative error in components. It is evident that while a certain combination of elements may theoretically yield a higher weight, this may not translate into a higher weighted output in reality due to a $\frac{2}{3}\varepsilon$ error if the condition $\varepsilon < \frac{3}{2^{n+2}-2}$ is not met.

In the context of parameter representation, utilising the configuration outlined in the preceding sections, real numbers are stored in memory and are as-

sociated with the weighting operation through the regulation of the switching states of pass-gates. For example, when a specific parameter is assigned a designated value, the binary code written to the memory is named a "command". It is important to note that, given the finite resolution of the numerical system, and acknowledging that the actual capacitance of the capacitors may deviate from the intended critical capacitance, the real parameter presented may not precisely correspond to the value specified in the configuration of the neural network.

While it is possible to organise output signal commands based on actual outputs and seek a closer command to achieve a desired weight, it would be advantageous to incorporate robustness optimisation during the training and validation phases to ensure a more universally optimal loss function in a broader range of potential implementations across different branches. Hence, it is crucial to adhere to the aforementioned training and tuning specification in all practical scenarios.

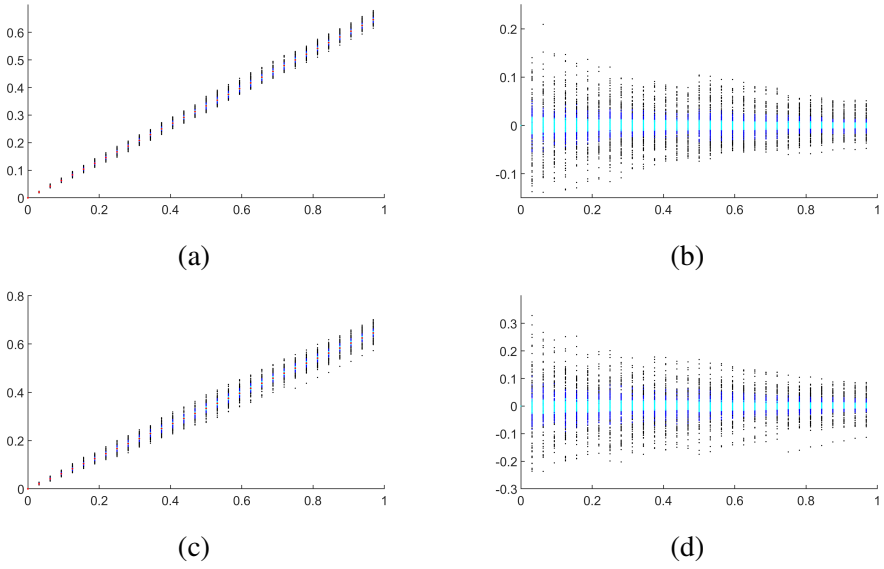


Figure 5.13: The simulated output and relative error distribution by the Monte-Carlo method. The perturbation ε is introduced separately for each component in the simulated components by $C_{\textbf{real}} = C_{\textbf{ideal}} * (1 + \varepsilon(\sigma))$. (a) The output seen at the **OUT** port of a 5-bit weighting circuit, with an input of 1 and a uniform distribution of perturbation on capacitors. (b) The relative error seen at the **OUT** port of a 5-bit weighting circuit, compared with the ideal output, with a uniform distribution of perturbation on capacitors. (c) The output seen at the **OUT** port of a 5-bit weighting circuit, with an input of 1 and a Gaussian distribution of the perturbation on the capacitors. (d) The relative error seen at the **OUT** port of a 5-bit weighting circuit, compared with the ideal output, with a Gaussian distribution of perturbation on capacitors. In all of the figures above, red, cyan, blue, and black dots stand for the ideal case, the case with $\sigma = 0.01$, $\sigma = 0.03$ and $\sigma = 0.1$ respectively.

In order to simulate the sub-optimal parameters for a resilient optimisation algorithm using the identical function as previously employed, we propose simplifying the actual parameter to a proportionally adjusted value relative to its desired value. This can be expressed as $C_{\textbf{real}} = (1 + \varepsilon_{\Sigma}) C_{\textbf{ideal}}$, where ε_{Σ} rep-

resents the permissible deviation for the entire weighting system. To calculate the tolerance for each component ε, one may use the aforementioned formula or resort to Monte-Carlo method, with the approximation that $\varepsilon_\Sigma \approx \frac{2}{3}\varepsilon$.

Fig. 5.13 showcases the potential parameter obtained by the set of non-ideal weighting circuits through a Monte-Carlo method simulation, along with their relative error compared to the proposed parameter to be generated. In the simulation, the capacitors of one single multiple circuit are assigned an additional error that is proportional to their capacitance. All potential commands are established, from 0 to the maximum presentable value. Subsequently, the actual weighting factors are gathered and compared with the theoretical factors for analysis. For example, if we introduce a Gaussian distributed perturbation as shown in Fig. 5.13c and Fig. 5.13d) , within the specified parameter tolerance $\varepsilon = 0.03$, we may observe a perturbation in the parameters of approximately 0.02 from the desired value. The maximum relative error that can be observed is approximately 0.05. This means that, in accordance with Eq. (4.19), the worst-case parameter can be estimated to be approximately $\hat{P} = (1 \pm 0.05)\, P$. In addition, it may indicate a mapping relationship between the variability in elements ε and the perturbation of the parameters in the mathematical model ε_X.

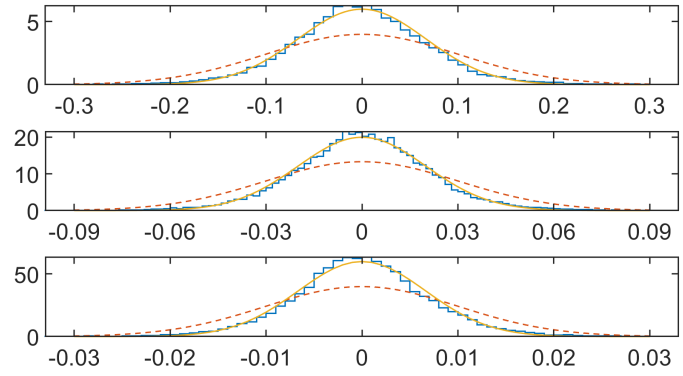
Based on the findings from Fig. 5.14 , it can be inferred that when the variance of the elements is Gaussian distributed, the resulting output is typically comparable to or slightly superior to the anticipated outcome. In contrast, when variance are uniformly distributed, the output is even more favourable, surpassing the expected value by a distribution with its standard deviation $\frac{4}{9}\sigma$. However, for the sake of simplicity in our simulation, we will continue to use σ as the standard deviation for the Gaussian-distributed model, despite the fact that this may result in a sub-optimal output compared to what could be achieved in actual practice.

Based on the aforementioned examples, it can be inferred that a specific distribution of element variance ε has a discernible impact on the systematic tolerance ε_Σ within the analog circuit. Furthermore, this phenomenon can be associated with the perturbation ε_P observed in the respective mathematical expression, thus facilitating a more comprehensive analysis of the forthcoming model to be specified in Chapter 6.

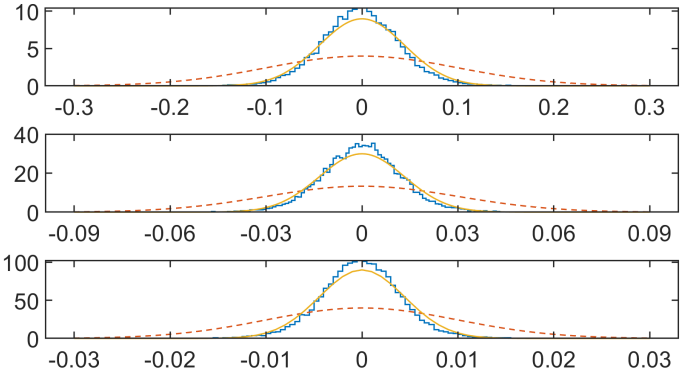
5.5.2 Parameter tolerance in neural network system

Based on the depicted relationship between component tolerance and the relative error in parameters presented, the discrepancy observed at the output of the multiplication circuit corresponds directly to the tolerance of the system. Through the calculations and simulations conducted so far, we can deduce that the relative error in the multiplying component can be approximated as a Gaussian distribution ε with a standard deviation of $\frac{2}{3}\sigma$, where σ symbolises the tolerance of the collection of components. The estimation has been conducted based on the most unfavourable scenario, which represents a significantly more adverse outcome compared to other possible error distributions as recognised within industry manufacturing standards.

In the Modified National Institute of Science and Technology (MNIST) dataset-based classification task outlined in this study, as shown in Fig. 5.15a, a



(a)



(b)

Figure 5.14: The probability density function (PDF) of weights generated by Monte-Carlo method of relative weight variance, and Gaussian function as reference. (a) Relative error generated by Gaussian distributed variance ε with stand deviation σ in each element (blue broken lines), PDF for Gaussian distribution with a standard deviation of $\frac{2}{3}\sigma$ respectively (yellow curves) and standard deviation of ε respectively (orange dashed curves). (b) Relative error generated by Uniform distributed variance within $[-\varepsilon, \varepsilon]$ in each element (blue broken lines), PDF for Gaussian distribution with standard deviation of $\frac{4}{9}\sigma$ respectively (yellow curves) and standard deviation of σ respectively (orange dashed curves). From upper to lower subplot in each sub-figure, the element variance ε has its corresponding standard deviation σ to be 0.1, 0.03 and 0.01 respectively.

randomly selected validation case of a hand-written digit, which should ideally be identified as the number **4** (or belong to class **4**) is fed to a well-trained neural network. The figure presented illustrates the output of a vector consisting of 10 elements as observed at the **logits** layer. Both scenarios—including those with and without parameter perturbation—are included for comparison. The output generated by this layer represents a non-normalized score that correlates with the confidence level poised to be processed by the subsequent **softmax** layer, or it can be translated into the index corresponding to the category with the highest confidence via a **max** layer. In the instance presented, assuming an optimal network configuration, the correct class **4** is projected to yield a confidence score of approximately 13, whereas the highest confidence score for any other class,

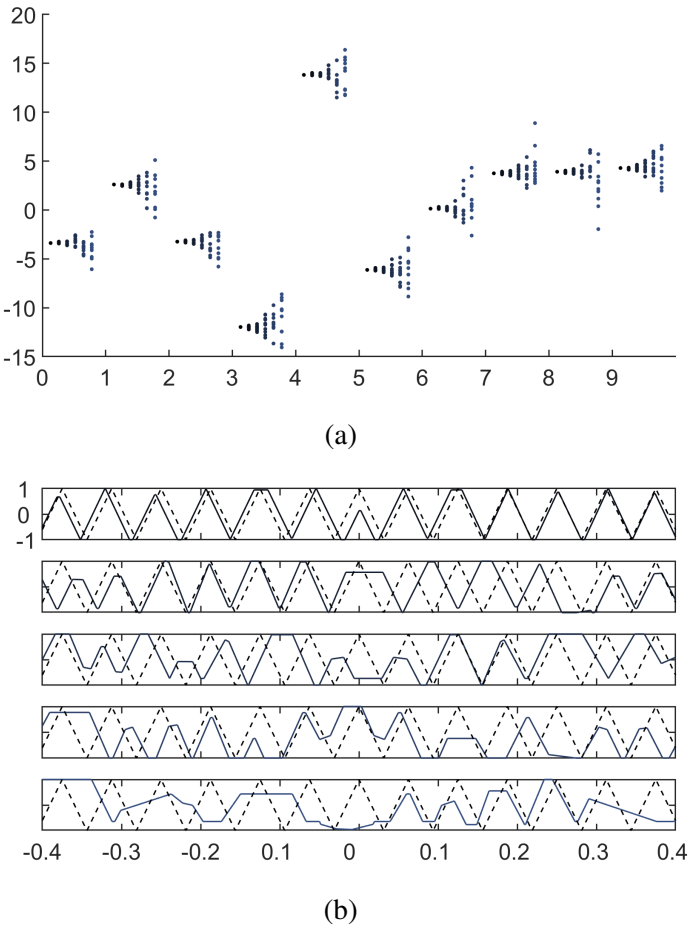


Figure 5.15: The outputs of well-trained and tuned neural networks with ideal activation functions and perturbed parameters are compared with the ideal case. (a) A network trained with full resolution Modified National Institute of Science and Technology dataset-based classification task, with an input of image showing a hand-written digit **4**, transforming from $\mathbb{R}^{28\times 28}$ input vectors to $\mathbb{R}^{10}$ output vectors at the final logistic layer, with two hidden layers sized of 28 and 14 each, before entering the **max** layer. For each category from 0 to 9, the leftmost dot indicates the ideal case output, and to the right are outputs with perturbed parameters. (b) Output of the 5^{th} bit for a regression network designed for Gray Code ADC representation, transforming from $\mathbb{R}^1$ inputs to $\mathbb{R}^1$ outputs, with three hidden layers sized of 4 each before entering the **sign** layer. For each subplot dashed lines indicate the ideal case outputs and from uppermost to lowermost solid lines indicate outputs with perturbed parameters. The relative error for each case in the two examples is 0.01, 0.02, 0.05, 0.1 and 0.2 respectively.

specifically class **9** , is anticipated to be around 4.3, as denoted by the leftmost dot in each respective column. It can be noticed from the illustrated case that it is possible to specify a distinct boundary separating the classes in this ideal-case network.

By introducing a Gaussian-distributed perturbation with its RMS value to be 0.2 times relative to the original values of the parameters, it is noted that the confidence level for class **7** can reach 8.9, while the lowest confidence level for class **4** is 11.8. in the worst-case scenario. Although these two cases do not

occur simultaneously, there is a significant risk that a boundary may no longer be able to be effectively differentiated between each class. Consequently, while the system is still capable of producing accurate output, the network's confidence in correctly identifying the number **4** decreases from 99.9% to 94.3%. Additional simulations suggest that a tolerance of 0.2 will likely be the highest amount within which the network can experience a slight decrease in accuracy when compared to the optimal scenario specified for the network (typically ranging from 92% to 90% in the majority of simulations conducted with Monte-Carlo method by introducing errors to the parameters of the neural network.

Table 5.2: The validation loss in cross entropy and validation accuracy for the Modified National Institute of Science and Technology dataset-based classification task with Gaussian distributed variability added to parameters. Ten iterations of simulation are applied.

Standard Deviation	validation Loss			validation Accuracy		
	min	avg	max	min	avg	max
$\sigma = 0$	1.5165			**0.9223**		
$\sigma = 0.01$	1.5164	1.5165	1.5167	0.9221	0.9225	0.9229
$\sigma = 0.05$	1.5154	1.5167	1.5173	0.9209	0.9220	0.9234
$\boldsymbol{\sigma = 0.10}$	1.5165	1.5187	1.5219	0.9163	**0.9201**	0.9234
$\sigma = 0.20$	1.5229	1.5264	1.5325	0.9063	0.9128	0.9168
$\sigma = 0.30$	1.5324	1.5393	1.5512	0.8907	0.9013	0.9072
$\sigma = 0.40$	1.5592	1.5712	1.5912	0.8522	0.8727	0.8822
$\sigma = 0.50$	1.5734	1.6237	1.6587	0.7876	0.8214	0.8709

Tab. 5.2 presents the impact of incorporating Gaussian distributed tolerances on the parameters of a well-trained neural network for operating classification tasks. The tolerances is chosen with various standard deviations in relation to the nominal values of the parameters obtained during training without tolerances. It can be observed that performance degradation is minimal for $\sigma \leq 0.01$, and only marginal for $\sigma \leq 0.10$. In practical applications, it is advisable to utilise components with tolerances of no more than 15%, which aligns with the condition $\sigma = 0.1$ as shown in tab. 5.2. In the context of classification tasks, the developed HNN in this study demonstrates a satisfactory level of resilience and accuracy when compared to its software counterpart when realistic components are employed.

In the regression task depicted in Fig. 5.15b, where the system is designed with a minimal number of neurons as demonstrated in Fig. 4.6, there is an observable trend of increased susceptibility to parameter variances leading to output corruption. It is important to note that even a minor relative error of 0.01

can lead to significant fluctuations in the pre-activation signals. This may subsequently cause phase shifts in the activated signals or potentially distort the output function altogether. Furthermore, increased variances can hinder the system's ability to generate any recognisable output. Through further simulations, it is evident that the low redundancy level and precise parameter selection render the system highly sensitive to inaccuracies in parameters and activation functions. Additionally, the regression nature of the weighting matrix exacerbates error accumulation within layer communications and prevents correction by neighbouring neurons. Thus we may conclude that for analog systems where tolerance is not negligible, training methods for "ideal" systems may be insufficient.

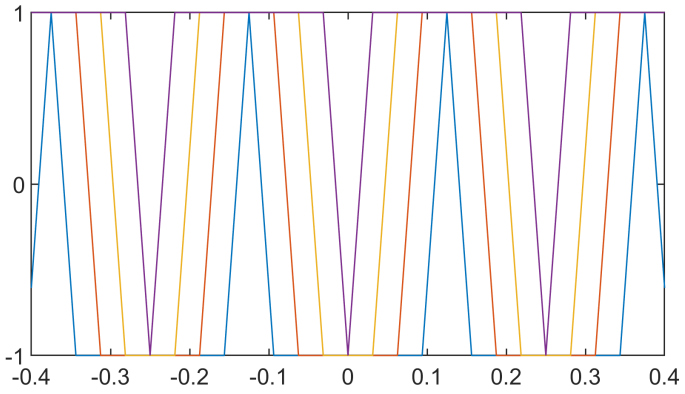


Figure 5.16: The optimal outcome of a single hidden layer within the neural network regarding a Gray Code analog-digital converter-centred regression task, as portrayed in an ideal scenario, where each distinct colour represents the output of an individual neuron.

In the context of regression, a practical approach to enhance the usability of the overall system is to implement an additional fine-tuning process. During the fine-tuning phase, it is essential to focus on minimising the discrepancies that may arise from the non-idealities inherent in practical implementations. Ideally, each layer's output should resemble the form depicted in Fig. 5.16, which corresponds to the philosophy illustrated by Fig. 4.7. By utilising a layer-by-layer fine-tuning process and incorporating additional neurons to align the sum of neuron outputs with the corresponding curves, we can effectively improve the system tolerance. To achieve this, we can systematically reduce the divergence between the activated signal of the practical system and the neuron output predicted by the idealized model. This can be accomplished through a layer-wise training and interval adjustment process for each respective layer.

5.6 Conclusion

The proposed design, aimed at modifying linear operations in floating points through a analog computing device of the multiplication and summation operations, has demonstrated a computing capability reaching *mega-Hertz* levels. This design operates in parallel with a reduced number of components when compared to arithmetic logic units (ALUs) and control blocks that interact with

memory. Additionally, this design allows for constant parameters to be stored statically while processing inputs dynamically in memory. Furthermore, it facilitates the application of a processing-in-memory (PIM) approach with a direct interface to analog or digital inputs and outputs.

The proposed design, driven by its low-current requirements, demonstrates a notable efficiency in time and energy consumption when compared to the state-of-the-art. This efficiency is achieved using only industry-standard technologies. Additionally, the system boasts high scalability in resolution and can seamlessly integrate into a crossbar design, allowing for further expansion in network size as shown in Section 5.4. The components of the design are highly standardised, ensuring efficient use of silicon space.

With an innovative ladder design in the weighting system, the impedance observed at the input remains constant, leading to a proportional relationship between current supply and energy consumption based on the system's bit-resolutions. Additionally, our design allows for convenient adjustment of relative error compared to ideal outputs, enabling us to establish a connection between component variance and systematic tolerance of the MAC. This in turn enables us to conduct an overarching estimation and simulation regarding the potential inaccuracies of neural networks or equivalent systems if the topology and configuration is given.

One of the key challenges associated with this system is the inherent trade-off between accuracy and the benefits it offers. In the context of simulating the curve fitting necessary for the **logit** layers of the neural network analytically designed for the Gray Code analog-digital converter (ADC) task, it has been observed that error accumulation can significantly compromise the integrity of the output. Therefore, we can deduce that in precision-critical systems, particularly those involving regression requirements, a hardware implementation of this technology may not be advisable without further enhancements and refinements.

A suitable use case for this technology is to integrate it as a front-end interface for sensor input systems or to use it as stand-alone modules to classify multiple types or to manipulate data for further processing. The power efficiency of the system enables operation with a minimal power supply, while its compact footprint facilitates seamless integration with other systems, without significantly increasing chip size or weight. Additionally, the primary drawback concerning response time limitations can be effectively mitigated in a nearly steady-state environment. In Chapter 7, we will further explore potential applications, as well as adjustments to neural network topology and training algorithms.

Chapter 6

Systematic Analysis on Performance and Behaviour

In the technical literature, there has been significant research interest in exploring the non-ideal behaviours of systems utilising hardware neural network (HNN) implementation. It has been identified that factors such as component tolerances, which may result in distortions in parameters, activation functions, and inputs, warrant attention and consideration. It is worth noting that the concept of "robustness" within the community tends to be discussed on a local level pertaining to individual inputs, with less emphasis on global behaviours. Furthermore, there is a noticeable lack of a comprehensive approach to effectively model how errors within each layer's operations propagate between layers.

In this chapter, we will be presenting a formula for estimating such factors based on the model we have established in Section 6.2, with the aim of establishing a connection between these factors and the overall system performance. Additionally, we will be outlining specific guidelines for implementing HNNs using the modules we have proposed and highlighting any limitations of this system.

6.1 Overview

In developing hardware neural networks (HNNs) systems, a key consideration is whether a HNN can efficiently and accurately perform predictions or calculations comparable to a mathematical model or software implementation of a neural network. It has been hypothesised that achieving a mathematical equivalence between the hardware system and the mathematical model will result in equivalent behaviour. Given the ability to represent all solvable mathematical problems with neural networks of infinite size and precision using various activation functions theoretically, the challenge lies in obtaining a close-to-ideal activated value by non-ideal activation function computing devices with finite size and precision in practice, while accounting for inaccuracies in functional blocks due to noise in sampling and transmissions and computing errors in algorithm design and component non-ideality.

We have examined the performance of each functional block outlined in preceding chapters and offered a succinct assessment of their resilience against device variability on an individual basis. However, there remains a significant

gap in the discourse regarding the construction of a HNN incorporating these two modules, as well as the potential propagation of perturbations and failures throughout the system.

The emphasis in the current research revolves around addressing the issue of error propagation in neural network implementation. This is primarily concerned with the potential impact of corrupted inputs, training data, or signal flow on the precision of the HNN system. It is important to note that under the limitations of hardware variability, any inaccuracies in the implementation can lead to a heightened risk of prediction failures.

Nevertheless, establishing a direct correlation between the learning outcome and the system-level disturbances resulting from noise and imprecision of the parameters can be challenging, particularly when employing a statistical-based methodology. Despite advancements in techniques such as parameter re-estimation and matrix-based computations [228], discrepancies in predictions persist among different methods, even in regression scenarios where output standard deviations serve as reliable indicators of implementation success. Furthermore, tasks commonly used as benchmarks for network models and optimisation algorithms fall short of achieving human-like resilience against noise or attacks introduced into input samples [229].

We will base our discussion on the local stability of the system derived from global stability. By utilising the concept of the global Lipschitz constant, we can determine the maximum tolerable variability possible in inputs, parameters, and non-polynomial activation stages without experiencing a loss in accuracy.

In addition to investigating the ability of HNNs to achieve its intended learning outcomes, it is imperative to examine the systematic parameters related to latency and power consumption associated with a specified neural network-topology. The primary aim of utilising analog designs of HNNs is to improve operational efficiency, specifically regarding power consumption and response time—two aspects that represent significant constraints in digital neural network . Furthermore, the energy efficiency of the overall design deserves careful consideration.

Moreover, compared to alternative implementations, a hardware-based neural network system may require diligent oversight of temporal sequencing due to operational frequency and signal transmission latency. While a majority of devices might demonstrate stable timing, the responses can exhibit considerable variability in both time and frequency. In practical applications, signals may be subject to distortions arising from group delay and frequency-specific filtering attributable to component mismatches. Additionally, the ability of a single output source to simultaneously drive multiple loading circuits is vital to determining the dimensions of each layer without relying on external buffers and amplifiers, although this may introduce a certain degree of distortion.

In this chapter, we will conduct an analysis of the activation function circuit (AFC) design and the two proposed designs of the multiply accumulate circuit (MAC). The aim is to provide a comprehensive overview of the overall performance, advantages, disadvantages, potential risks, and expected working conditions of the circuits. Specifically, our objectives are as follows:

1. Evaluate the propagation of inaccuracies throughout the hardware neural

network-level implementation.
2. Demonstrate the prediction of error accumulation through examples, highlighting how it may impact the learning outcome.
3. Identify and discuss the potential drawbacks and limitations of each combination of activation function circuit and multiply accumulate circuit designs.
4. Assess the changes needs to be made to maintain the working stability for each combination of AFC and MAC.

In the initial sections, we will begin by formulating an equation for the accumulation of errors in a perceptron. Subsequently, we will extend this analysis to a deep neural network (DNN) featuring k cascades of uniform perceptron layers. In the ensuing portions of the project, we will underscore the obstacles present in both scenarios if no additional modifications are made, along with a separate discussion on the analysis and subsequent solutions.

Moreover, it is possible that there are additional considerations related to redundancy and effectiveness of networks, such as a "dying" neuron on the brink of expiration that maintains a consistent activated value regardless of the inputs it receives [230], as well as the dimensions and complexity of hidden layers [120]. These aspects will not be explored in this chapter. The prevalent issue of adversarial attacks, whereby disruptions are introduced into input data [231, 232], will only be touched upon briefly in terms of random input variability ε_X. Likewise, the strategies for training and refining models—such as incremental learning with noise introduction [107] and the integration of external decoder architectures [229]—both existing and proposed, will not be expounded upon.

6.2 Performance and Robustness

The utilisation of neural networks may experience a decrease in performance when components in practical cases exhibit variability, as evidenced in Sections 4.5 and 5.5. Testing functional blocks and perturbed systems separately reveals a non-linear relationship between the decline of learning outcome and component tolerance. While there is no definitive benchmark for evaluating the performance of individual layers or modules within layers, as discussed in Section 3.3, it is possible to simulate and analyse the systematic variation in outputs and performance changes arising from that are of great interest to the computer science community [233].

In this section, we will focus on analysing the impact of output perturbation due to the variability of each functional block and signal transmission within the module. We will evaluate the overall performance in terms of accuracy and loss on the output side of the module using mathematical models of the system. The procedure for this analysis is outlined as follows:

1. Determine the perturbation at each layer due to element variability.
2. Calculate the perturbation of linear and non-linear layers pair by pair based on layer-wise inaccuracies.
3. Develop a formula to describe system-level perturbations accordingly.
4. Establish the minimum threshold for systematic robustness to ensure no decrease in accuracy in classification tasks.
5. Validate the predictive models established above.

6.2.1 Effect of variability in components

Through an analysis of the diverse components utilised in the deployment of neural networks, we have made individual projections on the margin of error present in the accuracies and losses of the HNNs simulated. Furthermore, as a hierarchical structure consisting of various tiers of distinct functional units, we are also able to highlight the correlation between component variability and its impact on the ultimate outcome of a particular system.

In the case of a single linear layer with an output error margin of ε_L, which will be passed to an ideal activation functions, serving as input for the non-polynomial layer, we may observe a magnified error margin of $\theta\varepsilon_L$ based on certain considerations as shown in Eq. (3.8) by first order Taylor expansion. This factor θ represents the approximation of gradient observed at the activated value under ideal conditions. However, there may also be an additional error introduced due to imperfections in the AFC, requiring additional consideration in the analysis.

To streamline the evaluation process, we simplify by setting $\theta = 1$ and assessing the impact of a given error distribution ε on the activated value of a specific linear activation function layer operating solely with identity transformations. This distribution ε impacts all parameters and the input provided, denoted by $\hat{p} = (1 + \varepsilon)\, p$, where p represents the parameter being studied.

In the case of receiving a corrupted set input signals at each layer, assuming that the relative error of the linear transformation part and the incoming signals are relative to their magnitude. Note the factors of each pre-activation value y_j contributed from input signal x_i to be $y_{i,j}$, with the relationship $y_{i,j} = p_{i,j}x_i$, the relative error seen at the factor can be calculated as $\varepsilon_P + \varepsilon_X$, where ε_P and ε_X are shared distribution of relative errors among transforming matrix P and input vector X, for each element p and x. Additionally, for each element y_j in the pre-activation value, there may be an absolute error of the sum of $\sum (\varepsilon_P + \varepsilon_X)\, p_{i,j}x_i$, which also equates to a relative error of $\varepsilon_P + \varepsilon_X$ when disregarding second and higher order factors. Furthermore, at a modular level, we are able to establish a correlation function that

$$\begin{aligned} \Delta Y &= \hat{P}\hat{X} - PX \\ &= (P + \varepsilon_P P)\,(X + \varepsilon_X X) - PX \\ &\approx \varepsilon_P PX + P\varepsilon_X X. \end{aligned} \tag{6.1}$$

In this context, P symbolises the matrix utilised for linear operations, with X denoting the input vector of the matrix and Y representing the output-side pre-activation vector of the transformation. Consequently, the relative error ε_Y observed at the output side of the linear operation section will be a result of the combined errors of $\varepsilon_P + \varepsilon_X$ obtained through element-wise operations. It should be noted that in the case of a purely linear system, a linear accumulation of errors will be evident on the output.

In regards to the activated values of non-polynomial layers, it is advised to omit the gradient-related scaling factor θ, and assume it takes a maximum value of 1 during worst-case simulations. We may present the distorted activation

function, which is denoted by $\hat{\alpha}(\cdot)$, in the following manner:

$$\hat{\alpha}(\cdot) = (1 + \varepsilon_\alpha)\,\alpha(\cdot) + \varepsilon_n, \tag{6.2}$$

where the ideal-case activation function is noted as $\alpha(\cdot)$. . The relative error associated with the error induced scaling of this function is denoted by ε_α, while the noise component, quantified by the absolute error, is represented as ε_n. As per previous derivation shown in Eq. (4.20), it is important to note that if we consider the activation function $\alpha(\cdot)$ to be a unity transformation in the worst-case scenario, the error $\Delta Z(Y)$ observed in the activated value relative to the input Y will be $\varepsilon_\alpha \alpha(Y) + \varepsilon_Y Y + \varepsilon_n$. The relative error in this case will be $\varepsilon_Z = \varepsilon_\alpha + \varepsilon_Y + \frac{\varepsilon_n}{\alpha(Y)}$. When we introduce a lower gradient or saturation regions where $\theta \approx 0$, assuming an average scaling factor of θ (where $\theta \leq 1$), which means $\alpha(Y + \varepsilon_Y Y) \approx \alpha(Y) + \theta\varepsilon_Y Y$, and the activated value does not have a significant static error or direct current (DC) offset ε_n, there may be a relative error modelled as follows:

$$\varepsilon_Z \approx \varepsilon_\alpha + \theta\varepsilon_Y + \varepsilon_n, \tag{6.3}$$

where the assumption is made that the expected value of the activated value $\langle Z \rangle$ is 1.

In order to assess the impact of the static error caused by activation functions in practical scenarios on the linear operation of the subsequent layer, it is also possible to encounter distortion generated from non-ideal activation function components introduced into a linear layer. We may also have

$$\begin{aligned} \Delta Y &= \varepsilon_P PZ + \varepsilon_Z PZ \\ &\approx \varepsilon_P PZ + \varepsilon_\alpha PZ + \theta\varepsilon_Y PZ + \varepsilon_n P. \end{aligned} \tag{6.4}$$

In instances where $\langle Z \rangle = 1$, it is possible to assert that the overall trend of error accumulation within the network exhibits a lesser degree of linearity.

Assuming the input $\hat{X}$ enters a non-ideal dual-layer configuration consisting of a linear layer with transformation matrix P succeeded by a non-polynomial layer denoted as $\alpha(\cdot)$, the total distortion observed at the activation layer's activated value will be:

$$\begin{aligned} \Delta Z &= \hat{\alpha}\left(\hat{P}\hat{X}\right) - \alpha(PX) \\ &= \varepsilon_\alpha \alpha(PX) + \theta(\varepsilon_X + \varepsilon_P) PX + \varepsilon_n. \end{aligned} \tag{6.5}$$

The calculation will result in the determination that $Z = \alpha(PX)$, $\varepsilon_Z = \varepsilon_\alpha + \frac{\varepsilon_n}{Z} + \theta\frac{PX}{Z}(\varepsilon_X + \varepsilon_P)$. Note $\theta_r = \theta\frac{PX}{Z}$, in a system of k cascading layers where linear and non-linear layers are used alternately if input is observed at a linear layer with a relative error of ε_X, the following relationship may also apply:

$$\varepsilon_Z = \frac{1 - \theta_r^k}{1 - \theta_r}\left(\varepsilon_\alpha + \frac{\varepsilon_n}{Z} + \theta_r \varepsilon_P\right) + \theta_r^k \varepsilon_X. \tag{6.6}$$

In accordance with the previous derivation Eq. (6.6), it has been established that for structures consisting of $2k$ layers where k is an integer, the possible degree of inaccuracy on the output will not surpass the expression that

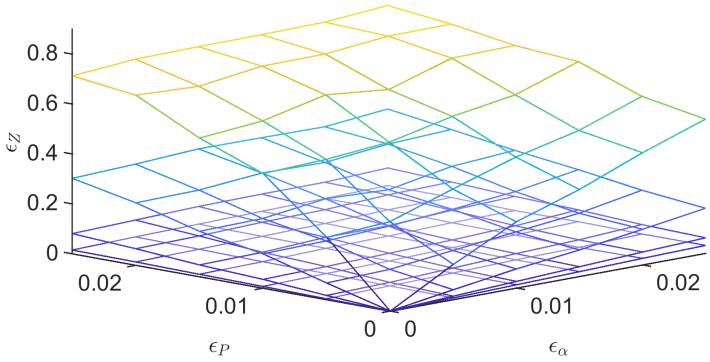


Figure 6.1: The relative error seen in each of the layers of a well-tuned network with activation functions and parameters separately perturbed. From the lowermost to the uppermost meshes are the relative error seen at the first to the fourth hidden layer.

$\varepsilon \approx k\left(\varepsilon_\alpha + \frac{\varepsilon_n}{Z} + \theta\varepsilon_P\right) + \varepsilon_X$. In the case where there is a zero gradient region within the domain for the activation functions, the total inaccuracy will be confined to the sum of $\varepsilon_\alpha + \frac{\varepsilon_n}{Z}$. In this case, the errors will not propagate within the neural network.

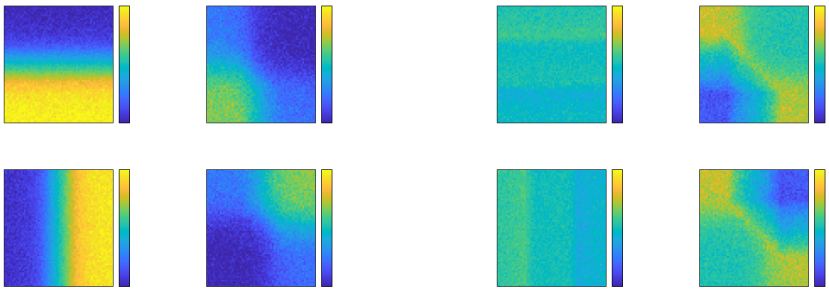
In the task of executing as an analog-digital converter (ADC) based on Gray Code, as outlined in Section 5.5, utilising a layer size of 4 neurons each, and deploying a scenario with two hidden layers capable of delivering 5 bits' resolution, there is potential for a noticeable distortion of the signal at the **Output** layer if the parameters are slightly altered, resulting in a relative error of 0.2 time of the proposed values.

In line with Eq. (6.6) and the matrix given in Section 4.3, it is plausible to anticipate a relative error escalating as we progress from the input of each linear layer to the activated value of the corresponding activation layer, primarily due to the influence of the term θ_r, conservatively estimated to have a value of 4 for this scenario, contributed by the transforming matrix as specified in Eq. (4.14) and a unity gain linear region of the proposed activation function. Furthermore, as depicted in Fig. 6.1, the relative error evident at each layer ε_Z increases proportionally with the relative error terms ε_α and ε_P, intensifying rapidly as the depth of the layers increases in a nearly exponential manner.

6.2.2 Effect of variability on neural network level

Based on the previous discussion, it is evident that adjusting the parameters of neural networks based on the feedback from the loss functions is an effective approach to aligning the functional relationship between high-dimensional inputs and outputs, thereby enhancing the decision-making accuracy. However, it is important to acknowledge potential risks associated with this method, as it may overlook the relationships between input features and the target output features, which are critical for understanding the characteristics of the given task and thus improve the generalisation. Utilising a model that relying solely on linear separation could potentially give rise to unforeseen disruptions stemming from

the element-wise variation caused by the formula for relative error propagation outlined in Eq. (6.6).



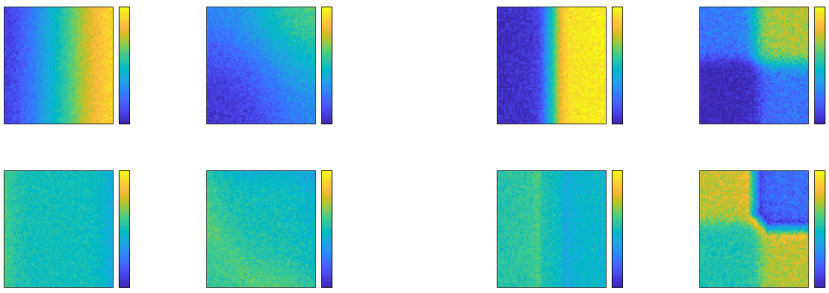
(a) (b)

Figure 6.2: The output of four neurons involved in the first two layer of a simple network representing an **XOR** gate's output with a $[2 \times 2 \times 1]$ sized network with ideal-case parameters and circuit transfer function shown in Chapter 4 as its activation function and relative errors of ε_P, ε_α and ε_n set to be 0.5. (a) The output of the non-ideal network seen at each of its hidden neurons. For the first hidden layer (two sub-plots on the left), the output illustrates the ability to linearly separate the input space by two parts. At the second hidden layer (two on the right) exhibits a weaker ability to drive two parts in the input space separately. (b) The difference the non-ideal case output and the ideal-case output. In the first layer (two sub-plots on the left), the difference is of medium value, while in the second layer (two sub-plots on the right), the difference reaches a greater magnitude.

Through the outcome of a basic experiment depicted in Fig. 6.2 replicating the functionalities of an **XOR** gate using the neural network, it has become evident that as demonstrated in Eq. (6.6), , the relative error associated with the activated value can significantly accumulate due to parameter tolerances and computational inaccuracies, particularly in relation to the depth of the layer. Besides, we may also hypothesis that with the proportion between parameters unchanged, a scaling factor applied on all of the parameters may also affect the robustness against parameter variance and distortions in activation function calculations.

With the simulation based on the example presented in Fig. 6.2, when scaling the parameter matrices uniformly, it is evident from Fig. 6.3 that the trend of perturbation accumulation aligns with our initial hypothesis. When dealing with larger weighting parameters, the model's robustness against parameter changes tends to decrease, while a parameter with a substantial absolute value is more effective in distinguishing between different classes. While the boundary may appear more distinct in the input space, there is also a notable increase in the overall error accumulation. Moreover, during the training phase of classification tasks, especially involving a **softmax** layer, it is common for a typical optimiser to implement significant parameter updates, particularly in the later layers, posing a potential risk.

In the instances we have previously addressed, the performance of the system in relation to its accuracy, loss, and resilience to variation in parameters or activation functions is specific to the current configuration. It is crucial to take



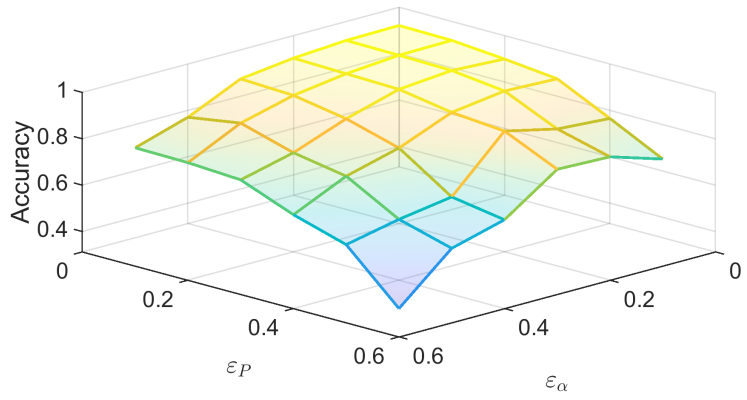
(a) (b)

Figure 6.3: The output of non-ideal case neural network and its difference from ideal-case output seen at hidden layer neurons of the same network and configuration as shown in Fig. 6.2. (a) The output and difference seen when all parameters are scaled by a factor of 0.5. The upper row shows the output of one neuron of each hidden layer. The lower row shown the corresponding difference between ideal case. The second layer (right-hand side) has a vague separation between classes while with medium level compared with ideal case outputs. (b) The output and difference seen when all parameters are scaled by a factor of 2. The upper row shows the output of one neuron of each hidden layer. The lower row shown the corresponding difference between ideal case. The second layer (right-hand side). A clear perturbation is observed at both neurons within the input space while it can clearly separate the input space into four parts as designed. The neuron in the first hidden layer (left-hand side) also exhibits slightly recognisable difference against ideal outputs.

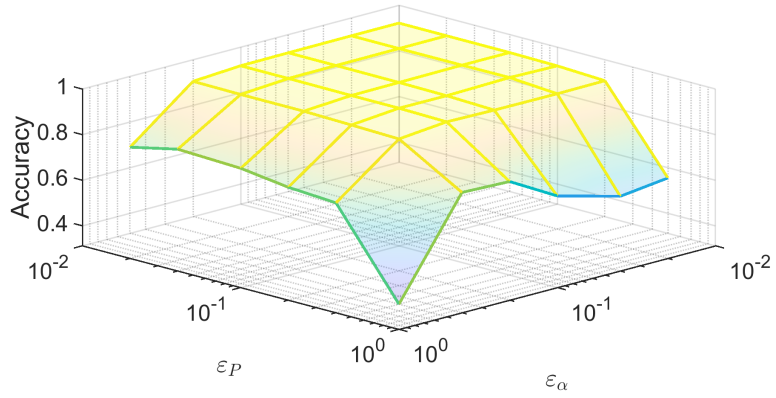
into account the surrounding sub-space of attainable average values within the hyper-space of the loss function near the local minimum achieved through training when evaluating the potential application of a neural network.

In a pragmatic case study utilising the Modified National Institute of Science and Technology (MNIST) dataset, we have developed a well-trained neural network model with dimensions of $[784 \times 28 \times 14 \times 10]$, as described in Section 4.5. To demonstrate the impact of parameter perturbation, we introduced variations into the parameters and activation functions simultaneously based on Eq. (4.20) and discussions in Section 5.5. The output depicted in Fig. 6.4 illustrates that these modifications have a nearly additive effect on the system's performance, with no significant decrease in accuracy observed during classification tasks conducted under normal distributed relative errors up to a standard deviation of 0.5.

In order to assess the robustness of a neural network at a system level, given that the network is represented by a function denoted as $f(\cdot)$, a common approach is to establish a maximum distance δ such that for all inputs $\forall x : \|x_0 - x\| \leq \delta \Rightarrow f(x_0) = f(x)$ [134]. In practical scenarios, such as hardware implementations, variations in parameters and activation functions can significantly impact the overall performance. These variations can be quantified as potential deviations denoted by ε_α, ε_n and ε_P. Therefore, it is necessary to consider all these factors collectively, leading to a reformulation of the criteria as $\forall \varepsilon_X, \varepsilon_\alpha, \varepsilon_n, \varepsilon_P : \|\varepsilon_X X\| \leq \delta, \|\varepsilon_\alpha\| \leq \delta, \|\varepsilon_n\| \leq \delta, \|\varepsilon_P P\| \leq \delta \Rightarrow$



(a)



(b)

Figure 6.4: The accuracy drop of a well-trained model for performing a Modified National Institute of Science and Technology dataset-based classification task when perturbations injected to the weighting-related parameters ε_P and activation functions ε_α. (a) Accuracy drop against perturbations, in linear scale. (b) Accuracy drop against perturbations, in logarithms scale. Within a reasonable and achievable tolerance range, where perturbation for both parameters in each layer is modeled as a Gaussian distribution with its standard deviation $\sigma = 0.1$, the accuracy maintains a higher-than-90% level, compared to the ideal case of around 92% for the model.

$f\left(\hat{x}, \hat{\alpha}\left(\cdot\right), \hat{P}\right) = f\left(x, \alpha\left(\cdot\right), P\right)$. For intricate systems, conducting such an analysis can incur substantial computational expenses, especially in a statistical manner with all data pieces available and for all possible inputs [234].

However, in a thorough analysis utilising Eq. (6.6) for multi-layer relative error assessment, we can establish a specific range within which the neural network's output remains unaffected by variations in input quality and network configuration during both the training and validating phases of a classification task.

When examining the output at the **logits** layers of networks in response to input stimuli, taking the classification based on the MNIST dataset shown in Fig. 5.15a as an example, discrete values are generated for each class as a "confidence score". The output is subsequently intended for processing by a **softmax**

or **max** layer, enabling the conversion of raw outputs into prediction confidence levels. This allows for a comparison between the generated output and the "target vector" as well as the "target" itself, facilitating the evaluation of loss and accuracy. From an information theory perspective, it is important to note that a substantial portion of the input data's information may be diminished as it traverses through the sequential layers, which introduces a certain degree of acceptable distortion within the system. Consequently, any discrepancies observed between accuracy and loss, in conjunction with the self-stabilizing dynamics of signal transmission and processing, suggest that the model's predictive capacity is unlikely to be adversely affected in scenarios where significant distortions are not present, thus maintaining stability in the relationships among accurately predicted instances.

The conclusions derived from this analysis can be expressed in a mathematical framework that facilitates the comparison of the maximum output value, represented as Z_{max} , against other output values denoted as Z. This comparison allows us to establish a criterion for the identification of potential inaccuracies within the system. Specifically, the integrity of the system's accuracy is called into question if the condition $Z_{max}\,(1-\varepsilon) \leq Z\,(1+\varepsilon)$ is not satisfied. It is essential to note that Z_{max} is presumed to always be a positive value. Furthermore, for levels of systematic perturbation ε that remain beneath a defined threshold, as determined by statistical methodology ($\varepsilon < \varepsilon_Z$), the system sustains robust performance across both training and validation samples.

In simulations using the Monte-Carlo method, it is common to observe that when the root mean square (RMS) values of perturbations smaller than the optimiser's step size, there is a possibility for the network to achieve an improved output compared to its trained state. This observation highlights the unpredictable nature of performance in relation to perturbations. Experimental results also indicate that reducing data precision (equivalent to introduce inaccuracy in some extents) can sometimes lead to overall performance enhancement [112]. It is essential to fine-tune neural networks implementations locally, despite achieving a well-trained model. In general, we can statistically observe a proportional increase in error rate with increasing levels of distortion allowing for a linear relation in cases of small perturbations.

With a comprehensive understanding of the potential sources and pathways by which errors can arise, spread, and accumulate within a neural network under consideration, it becomes feasible to anticipate the distribution of errors at each layer based on the specific implementation and parameters employed. Nonetheless, it is important to note that the presence of error terms does not always exhibit a direct positive relationship with the overall systematic performance of the system, as elucidated in the literature on explainability in this field [176].

6.3 Limitations and Challenges

While the efficiency of the AFC and MAC designs has been validated through calculations and simulations, it is important to note that the time-domain response differences could pose a challenge when integrating these modules directly into a system.

In Chapter 5, we have put forward two potential designs of MAC for a

HNN system, each with its own set of limitations and advantages in practical applications.

One of the proposed designs involving operational amplifiers (Op-Amps) allows for the construction of a cascaded system based on the topology and configuration of neural networks. However, the use of additional devices and stages for sign bit assignments may introduce distortion to the system in a same way as get non-ideally activated. Additionally, the latency of each linear follower or sample-and-hold (S/H) stage could potentially diminish the overall response quality of the design.

As evidenced by the large-scale fan-out capabilities required by the AFC as described in Section 4.4, the imposed current restriction on the sub-threshold mode circuit could result in limited effectiveness in driving multiple output loads, thereby causing a systematic delay. Conversely, the implementation of a weighted system utilising multiple switches may inadvertently create an undesired leakage pathway to the common ground **GND** or the opposing plate of the summation capacitor, necessitating frequent intervention in practice, as estimated in Chapter 5. Furthermore, the attenuation of signal strength and energy loss inherent in the use of exclusively passive devices within the layers necessitates the exploration of potential solutions through the integration of active devices for amplification and isolation.

In practical applications, additional amplifier-based networks can be utilised to conduct a S/H operation as a linear buffer in order to meet the response time needs of both circuits.

For the design incorporating Op-Amps and linear follower circuits as a signed summer, we will address the following subjects in the upcoming project:

1. Conducting an analysis and facilitating discussions regarding the time and energy consumption of the specified system through simulation
2. Executing simulations on clearly-defined tasks to evaluate the robustness of the system in relation to device tolerances arising from various levels of the neural network.
3. Developing conclusions on the practicality and limitations inherent to this particular implementation.

Furthermore, in reference to the H-bridge-based design with a charge pump-based accumulate circuit of the system, we intend to:

1. Investigate strategies for addressing discrepancies in frequency within weighting, summation, and activation circuits.
2. Develop a suitable circuit design for the sample-and-hold operation in practice.
3. Assess the incremental static and dynamic power consumption associated with the proposed S/H circuit introduced.

6.3.1 Typical limitations for Operational Amplifier based design in hardware neural networks

In considering the design of an Op-Amp-based MAC in HNN implementations, a key challenge that may arise is the issue of robustness, as well as the potential impact on overall time and energy efficiency within the system.

As illustrated in Fig. 6.5, the integration of each sub-module incorporated

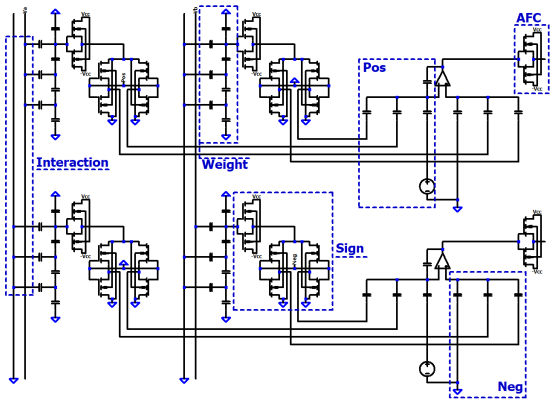


Figure 6.5: The schematic of one linear layer and one activation layer of the implementation of a $[2 \times 2]$ hardware neural network as proposed. The system is based on an operational amplifier design. The functional blocks **Interaction**, **Weight**, **Sign**, **Pos** and **Neg**, together with the Op-Amp connecting to **AFC** consist the multiply accumulate circuit. In the design, **Interaction** connects the MAC to the signal and ground bus. Module **Weight** and **Sign** follows the design of multiply circuit proposed in Chapter 5. The Op-Amp and the capacitor array **Pos** and **Neg** forms the accumulate circuit as proposed.

in the design has been streamlined to create a scalable cascade of circuit design. By employing a higher abstraction level, it is possible to extend the structure depicted to establish more intricate networks. The capacitors linking the Op-Amp in the **Pos** labelled box and the one linked to the **GND** in the **Neg** labelled box are referred to as summation capacitors. These capacitors have been specifically sized or scaled to a ratio of $1/r$ compared to the other capacitors in order to achieve an amplification of r times at the pre-activated value of the linear operation segment.

In this scenario, should we consider incorporating the pull-push linear follower structure outlined in the **Sign** module as an additional component of the activation function, we could then potentially reformulate the function $f(\cdot)$ described by this structure as follows:

$$\begin{aligned} f(x) &= \hat{\alpha}(\hat{p}\hat{x}) \\ &= \alpha(\mathbf{sign}(p)\,\alpha(|p|\,x)) \end{aligned} \tag{6.7}$$

In the equation mentioned above, the $\mathbf{sign}(\cdot)$ function is represented by a modified version of the linear follower, with a signal on its common **Gate** capable of driving it to the off-state for either of the transistors. In this scenario, it can be assumed that the conductance of the transistor in the on-state is significantly higher, leading to a slight adjustment to the weighting factor p due to the non-ideality of the linear following nature and the voltage dividing behaviour of the pair of subsequent transistor pairs. However, the variable $\hat{x}$ is significantly affected by the replication of the transfer function of the follower. As is shown in Eq. (5.4), the weight generated by the **Weight** circuit is consistently below 1, and the gradient of the activation function based on the transfer function also does not exceed 1, ensuring that the input of the **Sign** circuit always operates within a linear range of the device. Consequently, the output observed in the initial stage

of the circuit by the first follower can be expressed as

$$y = |p|\, x\,(1 + \varepsilon_\alpha) + \frac{\varepsilon_n}{\alpha\,(px)}. \tag{6.8}$$

Therefore, the actual input perceived by the **Pos** and **Neg** segments can be simplified as

$$\hat{p}\hat{x} = px\,(1 + \varepsilon_\alpha + \varepsilon_P) + \frac{\varepsilon_n}{\alpha\,(px)}. \tag{6.9}$$

Through adjustments, Eq. (6.6) can be reformulated as:

$$\varepsilon_Z = \frac{1-\theta_r^k}{1-\theta_r}\left(\varepsilon_\alpha + \frac{\varepsilon_n}{Z} + \theta_r\left(\varepsilon_\alpha + \varepsilon_P + \frac{\varepsilon_n}{Z}\right)\right) + \theta_r^n \varepsilon_X. \tag{6.10}$$

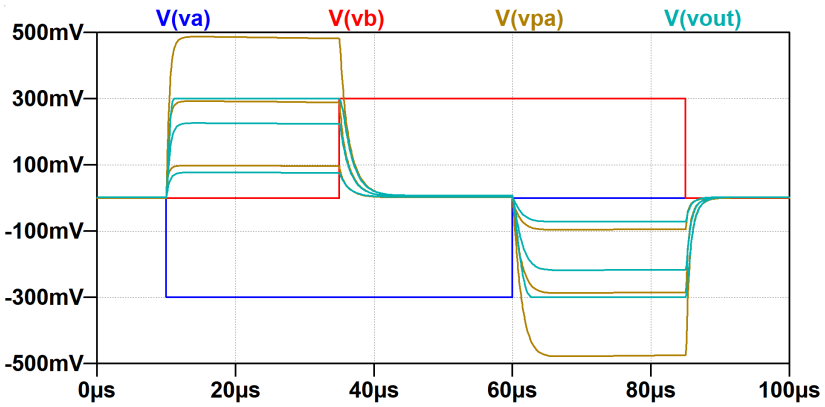


Figure 6.6: The response of a single layer perceptron receiving two pulse inputs **Va** and **Vb**. Their sum seen before the activation function circuit **Vpa** and the activated value of the AFC **Vout** are presented. The curves are tested with an amplification of $1\times$, $3\times$ and $5\times$ at the pre-activated value. Time response is nearly constant at $10\ \mu s$. A clear linear relationship between the inputs **Va** and **Vb**, the pre-activated values **Vpa** and the activated value **Vout** can be seen from the plots.

Regarding the timing delay of the system, it is evident from Fig. 6.6 that a nearly consistently uniform charging time of $10\ \mu s$ is necessary for the system when utilising $1\ fF$ capacitor in all components except for the two summation capacitors connecting the Op-Amp and from the positive port of the Op-Amp to the ground. In the simulation displayed in Fig. 6.6, scaling ratios $1/r$ of 1, $\frac{1}{3}$ and $\frac{1}{5}$ are implemented to demonstrate the system's behaviour in a perceptron model. When employing a smaller scaling ratio, the pre-activated value can be adjusted to 1, 3 or 5 times the original value. It is apparent from the results that there is a linear scaling factor for scenarios where the scaling factor is either $r = 1$ and $r = 3$. However, when the scaling factor is set at $r = 5$, the scaled pre-activation value surpasses the linear range and becomes saturated at approximately $\pm 0.3\ V$.

A scaling ratio of $\frac{1}{5}$ is utilised for the summation capacitors in the scenario depicted in Fig. 6.7 for all four neurons. The system has been tailored in accordance with the experiment specifications as applied in Fig. 6.2 to carry out an **XOR** operation on the two inputs. The experiment employs a $[2 \times 2 \times 1]$ network, with analytically assigned parameters and an activation function gen-

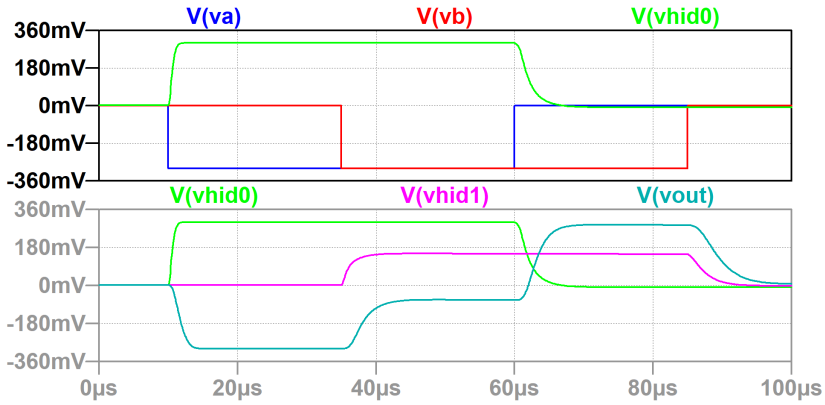


Figure 6.7: The response of the first two layers of cascading in the hardware neural networks (HNNs) performing **XOR** operation of two pulse inputs **Va** and **Vb**. The activated value seen at the two neurons in the first hidden layer are labelled as **Vhid1** and **Vhid2**, and the activated value seen at one of the two neurons in the second hidden layer is labelled as **Vout**. The delay between two layers are difficult to notice. A clear linear relationship between the inputs **Va** and **Vb** and the pre-activated values **Vpa**, activated values **Vhid0**, **Vhid1** and **Vout** can be seen from the plots.

erated by the transfer function of the AFC. The lower sub-plot reveals a similar response time for both layers, suggesting that the propagation delay may not be purely additive and have a significant impact on the overall time response of the network. From our estimation, the overall responding time from receiving the pulse signal to reaching a steady state at the output is proportional to the time constant of one stage, rather than the overall responding time charging the capacitors employed in weighting circuits to "fully-charged" state (which is much larger than the time constant). Should the assertion be substantiated through additional practical experiments of the HNN, a deeper and narrower, rather than shallower yet wider network would be favoured for the design implementation, to reduce the overall time delay of the system.

Each component will operate with minimal static energy consumption in the *pico-Watt* range as described in Chapter 4, or will only exhibit transient current flow during dynamic states. To optimise energy efficiency, the selection of the amplifier utilised will be determined by its efficiency and linearity at the operational frequency, which in this scenario is of a *mega-Hertz* level to align with the responding frequency of the MAC.

6.3.2 Compromise for H-bridge design in hardware neural networks

For the incorporation of a HNN comprising of AFC and MAC units utilising an H-bridge and charge pump design as presented in Chapters 4 and 5, a trade-off between operational frequency and potential failure due to leakage and the leading degrade of signal level needs to be addressed. As illustrated in Fig. 4.12, it is evident that increasing the operational frequency of the AFC will result in a decreased fan-out capability, thereby restricting the system's capacity to deploy a large-scale network in real-world scenarios. Furthermore, as out-

lined in Section 5.4, the non-ideal off-state resistance of the pass-gates within the accumulation stage, necessitating four switches for both the H-bridge and charge pump, introduces signal leakage that constrains the minimum frequency at which the pre-activated value remains stable, as illustrated in related diagrams and discussions.

Given the assumption that the equivalent capacitance of the weighting system remains a constant $\frac{2}{3}\ fF$ and the on and off state impedance of the pass-gates are $1\ k\Omega$ and $1\ M\Omega$ respectively, it can be inferred from the data and discussions in Section 5.3 that the leakage path from one plate of the capacitor to its opposite plate or the **GND** node will be approximately $3\ M\Omega$. This indicates that the time needed for one plate to be discharged by 63% will only be $2\ \mu s$. Moreover, in order to achieve less than 1% leakage a $200\ ns$ response time represents the upper limit, which is significantly faster than the operational frequency at which a single-load AFC can achieve a stable activated output. Furthermore, enhancing the ratio of resistance between the on- and off-states of the pass-gates may still present a viable solution. Nevertheless, as channel lengths decrease and off-state resistance diminishes, there remains a growing demand for increased operational speed.

Henceforth, it is recommended to utilise each module individually for various applications, as introduced in this study. In situations necessitating the concurrent application of both modules as linear and non-linear look-up tables (LUTs) in potential neural network implementations, a S/H circuit will be essential. In this particular scenario involving amplification tasks with passive components, a critical issue to consider is the signal strength and its robustness against leakage and other forms of distortion. For networks with a large number of layers, there is a possibility of experiencing exponential decay at the output end with purely passive components as demonstrated in the relevant design, necessitating a certain level of amplification to prevent degradation from noise interference. As detailed in the systematic analysis on performance and robustness in Section 6.2, it is acknowledged that while both summation and amplification operations are generally regarded as linear processes, non-linear factors can affect each operation due to voltage limitations and inherent non-linearity of devices employed, which may be underestimated in second and higher-order terms as described by Taylor expansion of the transfer function. Therefore, it is imperative to recognise the importance of operating within the linear region of the transfer function provided by the devices, while also allowing for adequate compensation for non-linear factors. This approach is essential to minimise adverse effects on the output of the HNN to be implemented. For simplicity and stability, it is advisable to utilise a traditional Op-Amp-based summation circuit in the implementation of large-scale networks.

An alternative approach that could be considered involves implementing a S/H circuit at the output side of the accumulate circuit discussed, as depicted in Fig. 6.8. This circuit has been designed to be devoid of resistive components or any current leakage paths, thereby minimising static power consumption. The control signal for this functional block can be directly derived from the **APPLY** signal introduced in a previous Section 5.3. To ensure optimal performance, a two-Op-Amp structure is recommended for the circuit that effectively eliminate potential leakage paths. Additionally, the possibility of integrating the AFC into

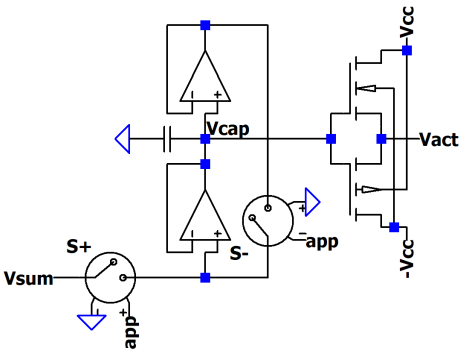


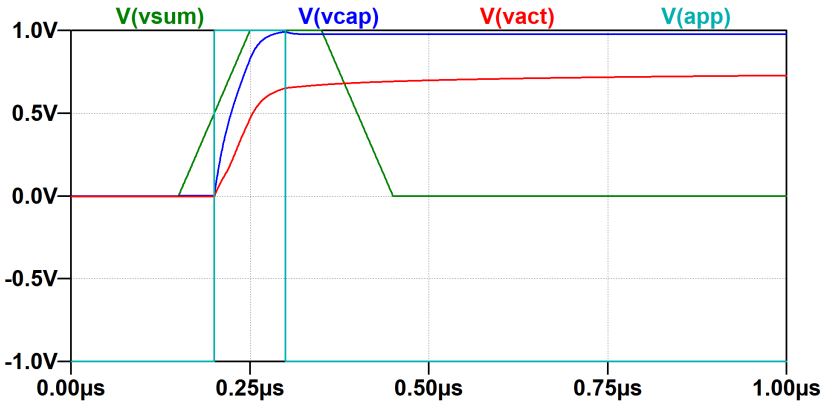
Figure 6.8: The schematic of the proposed sample-and-hold circuit, with its output signal **Vcap** connected to the input of an activation function circuit. Two unity-gain amplifiers are applied to produce a two-stage feedback loop maintaining the sampled signal to be a constant. The two switches are controlled by signal **App** and its complement, which is also the signal involved in the summation circuit. Summed signal is inputted via **Vsum**

the S/H system should be explored. This integration would require recalculating the activation function based on a combination of with the transfer functions of the amplifier and the AFC involved as the updated function to be applied in the HNN.

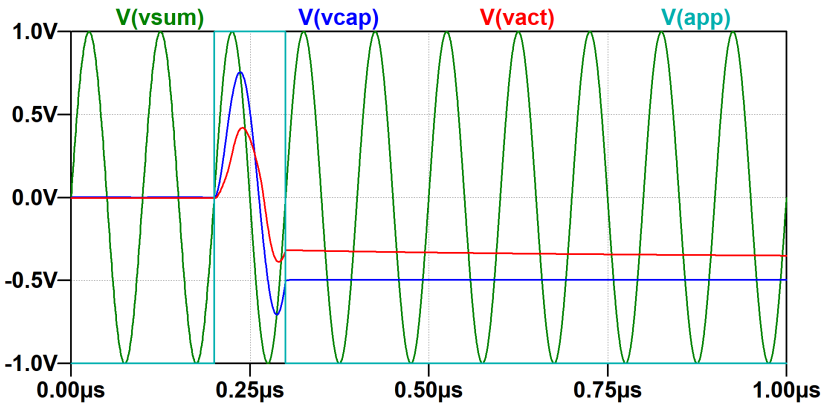
Utilizing a S/H circuit can assist in achieving the necessary frequency parameters for both sub-modules, albeit at a sacrifice of efficiency and resilience, as depicted in Fig. 6.9. It is demonstrated in Fig. 6.9b that with a norminal $1\ pF$ S/H capacitor, we can adequately sample a $10\ MHz$ signal and maintain it for a period of at least a *micro-second* without notable decay (not depicted in the figures). This capability ensures the scalability of the AFC for implementations of neural networks with wider layers.

However, it is noted that the trade-off for combining these components is a decrease in energy efficiency and an increase in inaccuracies. Upon examining the two-stage amplifiers utilised in this configuration, it can be observed from Fig. 6.9 that there is already a phase shift of $\frac{\pi}{6}$ and a reduction in amplitude of approximately 25%. While the decay and potential phase response to various frequencies can be considered as part of the activation function's behaviour, it is evident that the overall signal strength diminishes across layers. To effectively accommodate a deep network, additional amplification weights will be necessary. Ultimately, the convergence may lead to a less stable system with higher levels of error ε_α, ε_n and $\frac{PX}{Z}$.

On the contrary, with regard to energy usage, each calculation and activation process necessitates a charging and discharging procedure. Despite the fact that dynamic energy consumption increases proportionally with the number of neurons, it can still become a significant expense for larger neural networks. Additionally, there is a requirement to operate a pair of Op-Amps in the S/H circuit, charging hold via a capacitor which consumes no power under the control of a pair of switches, along with a potential additional amplifier at the output stage of the AFC. While it may be necessary to incorporate an extra amplifier between the S/H capacitor and the AFC, in practical terms, due to potential leakage issues



(a)



(b)

Figure 6.9: The time domain response of the sample-and-hold circuit. (a) The response to a soft step-form input. (b) The response to a sine wave input. The waveform of a reference voltage signal **Vsum** representing the input at the final stage of summation circuit, the sampled signal seen at the capacitor **Vcap** and the activated value of the activation function circuit **Vact**. The controlling signal **App** is also shown as a reference. Phase shift and decay in amplitude at both **Vcap** and **Vact** can be observed by non-ideality of components.

that could arise from adjustments made to the system, this option may not be as appealing.

6.4 Conclusion

It can be inferred that in a series of cascaded neural networks, the final error in relative terms will be a result of the cumulative impact of parameter perturbations, activation functions and noises in signals within the system. However, currently there is no definitive link between systematic errors and the accuracy of predicting behaviour of a well-trained neural network. However, there is an observed tendency suggesting that the distortion in parameters and signals may be positively associated with the loss seen at the output. In simpler terms, for regression scenarios where accuracy is directly tied to the final output, it is feasible to deduce or estimate a reduction in performance with less computational resources by analysing the factor $\theta \frac{PX}{\alpha(PX)}$ by the behaviour of activation function

applied and feed to the model shown by Eq. (6.6). This can lead to an overall improvement in efficiency to create a robust neural network model. On the other hand, for classification tasks, the global Lipschitz constant-based method can be employed to rigorously assess situations where accuracy is required to remain constant.

Through the outcome gathered with the experiments, it is possible to further corroborate the validity of the formula we have derived regarding neuron outputs. Nevertheless, it remains challenging to accurately predict the extent to which errors in layer outputs may impact final inaccuracies. In our examination using the MNIST dataset, a discernible proportional relationship between final stage distortion and inaccuracies was observed. This suggests that by perturbing each individual functional block within a certain threshold, we can still attain a satisfactory output, approaching less-than-1% accuracy drop compared to the theoretical optimal scenario. It is important to note, however, that this conclusion is specific to the minimum threshold we have identified and has not been statistically validated across other potential local minima.

Upon analysis of the two potential combinations, it has been determined that the Op-Amp-based design could potentially align well with the AFC with minimal time delay. However, it is important to note that energy consumption, while increasing linearly with the number of neurons, should not be underestimated. On the other hand, the charge pump and H-bridge offer the advantages of significantly lower energy consumption and circuit footprint, and are more suitable for use at higher frequencies. However, they may not be suitable for low-frequency conditions without high-impedance switches or S/H devices. As a result, it may not be advisable to directly integrate them as a functional block with the proposed AFC proposed. Even with the use of S/H circuits, there may be a decrease in energy and time efficiency compared with our aforementioned theoretical estimation.

Chapter 7

Potential Adjustments and Applications

Despite the advantages of hardware neural network (HNN) compared to its corresponding software implementations in terms of energy, time and space efficiency, it is important to acknowledge the operating limits due to its inherently lower precision. The inherent challenges associated with the back-propagation algorithm, particularly in systems experiencing variations in parameter representation due to production processes, necessitate the optimisation or redesign of the optimiser algorithm to ensure the stability of local optimal solutions. This section offers a general review of the current research on optimisation algorithms that involve discrete features for HNNs and its applications, the use of local search algorithms in this domain, and the potential of Hebbian method algorithms in this context. Furthermore, a brief assessment of the feasibility of implementing such algorithms on hardware for on-chip training is provided. Finally, we present examples of practical application and potential adjustments in the topology of HNNs for future research, and specific case studies that can serve as valuable benchmarks for hardware design in both industry and academia.

7.1 Overview

As a versatile tool, neural networks have been used in various industries. However, when considering highly specialised hardware neural networks (HNNs), the primary challenge lies in the accuracy trade-offs against efficiency. This can make it unsuitable for applications in precision-driven or high-performance environments. On the other hand, in scenarios where energy and space constraints are a concern, particularly when rapid responsiveness is essential, HNN can be a favourable option. To fully unlock the capabilities of this system, it is essential to identify applications where it can excel and devise algorithms to help alleviate some of its inherent limitations.

The design of HNNs has shown significant variability in numerous studies in recent decades. Typically HNNs aim to provide efficient solutions for tackling complex and poorly defined problems as other forms of neural network implementations, with a reduction of latency in operations and a boost in efficiency of power consumption and system footprint. From our estimation and simulation, the proposed design of the HNN may lead to a reduction in operational latency

and a decrease in the number of computing devices required to perform equivalent tasks compared with the state-of-the-art (SOTA) works. However, it is important to note that this efficiency often comes at the expense of precision, with lower resolution parameters and increased noise within the system during operation. Thus, advanced optimisation algorithms are required for the low-precision and noisy case while insufficient knowledge of activation functions is provided.

Recent studies have indicated the potential for maintaining output accuracy in lower-resolution implementations through additional optimisation and pruning techniques. There is awareness that the error observed in inputs and the implementation of each parameter of the system can be modelled as perturbations of a given mathematical model, and we may also find works suggesting how errors can propagate along the system and affect the final-stage outputs. However, there are few works highlighting how distortions seen in the output stage are related to the final accuracy of a given neural networks.

Hence, this particular field holds significant appeal for developing an algorithm for fine-tuning or training schemes to update network configuration involving its topology and parameters. However, training analog systems can be challenging due to device tolerance and a lack of awareness of the said inaccuracies in the computation and transmission of signals [37]. Optimisation efforts, particularly those occurring off-chip, may encounter issues related to inaccuracies in both the multiply accumulate circuit (MAC) and activation function circuit (AFC) components.

Initially, it can be challenging to dynamically obtain the inverse of the original value of a weighting parameter, which serve as the key point of the gradient descent algorithm, for a real case in analog systems, even with full control over its connection. As mentioned previously, the parameters of the neural network implemented by hardware can be distorted in various ways and may deviate significantly from the desired value (*e.g.* the 5-bit weighting system with 25% relative error illustrated in Fig. 5.12). Even though it is conceivable that a precise implementation of parameters might be achieved in a close-to-ideal circumstances, the rounding errors of the low-resolution system may still result in an imprecise response to adjustment signals.

Furthermore, when it comes to the generated activation functions by analog circuits, it is important to take into consideration the potential mismatches and technology scalability as discussed in Chapter 6. It is noted that the gradient of the proposed activation function derived from a mathematical model of a single AFC may not align with the actual derivative seen in the transfer function of the implemented AFC at the same point of pre-activation input. We have not been able to find works or approaches to identify any definitive information regarding the potential impact on the training process when dealing with the inconsistence between the activation function and the derivatives applied in the training processes.

In this part, we will address certain areas of interest that we have begun exploring but have not yet obtained satisfactory results concerning decision-making accuracy and the generalisability of findings across various tasks or topological structures. In addition, we will propose some experiments and technological route for future exploration according to the limitation of the HNN proposed.

7.2 Gradient-Free Robust Optimisation

HNNs, identical to cases with MACs constructed with resistive crossbar designs, share a common characteristic of presenting tunable yet stochastic operations in practical cases. The sensitivities and potential limitations on the number of states available in non-ideal hardware implementation are observed. The training of such a set of designs will require algorithms to be capable to drive the weight updating scheme in a numerical probabilistic method with local gradients involved. In such case, the gradient of activation functions used in the algorithm and the parameters seen at the mathematical expression of the neural network may not be consistent with the transfer functions of the AFCs together with the amplification ratio of the MAC. Compared with the analytical method for classical gradient descent-based optimisers, the methods to be proposed are commonly expected to be less sensitive to hardware non-idealities [235].

Based on the discussion on the proposed HNN, taking into account non-ideal circumstances, it can be inferred that utilising a direct method for calculating the final stage loss and back-propagation in traditional algorithms based on gradient descent could pose risks to the convergence of the optimiser. Therefore, we are in search of methods to optimise a network without direct access to the actual magnitude of the parameters or the analytical model of the behaviours of the layers.

It is vital to validate and deploy the HNN as neural chips within real-world optimisation processes to ensure enhanced integration with the sensor and control systems of modern electrical systems. To accomplish this, it is pertinent to explore interface design, the practical application of the system, and how algorithmic and hardware implementations can be integrated.

We are optimistic that this preliminary phase, encompassing the design of the circuit and the development of the optimisation algorithm, will establish a foundational basis for constructing a system of suitable scale that can be deployed in practical applications. Additionally, the integration of training and refinement technologies will be essential to facilitate the mass production of the neural chips detailed in our proposal, along with other SOTA designs referenced in the preceding chapters of this document.

7.2.1 Hardware orientated optimiser

When representing parameter values with hardware components that experience random perturbations within analog devices' tolerance, adjustments are fixed by the selection of devices after calibration., the use of optimisers relying on derivatives may not be appropriate. As demonstrated in Tab. 2.2 and Fig. 5.13, traditional analog MAC devices may only offer a limited range of parameters with stochastic deviations from their intended values. Consequently, systems based on gradient descent may not always be effective in this context.

According to the errors identified on the output of a linear classifier, a weight update algorithm known as the "minimal disturbance principle" is proposed in the last century [147]. In the algorithm, we need to consider the input vector denoted as X, the error δ in its output, the learning rate η, and the weight update formula for each iteration given by $\Delta W = \eta \frac{\delta X}{|X|^2}$. In cases where a non-

linear system (specifically a balanced binary system whose range is $\{-1, 1\}$ in this context) is present, the updating process will be adjusted to $\Delta W = \eta \frac{\delta X}{2|X|^2}$. Additionally, in cases where the pre-activation input Y is less than or equal to a specified constraint γ ($|Y| \leq \gamma$), the term $\frac{\eta}{2}$ can be adjusted by a fixed term d based on a derived concept.

On the other hand, there are efforts being made to approximate the partial gradient term $\frac{\partial \delta}{\partial w}$ by using the zero-order approximation $\frac{\delta(w+\Delta w)-\delta(w)}{\Delta w}$ [148]. The algorithm eliminates the need for backward transmission of the weighting information of subsequent layers to the former. The algorithm has demonstrated the ability to effectively perform classification tasks using a hardware activation function similar to a *Sigmoid* function. In their work, a slight adjustment rate of $\Delta w = 0.001$ is suggested, which still exceeds the level of accuracy achievable with our current design if not applying cutting-edge devices and technologies. In a similar fashion, drawing inspiration from optimisers that incorporate momentum, the momentum term m is quantised as $m' = 2^{\lfloor \log_2 m \rfloor}$ [236]. Results from the study show that for basic regression and classification tasks, the quantization process of the system does not show any discernible impact on performance compared to popular optimisers such as Momentum and Adam.

It has been demonstrated that using a fixed randomised matrix of equal size for the back-propagation path can result in increased efficiency during the training process. This presents a potential route for implementing the Hebbian learning rule [237, 238]. This approach, known as "feedback alignment", is described by the equation $\Delta W \propto BeX^T$, where B is a fixed random matrix, e represents the error term calculated by $T - Z$, and X^T denotes the transpose of the input vector X. T represents the target and Z signifies the output of the neural network.

The examples suggest that, for optimisation purposes, the precision of backward signal accuracy and updating step sizes may not significantly impact the ultimate learning outcome. An analog HNN could be successfully trained on-chip with satisfactory results tailored to specific applications as shown in the case studies demonstrated in literature. However, despite advancements in reducing operation power consumption, the precision requirements outlined in the aforementioned studies continue to pose challenges.

A potential method for parameter acquisition under finite resolution and stochastic perturbations is meta-heuristics, such as the "simulated annealing algorithm" [239]. While it may converge slowly, the simplicity of the algorithm makes it suitable for lower computational power systems [240]. Variants like Creutz's "micro-canonical annealing" [241] and Dueck's "threshold accepting method" [242] effectively tackle combinatorial optimisation problems. Other local search strategies, such as Charon et al.'s technique [243] and Glover's "Tabu search" [244], can also be useful. When multiple devices are available, population-based meta-heuristics can be applied. However, the effectiveness of search algorithms is limited for small neural networks due to efficiency concerns [245].

The main challenge in implementing search algorithms without gradient involvement is the massive neurons and the connections between them. Research on estimating the importance of specific synapses before training concludes is limited, making pruning during training impractical. A viable alterna-

tive is to create multiple pre-trained sub-networks on a smaller scale, inspired by the "capsule neural network" concept by Sabour et al. [246]. Although there is little empirical evidence for its universal application, this approach could facilitate incremental learning and improve task generalisation. Additionally, training the neural networks off-chip with gradient descent-based optimisers, using an approximation of the derivative of the activation function, followed by fine-tuning with the methods discussed, is a potential approach recommended.

In addition to the algorithmic functions focused on parameter adjustment through explicit functions, there is also research in the realm of physical learning. This field views natural systems as analog computing devices and typically suggests *a posterori* approach to explaining the self-modulation process [247]. Although the system we have put forth bears resemblance to an analog computer, the lack of clarity in the specifications of the training progress mentioned in this work makes it challenging for the system to be trained accordingly in real-world scenarios effectively. As a result, designing the proposed system for enhanced learning capabilities presents significant complexities.

From the literature as cited in this section, the research on optimisation neural networks has illustrated that there is no explicit necessity to apply a gradient-dependent algorithm or knowing the exact magnitude of the parameters involved to search for a local minima of the loss surface. The conclusion hints that for analog systems, there do exist potential optimisation algorithms that can work in the "black-box" system. Besides, as can be learnt, the major drawback of the searching algorithms is the effieiency to search through the high-dimension parameter space consisted of the large number of parameters involved in neural network, the philosophy of divide and conquer may illustrate a feasible starting point. In our research, , the integration of the previously mentioned algorithms in this section has produced initial findings in basic classification and fitting tasks. Nevertheless, in light of the limitations observed in our learning outcomes when contrasted with SOTA methodologies, as well as the challenges associated with the generalisation of the algorithm, we assert that additional research is warranted in this area.

The primary challenge in this aspect pertains to the ability to align the algorithms with hardware implementations. Even though the approximations mentioned in this part are tailored for the systems they examined, they may not always lead to accurate convergence in real-world scenarios, especially when the proposed system is highly analog-based. Although our weighting system has been designed with a high level of discretization to minimize the risk of error accumulation, there is no certainty that the suggested step direction aligns with the actual parameter shifts, as depicted in Fig. 5.12. Furthermore, the computational and storage demands of the search algorithms may render direct implementations impractical.

7.2.2 Biology inspired optimiser

It has been demonstrated that hardware implementations of neural networks may not achieve the same level of precision as a software model with high-resolution parameters [19]. Nevertheless, the generalisation ability observed in biological nerve systems makes it more convincing that complex tasks can be

accomplished by a slower and fuzzier yet a more efficient structure. If the assumption that a mathematical model of neural network accurately represents the behaviour of biological neurons is valid, there may be heuristic algorithms that can overlook inaccuracies or disturbances to facilitate more efficient learning in brain-like systems.

Additionally, as demonstrated in previous chapters, the activation function produced by the proposed circuit may vary depending on manufacturing and operational variables. This variability poses challenges in accurately predicting the performance of individual neurons in practical applications. Gradient-based optimisation techniques may struggle to reach optimal local minima due to discrepancies between ideal and non-ideal activation functions and their gradients.

To address the issue, we decided to seek guidance from the bio-inspired Hebbian algorithm. As observed by Hebb,

> "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased [248]."

The mathematical representation of this update is $w_{ij} = x_i x_j$, or more generally, $w_{ij} = \langle x_i x_j \rangle$ when considering multiple instances from a specific set of input-output pairs. Here, The terms x_i, and x_j represent the outputs of neurons i and j respectively, with their connection strength denoted by w_{ij}. The function $\langle \cdot \rangle$ signifies the average of its inputs. Alternatively, $\Delta w_{ij} = \eta x_i x_j$, where η symbolizes the learning rate of the system. If external targets t are available, it is feasible to modify the output term of the dependent variable (or neuron in the latter layer) by the disparity between its output and the target. In this scenario, the formula can be displayed as $\Delta w_{ij} = x_i \left(t_j - x_j\right)$, under the assumption that neuron j is reliant on (or connected from) neuron i [249]. Additionally, it has been observed that the performance of this algorithm aligns with traditional gradient descent based algorithms when the loss function is characterized by mean square error (MSE) $\delta \left(z, t\right) = \left(t - z\right)^2$, where z represents the output and t denotes the target for the specific example [249, 250].

The weight updating algorithm, as outlined in its expression, does not necessitate detailed knowledge regarding the characteristics, behaviour, or attributes of the parameters or activation functions. This feature contributes to the widespread use and acceptance of the weighting update algorithm within the realm of spiking neural networks (SNNs) as noted in literature [251].

The results obtained from our testing of single-layer perceptrons have demonstrated an acceptable level of performance, with an accuracy of over 85% accuracy on the Modified National Institute of Science and Technology (MNIST) dataset. As a reference, with gradient descent algorithms, the prediction accuracy of the trained network may also reach a similar accuracy seen at the validation process to be roughly 85%. The heatmap illustrating the parameters of the neural network trained using the Hebbian method effectively demonstrates a weighting that more accurately reflects the pattern of the input characteristics. Nevertheless, some challenges must be addressed to effectively integrate the algorithm into the realm of deep learning.

When implementing the technique on deep neural networks (DNNs), the adjustment of connections within hidden layers continues to rely on the gradi-

ents of neurons. The formula commonly used for updating hidden connections is $\Delta w_{ij} = \eta x_i \left(t_j - x_j\right) \frac{\partial}{\partial w_{ij}} f(w_{ij})$, where $f\left(\cdot\right)$ represents the network function [249,250]. In the absence of an alternate algorithm facilitating the transmission of error information to each parameter without the use of gradient descent, the system may still encounter issues related to inconsistent gradients as previously mentioned in the reason for not applying gradient descent related optimisers.

It has been observed that in situations where external targeting information is not readily available for supervision, the algorithm tends to reinforce the initial bias or impression of the model at the onset of training, irrespective of its accuracy. The finding also gets confirmed according to the observation of human learning activities [252]. From a biological standpoint, there are also concerns regarding the lack of a definitive equivalent of loss functions for humans (and other cognitively capable animals). Furthermore, the bi-directional structure enabling back-propagation has yet to be discovered [253,254]. The requirement for each neuron to have explicit information about weighting the former ones and the absence of randomness in data transmission or translation is also non-physical in mammal brains.

Based on the aforementioned discussion of potential failure of optimisation algorithms based on gradient descent, it is imperative to consider making appropriate modifications to the weighting update scheme. Multiple methods have been suggested for deep learning utilising the Hebbian method. In a dynamic system where there is a direct feedback matrix denoted as $W_{l+1,l}$ connecting from layer $l+1$ back to layer l, which is the transpose of the reversed directional matrix ($W_{l+1,l} = W_{l,l+1}^T$), there is the formation of a temporal-error model according to O'Reilly et al. [255]. The dynamic behaviour of the system can be described by the equation

$$\dot{X_l} = W_{l-1,l} X_{l-1} + W_{l+1,l} X_{l+1} - X_l, \tag{7.1}$$

and the updating formula is given by

$$\Delta W_{l,l+1} \propto \underbrace{T X_l^T}_{\text{Hebb}|_T} - \underbrace{X_{l+1} X_l^T}_{\text{Hebb}|_{\neg T}} \tag{7.2}$$

in a steady-state condition as $\dot{X_l} \to 0$ [254]. For previous layers, it will be necessary to adjust the target information T in the successor layer to $T_l = W_{l+1,l} T_{l+1}$ from the previous layer. A practical application involving hardware, specifically a memristive crossbar, follows the same principle as the specification of the HNN. It achieves faster and significantly more energy-efficient learning performance, resulting in a 95% accuracy in recognising Braille words training process based on central processing unit (CPU) or graphics processing unit (GPU) [235].

In the context of improved separation of forward prediction and backward optimisation, together with the aforementioned dynamics, it is possible to establish a "predictive coding model". During the predictive phase, the formula $X_{l+1} = W_l X_l$ is utilised, while in the learning phase, $\delta_l = W_{l+1}^T \delta_{l+1}$ is employed. The behaviour of the signals in each layer X and the difference δ is

represented by $\dot{X_l} = -\delta_l + W_{l,l+1}^T \delta_{l+1}$ and $\delta_l = X_l - W_{l-1,l} X_{l-1}$ [256]. By substituting X_l for δ_l, the resulting dynamic can be expressed as

$$\dot{X_l} = W_{l-1,l} X_{l-1} - X_l + W_{l,l+1}^T X_{l+1} - W_{l,l+1}^T W_{l,l+1} X_l, \tag{7.3}$$

leading to the formation of a "dendritic error model" [257, 258]. The learning formula still satisfies $\Delta W_{l,l+1} \propto \delta_{l+1} X_l$. This approach can be viewed as an approximation of an error propagation algorithm. The algorithms referenced utilise simplified linear activation functions, and it is important to make appropriate adjustments for real-world applications. Additionally, data pre-processing is necessary for input-output pairs that do not conform to the specifications of our proposed system.

The corresponding updating algorithms using the temporal error model, predictive coding model, and dendritic error model have been analysed in the course of our research. In the classification task based on the MNIST dataset, the employed algorithms demonstrate a markedly reduced time requirement for the optimisation programme to achieve convergence. It is important to note that the learning outcome does not consistently reach an optimal state, and is not as favorable when compared to those achieved by networks utilising gradient descent algorithms. Additional analysis is necessary to enhance the programme established in our study and evaluate its effectiveness. Furthermore, the algorithms must be refined to align with the discretised nature of the suggested MAC, requiring a fusion of hardware-focused algorithms.

In order to establish a training system for the analog neural network-on-chip, it is advised to start with Hebbian or similar methodologies. The process of adjusting weighting factors by non-integer increments can be translated into a series of probability-based adjustments for the parameters. When this approach is paired with a sufficient number of iterations and examples, it can yield a statistically significant result based on our simulations. To streamline the training process, it can be segmented into smaller modules or sub-problems in *a priori* manner. Subsequently, each module can be trained individually before being consolidated, resembling the structure of the capsule neural network or mixture of experts (MoE) systems. Upon the completion of training, the implementation of local searching algorithms such as simulated annealing or Tabu search can be utilised to further optimise the system.

If these methods prove to be effective, it may be beneficial to investigate the development of hardware systems and circuits that are compatible with our design. This could involve the creation of gate-controlled systems utilising analog storage and comparison amplifiers to execute a Hebbian updating algorithm. Nevertheless, the incorporation of local searching algorithms into hardware poses a significant challenge.

Even if the system is not pursued in future research, it is recommended to consider developing stand-alone chips that can interact with our proposed system. These external systems should have the ability to read network configurations and write onto the proposed MAC. Without this functionality, manual tuning alone may not fully exploit or evaluate the potential to generalise these systems in practical verification processes or manufacturing. In summary, it is both practical and advantageous to implement algorithms as an alternative to

gradient descent for optimising the hardware-implemented network, even in the absence of direct access to the specific parameter configurations and the operational characteristics of the applied activation functions. This approach has the potential to contribute significantly to advancements within the computer science community.

7.3 Potential Applications

The utilisation of HNNs, while accurate to a certain extent, is still limited in its generalisation abilities due to precision and the lack of practical proofs of concepts. Previous chapters have demonstrated its potential for carrying out basic classification tasks, but there are concerns about its ability to handle regression tasks effectively in practical cases with non-ideal parameters and activation functions whose resolution will not be high enough. This raises questions about the system's practical limitations, prompting the need for case studies or a mapping of the HNN developed to specifications for applications to evaluate its performance using real-world experiments.

To achieve the primary goal of developing a HNN, it is crucial to evaluate the proposed hardware system through practical implementations rather than relying solely on software simulations. The verification process is limited by the computational capacity of simulation tools and the complexities of manual circuit design. Therefore, a straightforward and robust testing platform and benchmark are essential for effective validation.

Experimental cases should be small and simple to allow researchers manual control at the start. They should be clear and have specific objectives for researchers to analyse and validate the training scheme. This enables the identification of learning patterns and potential risks during training and tuning.

In the foregoing chapter, we analysed the performance of the HNN and identified robustness limitations due to systematic errors observed in simulations and calculations. To address these issues in practical applications, it is essential to impose constraints on the implementation of this system and the topology and configuration of the neural network based on the aforementioned model. We must also evaluate the operational complexities of implementing this technique and consider whether additional complexity is necessary during verification. This section will examine potential applications that could help tackle these challenges.

A verification-orientated application of this research involves developing complex logic gates using neural networks. The goal is to replicate logic gates or design complex Boolean circuits with multiple digital inputs and outputs using HNNs with up to three hidden layers. Another testing scenario could involve assessing input-output vector pairs from a randomly chosen function. The two experiments have been carried out during the research.

The tasks are well-defined, and results can be evaluated against established mathematical models. However, the former experiment may seem overly simplistic and the latter may lack practical context, causing confusion. Additionally, they do not highlight the importance of optimisation efforts. Therefore, further case studies are needed to evaluate the capabilities and limitations of HNNs in real-world applications.

In the subsequent part, we will present two potential areas, tiny machine learning (TinyML) and neural decision tree (NDT) of probable research interests along with a tangible example that can serve as the initial focus for each area.

7.3.1 Tiny machine learning

TinyML is a specialized branch of machine learning (ML) focused on deploying models to low-power edge devices, such as micro-controllers or microprocessors [259]. The applications of the system enable devices to efficiently solve complex problems using algorithms identical to variants of neural networks while conserving time and energy. Additionally, it eliminates the need for the system to rely on accessing the global network [260]. The system is typically integrated with sensors and utilised in Internet of things (IoT) applications. The primary objective of such an application is to optimise time and energy usage while incorporating smaller and more cost-effective devices for deployment [261]. Our proposed system may offer efficiency in these areas, as previously demonstrated.

The potential applications of TinyMLs are diverse and include anomaly detection and control in industrial settings [262], smart agriculture by forecasting crop and weather conditions [263], or animal tracking and identification in environmental engineering [264]. The approach has also shown its potential in the fields of medicine and healthcare [265] or can function as operational components within smart wearable devices [266]. Among the various areas of application, the TinyML approach can operate independently with minimal power or energy requirements, making it suitable for integration into small-scale devices. One potential use case is in scenarios where constant connectivity to the Internet is not feasible (*e.g.* augmented reality (AR) devices) [267].

Based on a careful analysis of the strengths and weaknesses inherent in a typical TinyML system, we can accurately anticipate the specific criteria our proposed design must satisfy to emerge as a competitive alternative to existing implementations:

- Devices tailored for TinyML operations are designed to operate efficiently under energy constraints, typically at a *milli-Watt* level or lower [260,267]. In contrast to GPU-based devices consuming around 100 *Watts* of power, TinyML systems offer significant efficiency advantages, potentially allowing battery-powered functionality for devices incorporating TinyML features [261].
- The devices need to be cost-effective, making them suitable for widespread use in various regions [268]. These devices are designed to operate with minimal storage and computing capabilities, resulting in lower costs compared to traditional server-based systems [259, 269]. By eliminating the need for data transmission to a cloud platform, the system can also reduce bandwidth expenses [260].
- The potential time delays inherent in the interaction with cloud or fog computing devices can be mitigated by leveraging embedded systems to manage computationally intensive tasks [269]. This enables the provision of real-time user interfaces and prompt notifications for critical situations [260, 270]. The lack of network connectivity in both directions can

help safeguard sensitive data and enhance overall reliability [261, 269]. This aspect can address privacy concerns, especially in fields like business and decision-making systems [259].

However, the multitude of hardware platforms and software used for optimisation results in a lack of standardised testing and benchmarks across various systems. Determining the total energy consumption for each implementation presents difficulties due to the unique nature of the platforms and the computing power demanded for each model. The challenge of pursuing greater efficiency by using smaller memory and less powerful processing units hampers the completion of complex tasks [261, 267, 269].

Our design, featuring fully analog computing components, has demonstrated its capabilities in efficiently performing tasks related to detection or classification with minimal time and energy consumption utilising a limited number of elements that align with industry standards as noted previously. In the analysis conducted to determine the maximum operating frequency and energy consumption required, the design has demonstrated the capability to attain performance levels that are considered SOTA for functional blocks systematically as introduced. Additionally, our network design offers the advantage of maintaining quick response times without being hindered by the number of layers, unlike other designs that incorporate mixed-signal or digital implementations. Thus, it is convincing that it could meet the design specification and objective of the TinyML approach systems.

The inherent analog nature of the design also ensures the direct communication between sensors or crucial elements with transfer function connecting external physical or chemical reactions to electronic interfaces compared to the commonly used field-programmable gate array (FPGA) platform or other digital systems. This advantage also minimises the space needed for converters bridging analog and digital signals. When considering the conversion necessary for interacting with other devices as a multi-output classification or regression task, the system could form an interface between the elements and computing systems with customised functions including dimensionality reduction and fault detection. To conclude, the system we have implemented has the potential to serve as a reliable edge computing apparatus.

Among the various applications associated with the concept of TinyML, a notable area of research focuses on the interaction with sensor systems and their integration into IoT frameworks. This encompasses straightforward tasks such as comparing input patterns with pre-established models or compressing input dimensions to facilitate more efficient data transmission. Furthermore, it may be feasible to regard each network within an interconnected IoT framework as capsules within the aforementioned capsule network, thus improving decision-making and control capabilities. The applications typically necessitate neural networks of minimal size, and the operation of the implemented network must be both space and energy-efficient to ensure compatibility with sensor and data exchange devices. This approach aligns with the specifications of our proposed design, positioning it as a viable application opportunity.

The challenge identified in the realm of TinyML lies in the absence of a universally accepted standard and benchmark for assessing and predicting performance. However, this challenge does not act as a definitive obstacle to our

implementation efforts. As previously suggested, our criteria of HNNs for evaluation of their performances should be established based on a meticulously trained and finely-tuned implementation that delivers satisfactory performance aligned with the specific task requirements.

In light of the requirements outlined, it is necessary to prioritise efficiency in time, energy, and space utilisation over focussing solely on accuracy or model performance. Furthermore, it is essential to make necessary alterations to guarantee the scalability of our technology across various tasks, as stated in Chapter 6. The evaluation should be impartial and unbiased, taking into account overall performance rather than specific cases.

An example proposed for testing purposes in this field involves creating a refractometer without the use of a lens. In this scenario, we can determine the ratio of refractive indices of two media by observing the position of the image formed by a refracted incident beam at the interface. The receiver can be created using an array of photoresistors. This setup allows us to use the electronic signal output as inputs to create a composite function, or we can approach the problem by separately considering the non-polynomial transfer function of the photoresistor array and a ratio determined by a sine function. In our assessment, we may consider employing a divide-and-conquer philosophy in our design approach. Initially, it would be prudent to develop a network that translates the signals obtained from the photoresistor array into the angle of emergence. Subsequently, a separate network could facilitate the conversion of this angle into the refractive index ratio. Furthermore, the design could be optimised to allow for direct mapping between the signal array and the refractive ratio.

Regarding system communication, constructing potential dividers using linear resistors and photoresistors, while employing transistors for biasing and amplification, would provide a voltage-based interface. In scenarios where the angle of incidence remains uncertain, the input may be represented by the difference vector between readings taken with and without the presence of the medium under examination.

The benefits of this experiment include its clear definition and the ability to quickly assess results against predetermined targets. It is a relatively straightforward experiment to conduct, as all components are easily accessible. The experiment also involves a manageable number of variables, including the angle of the incident beam, the distance between the interface and the array, and the properties of the two media involved.

The experiment aims to enhance our understanding of the system's capacity to model a function through the assigning, training and fine-tuning of parameters. Additionally, it will facilitate the measurement of power consumption in real-world scenarios. The results of the experiment will demonstrate the viability of integration into a smart sensor design, a foundational component of the IoT system. The system's capacity to effectively manage non-linear input responses and mitigate noise interference serves as critical proof of its potential utility [271].

7.3.2 Neuron decision tree and mixture of experts

As evidenced in our previous chapters, even minor inaccuracies in individual components can rapidly magnify throughout the system. A significant contributor to the accumulation of errors is articulated as $\theta_r = \theta\frac{PX}{Z}$, as demonstrated in Eq. (6.6). This expression is associated with the local gradient at the point where the pre-activation value is fed into the activation function and the ratio between the pre-activation value of the activation function and the resulting activated output. In instances where discrete values represent digital signals, it is feasible to reduce the local gradient to approximately zero. Furthermore, the ratio between pre-activation and activated values can be constrained with an upper limit. This enables us to effectively block the propagation of error during the feed-forward decision-making process.

A similar principle has been demonstrated in a study that developed and optimised a neural network consisting of 13.96 million neurons on a photonic analog platform [272]. This supports our hypothesis that by partitioning a neural network into multiple sub-networks, named as capsules, and discretising the inputs and outputs of each capsule network, we can enhance the scalability of the neural network while maintaining its robustness against noise in signals and device tolerances.

The ML algorithm decision tree (DT) is one of the potential topology capable to block the error accumuation path. The algorithm has attracted significant research interest since its inception, notable for its versatility in classification, robustness and transparency [273]. Its impressive achievements, akin to the famous AlphaGo case, illustrate its untapped potential when integrated with neural network frameworks [274]. Recent research has not only examined its generalisation and efficiency, but also explored its applications in Industry 4.0 [275] and conducted large-scale comparison studies [276]. The idea of integrating neural networks with the system to improve precision while preserving interpretability has been extensively explored [277].

A novel framework, named NDT, combines the widely used concept neural network and DT to create a versatile algorithm. The NDT algorithm has the ability to classify data with a logarithmically increasing number of justifications versus the number of classes, as seen in DT, while also being able to fit any function like artificial neural networks (ANNs). This system has been shown to learn more efficiently than either of the two individual algorithms in various scenarios, including supervised, semi-supervised, or unsupervised cases. [278].

The hierarchical structure of neural networks presents challenges for researchers in providing explanations for the decisions made during training or when used in applications. The depth of layers in neural networks results in a decreasing trend in explainability, as the information of input vectors gets mixed and transformed [179]. In contrast, DT, while not as capable in complex classification tasks with high-dimensional inputs as neural networks, offer clearer relationships between the input and output of each classifier (referred to as ”nodes”) in their decision paths.

Recent studies have shown that the NDT systems can be trained using methods similar to those used for neural networks [279], indicating the potential to reach reliable local minima using Hebbian-based algorithms or local search algorithms. Furthermore, it is possible to extend the algorithm to function as a

random forest with a gradient descent-based approach [280]. Research by Frosst and Hinton has shown that an 8-layer soft NDT can achieve an impressive accuracy of 96.8% in validation on the MNIST dataset [179], making it a promising benchmark for future work.

In the realm of electronics, there have been advances in the use of molecular memristor crossbars to produce DTs. This study explores the possibility of using the multi-state transfer function that occurs between the applied voltage and the current flow, to serve as a decision-making system. The experiment involved implementations to simulate the outputs of various logic gates to assess the efficacy of this technique [281].

The focus on the area, as seen from the perspective of this work, has the potential not only to streamline decision-making processes for researchers but also to address the issue of error accumulation in practical scenarios. As can be known from the related literature, the structure of DTs disrupts this accumulation by using the original input data at each layer, rather than the cascading output. This approach enhances modularity, allowing for independent testing and tuning of each module. Additionally, the manageable size of each sub-network ensures the algorithm's effective application. Thus, this topology presents a viable strategy for scaling the proposed HNN design in terms of complexity while hence improving decision-making capabilities.

In order to rigorously assess the implementation and optimisation of the HNN system, it is imperative not only to replicate the results reported in previous studies utilising the MNIST dataset [179], but also to develop an HNN-based analog-digital converter (ADC) for preliminary validation of the design's feasibility. This proposed example exhibits well-defined characteristics and demonstrates a significant degree of linear separability. Even not being able to achieve a expected learning outcome with optimisers, it is conducive to analytically assign parameters to the neural network, thereby ensuring the reliability of the hardware design whilst utilising these examples as benchmarks.

For the ADC application, one can deduce that the resulting output is contingent upon both the input and the preceding output, with each output being distinctly quantized. Consequently, it would be straightforward to design a tree structure to encapsulate all potential outcomes. Furthermore, it is advisable to consider the self-similarity and repetitive patterns present in the branches to streamline the overall topology. Our research indicates that the resultant design will closely align with conventional ADC in practise.

The MoE employs a top-down approach that effectively models complex and diverse data generation processes. This methodology is designed to meet the criteria necessary for enabling digitized outputs within its sub-networks. It is commonly used for tasks like classification, clustering, and regression in various industries such as business, science, and technology. Different variations of this framework have been proposed, each offering unique advantages tailored to specific fields [282].

This system consists of multiple Gate-controlled sub-modules referred to as "Experts". Each Expert, along with its corresponding "Gate", receives a duplicate of the original input data vector. The Gates uses the **softmax** function to determine which Expert is best suited to address the specific problem at hand. These Experts, which are multi-layer perceptrons, focus on processing a sub-

set of the task based on the Divide and Conquer approach. The sub-division of tasks can either be implicitly determined through stochastic partitioning initially and refined by Gates during training, or explicitly pre-defined using clustering methods [283].

The training procedure for this method follows a similar structure to traditional neural networks, utilising back-propagation to adjust the parameters of each Expert based on their respective loss function. The loss functions utilised for training each component of the system are variants of the MSE function. However, the key distinction lies in the fact that in optimising the Experts, the loss term is influenced by a normalised output from the corresponding Gate.

As discussed previously, building a back-propagation system that integrates a fuzzy system on hardware or software platforms poses a significant challenge. However, the implementation of this design remains a feasible goal in practical applications. The design of the multiplier connecting Gates to its corresponding Expert and the accumulator in the final stage can both be simplified using a multiplexer (MUX) and applying the modifications of the **max** functions as an alternative to the **softmax** function applied for enhanced simplicity. The training process can also be adjusted and streamlined accordingly. To be more specific, the loss functions of Experts can be reverted to a standard MSE function for the examples within their designated sub-regions.

In a similar vein, in addition to employing the illustrative examples presented in related research, it is advisable to initially develop a basic arithmetic logic unit (ALU) as a foundational step towards the hardware realisation of MoE structure. While it is standard practice for modern devices to utilise ALUs with 32 or 64 bits in terms of command sets and inputs, for the purposes of verification within this study, it is strongly recommended to utilise 4-bit structure as the starting point, and accordingly estimate the power and footprint consumption for practical-case systems.

Regarding the design of the 4-bit ALU, it is logical to conceptualise the command as the Gate required in the MoE configurations, with each functional block consisted by neural networks serving as specialised Experts. This framework allows for the efficient progression of optimisation and analytical assignment processes. The operational procedures to be implemented within the HNN-based ALU may incorporate a synergistic blend of both analog and logic operations, enabled through the utilisation of neural networks.

To demonstrate the reliability and generalisation ability of the system and associated optimisers, we propose utilising the MNIST dataset as a benchmark for testing purposes. Given the complexities associated with the high-dimensionality of this dataset, constructing a demonstration system manually may present certain challenges. Therefore, our primary objective is to modify the layer-cascading topology through a combination of various sub-networks featuring binary or ternary inputs and outputs. This objective can be effectively achieved by training multiple filters to categorise the dataset into two roughly balanced classes, thereby facilitating the creation of a Gate to optimise a network that classifies data pieces corresponding to these categories as Experts and subsequent Gates.

Moreover, to enhance the optimisation processes for the Gates, we recommend employing low-resolution input derived from the original $[28 \times 28]$ pixel input. This approach aims to reduce parameter usage while ensuring that the

Experts are optimally configured with a maximum number of parameters assigned to zero or set to a low-resolution mode, consistent with the MAC specifications. Consequently, within this architecture, it is advisable to design distinct sub-networks for each pixel output, integrating edge-detection capabilities through a series of **convolution** layers.

Furthermore, leveraging the outputs from each sub-network as inputs to other sub-networks can significantly strengthen the system's self-correction capabilities. It is advisable to design the architecture of each sub-network with a maximum of one hidden layer to facilitate optimisation through Hebbian learning algorithms in scenarios where advancements in the optimiser are not feasible.

The topologies under discussion exhibit a markedly increased number of connections in comparison to a single neural network. In particular, all nodes within the NDTs and all Experts and Gates in the MoE models necessitate full connections to the input, resulting in a notably complex hardware implementation.

To address this challenge, one proposed strategy involves leveraging the Hebbian learning principle to evaluate the influence of each pixel on the outputs based on its variability. For the initial layer, the weight update process can be articulated as follows:

$$\Delta w = (x_i - \langle x_i \rangle)(t_j - x_j), \tag{7.4}$$

where $\langle x_i \rangle$ denotes the average activation level of the input neuron x_i.

The methodology employed has undergone thorough analysis utilising the MNIST dataset within a single-layer perceptron structured as $[784 \times 10]$. Results from this experimentation indicate that the optimised parameters successfully highlight the unique attributes of each class while also identifying pixels exhibiting minimal alterations in the training datasets. The single-layer architecture achieved a validation accuracy rate marginally below 90%. In contrast, a convolutional neural network (CNN) can reach an output accuracy exceeding 99% [179], thereby providing a basis for assessing the relative performance decline of the current system under development, as well as facilitating a fundamental comparison regarding the utilisation of parameters and neurons.

7.4 Conclusion

We analyzed the proposed HNN system and identified the necessity for a new optimisation algorithm that operates without explicit knowledge of the system's internal workings including the magnitude of each of its parameter and behaviour of activation functions involved.

We identified several technical approaches from the literature, including back-propagation approximations, local search algorithms, and Hebbian optimisation methods. Although some have been tested in our experiments (*e.g.* the local searching algorithm and Hebbian-related algorithms), they have not consistently yielded satisfactory results, necessitating further performance analysis.

We recognize the need to design an interface between the proposed HNN circuit and optimisers built on software or hardware, as its performance depends on circuit implementation and optimisation methods. The interface should up-

load datasets to the HNN circuit, interpret outputs to compute error terms, and update the circuit's memory with revised parameters. However, this step can only be proceed after verifying the feasibility of optimisation algorithm and neural network topology.

Furthermore, we have delineated two prospective applications informed by recent developments in the field. The first application focuses on the integration of TinyML as it interfaces with IoT systems. This approach emphasizes the utilisation of embedded neural networks to enhance smart sensor systems capable of executing non-polynomial transformations and self-calibration. The second application pertains to advancements in NDT or MoE systems aimed at facilitating complex decision-making processes. This application seeks to improve topology to sustain decision-making efficacy while mitigating the risks of failure associated with error accumulation during analog computing process. Alternatively, we could also explore the implementation of more advanced technologies to expand the system's applicability across diverse industries. By synergising these research trajectories, we are confident that our proposed system can be rigorously validated and effectively applied to real-world scenarios.

Chapter 8

Conclusion

In light of the premise that analog computing schemes and their hardware implementations may prove to be more effective than conventional methods of using servers to solve neural networks, this study has undertaken a comprehensive examination of recent advancements in this field. It has successfully introduced and verified a novel activation function circuit (AFC) to address existing research gaps, as well as enhanced the multiply accumulate circuit (MAC) using the latest technological processes. As a result, efficiency in terms of power consumption and response time has improved significantly, reaching a cutting-edge level of performance.

In Chapter 4, we have verified the systematic robustness of our proposed push-pull linear follower in producing a stable transfer function. The transfer function produced has been found to be a well-performing activation function when compared to commonly used activation functions such as *Rectified Linear Unit* (*ReLU*) and $\tanh(\cdot)$ in terms of learning efficiency and final performance according to our validations. Furthermore, the behaviour of the proposed activation function based on this transfer function is similar to the function $\tanh_{\mathrm{H}}(\cdot)$, which enhances the neural network's capacity to solve practical problems with self-similarity in a recursive manner. Additionally, we have identified an Achilles' Heel of this design when utilised as an AFC in terms of output impedance and fan-out capability.

We have developed a novel weighting circuit that addresses the limitations of the AFC discussed in Chapter 5. This circuit has demonstrated scalability in terms of bit resolutions and boasts exceptional efficiency, reaching a state-of-the-art (SOTA) level. Moreover, the manufacturing process for this circuit is compatible with current industrial technology. Building upon this innovation, we have also designed two summation circuits that prioritise stability and efficiency, respectively. Additionally, we have discussed the scalability in terms of neural network topology with the incorporation of a two-stage MAC. Our simulations indicate that the push-pull circuit proposed as AFC will greatly enhance scalability and performance.

Based on the previous chapters, a detailed analysis has been conducted on the performance of the hardware neural network (HNN) system comprising the modules discussed in Chapter 6. By carefully considering inaccuracies and potential failures, a model has been developed to depict the accumulation of errors throughout the sequential operations. The analysis of non-idealities encom-

passes the potential variations in each element and noise present in the signal channels, establishing a new and comprehensive foundation in an underexplored area. Furthermore, adjustments have been made to the modules in response to the system's non-ideal nature, with an assessment of the additional system's impact on robustness and efficiency.

Based on our previous analysis, it appears that the primary challenge facing the system is an optimisation issue related to inaccuracies in both the feed-forward and back-propagation processes. Addressing this issue is crucial for future implementations and practical applications. With this in mind, we have reviewed the literature on optimisation techniques and identified approximation, local search, and Hebbian learning as the most promising methods that could potentially be implemented efficiently, even on a stand-alone chip. Furthermore, we have identified two specific areas where our proposed system could be beneficial. Given the model we have developed, the characteristics of the proposed network are outlined in Chapter 7. Furthermore, Chapter 7 suggests potential experiments for the two topics that have not yet met our expectations.

To summarize, the work primarily established the following key objectives:

- Developed circuits capable of integration into implementations of hardware neural network.
- Presented a model for predicting final stage output error through system perturbations.
- Proposed an optimisation scheme to facilitate on-chip learning in HNNs.
- Identified potential applications for neural networks with fuzzy characteristics.

8.1 Summary of Contributions

It is worth noting that, according to Mhaskar's conclusion, any non-polynomial function can be utilised as an activation function in neural networks [10]. It is also important to highlight that non-polynomial transfer functions can be created through basic analog or digital circuits. Therefore, it is possible to design a circuit with specific input and output requirements, assuming the transfer function is non-polynomial within the operational range, to act as an activation function generator. Instead of replicating existing activation functions found in literature, it may be more effective to design a custom circuit and emphasis its learning efficiency compared to other commonly used functions.

The primary focus should be on the efficiency and reliability of generating the function through physical implementations in various working conditions for HNNs. The mathematical expressions of transfer functions in circuits may appear intricate, but they are actually straightforward to achieve with hardware. This is in contrast to popular functions like *ReLU* or $\tanh(\cdot)$, which may appeal to be effort-consuming during implementations.

With the insightful information gained, we are able to underscore the significance of the topics covered in Chapter 7. Using a black box transfer function as an activation function poses challenges in the efficiency of utilising well-researched and commonly used derivative-based optimisation algorithms. However, it is still possible to view linear transforming circuits as finite-state ma-

chines and implement specific searches or optimisations on their foundation. Despite not yet achieving optimal results, we have observed promising signs of untapped potential within these algorithms.

If our hypothesis that these algorithms can effectively reach a local minimum like gradient descent-based algorithms proves correct, a significant obstacle in implementing and on-chip training of HNNs will be overcome. By understanding the principles and theorems behind these algorithms, particularly those based on the Hebbian method, we can gain insight into the behaviour of the neural networks they produce. Combined with discussions on the topology of the neural decision tree (NDT) or mixture of experts (MoE), we may be able to develop a transparent and reliable system that is also highly resilient and easy to troubleshoot. Therefore, we can address concerns related to precision in this field.

Based on the findings of our conducted experiments, particularly within the context of the regression problem utilising neural network-based Gray Code analog-digital converter (ADC)it can be asserted that a methodical examination of issues and the precise allocation of parameters has the potential to greatly enhance the overall performance of the system. This approach may result in improved systematic efficiency and a reduction in unnecessary duplications. Furthermore, this assertion could be extrapolated to other unresolved issues, offering a clearer insight into the processing advances that may not have been previously understood. As per the assertion, enhancing the functionality of capsule neural network, NDT or MoE system has the potential to enhance the comprehensive systematic performance in terms of accuracy and efficiency in this capacity.

Although the previous assertion may not hold true in its entirety, the system we have devised demonstrates potential as a viable solution for edge devices and stand-alone smart sensor systems seeking to execute intricate calculations. Impressively, the system has been proven to deliver optimal efficiency in terms of both response time and power consumption. Furthermore, the required chip area for implementing neural network is noticeably reduced compared to alternative designs offering a similar resolution for the corresponding network. This system also boasts ease of implementation using existing technologies, presenting a significant advantage.

8.2 Limitations and Challenges

The research conducted has some limitations, as the experiments were mainly conducted in a simulated environment and the study is a combination of two loosely related fields of research interest. These limitations may affect the representativeness, reliability, and applicability of the study, as well as the quality, diversity, and validity of the data.

From a methodological perspective, the simulation program with integrated circuit emphasis (SPICE) and the predictive technology model (PTM) utilised can be seen as an analytical approximation of the real behaviour of the system being studied. As a new design of the complementary metal oxide semiconductor (CMOS) circuit, adjustments may need to be made to the model in certain parameters that differ from the typical range. This could include examining performance with depletion mode transistor pairs or in cases of significant

mismatch. The complexity of the system being analyzed also creates challenges in reaching a static solution. While we can theoretically predict the final outcome, in practice, manual adjustments to initial conditions may be necessary to expedite the process. Given the multiple free parameters involved, there is a concern about whether the solved output is the only stable transfer function of the system or if there are other stable or unstable solutions to the transfer function.

Furthermore, in conducting both the theoretical and empirical aspects of this research, it is plausible to uncover substantial variances in terminology and emphasis within the realms of electronic engineering and computer science, as well as mathematics and physics. This situation leads to the recognition that there exists a gap in the current body of knowledge, with only a limited amount of literature delving into the foundational components of this area of study.

As detailed in Chapter 2, , the predominant focus of investigations into systematical robustness primarily revolves around what is known as local robustness. That is, the system's sensitivity to perturbations within specific examples. Regrettably, there seems to be a tendency to overlook the potential existence of errors in system operation, which could occur universally across all potential inputs. Similarly, individuals researching hardware implementations of neural networks seem to disregard the essential principles of neural networks and instead, direct their energies towards replicating specific mathematical functions without establishing a comprehensive framework delineating the system's overarching traits.

Consequently, the absence of a standardized benchmark raises questions about the validity of claims regarding a system's distinctive features or advantages. The absence of empirical testing also renders the study less responsive to possible unpredictable outcomes in real-world applications. Therefore, the expeditious creation of independent circuits and chips for authentication purposes may be a critical point of consideration.

During our research, we have proposed an alternative method of linking the transfer function of both linear and non-linear systems to their corresponding equivalents in neural network applications. By doing so, we can evaluate efficiency and robustness effectively. Our hypothesis suggests that a circuit's ability to mimic neural network operations only depends on the individual abilities of its sub-modules to perform equivalent tasks independently. we may thus be able to simplify the overall simulation boundary and analyse the performance accordingly. If our hypothesis is valid, the approach allows for a simplified simulation boundary and performance analysis. Additionally, by establishing mathematical equivalences, we can potentially estimate final-stage errors based on input and output errors at each stage.

It is advisable to create and assess a circuit utilising predefined tasks to confirm the validity of our assertions. The criteria can be established according to the standards outlined in this project, focusing on the effectiveness at both the systematic and component levels, along with assessing its robustness against any randomized signal distortions and element variabilities. This should be done assuming the system with proper modifications and fine-tuning can achieve an output that aligns with the system's specifications, or attains a certain level of accuracy when compared to the current cutting-edge practices in the field.

Bibliography

[1] V. S. Dave and K. Dutta, "Neural network based models for software effort estimation: a review," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 295–307, 2014.

[2] A. Bulsari, "Some analytical solutions to the general approximation problem for feedforward neural networks," *Neural networks*, vol. 6, no. 7, pp. 991–996, 1993.

[3] N. Izeboudjen, C. Larbes, and A. Farah, "A new classification approach for neural networks hardware: from standards chips to embedded systems on chip," *Artificial Intelligence Review*, vol. 41, pp. 491–534, 2014.

[4] A. Mozaffari, M. Emami, and A. Fathi, "A comprehensive investigation into the performance, robustness, scalability and convergence of chaos-enhanced evolutionary algorithms with boundary constraints," *Artificial Intelligence Review*, vol. 52, pp. 2319–2380, 2019.

[5] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[6] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications:

A survey," *Heliyon*, vol. 4, no. 11, 2018.

[7] G. B. Kingston, H. R. Maier, and M. F. Lambert, "Calibration and validation of neural networks to ensure physically plausible hydrological modeling," *Journal of Hydrology*, vol. 314, no. 1-4, pp. 158–176, 2005.

[8] V. Camus, C. Enz, and M. Verhelst, "Survey of precision-scalable multiply-accumulate units for neural-network processing," in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 57–61, 2019.

[9] H. N. Mhaskar and C. A. Micchelli, "Approximation by superposition of sigmoidal and radial basis functions," *Advances in Applied mathematics*, vol. 13, no. 3, pp. 350–373, 1992.

[10] H. N. Mhaskar and C. A. Micchelli, "How to choose an activation function," *Advances in neural information processing systems*, vol. 6, 1993.

[11] C. Reams, *Modelling energy efficiency for computation*. PhD thesis, University of Cambridge, 2012.

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

[13] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," *Advances in neural information processing systems*, vol. 20, 2007.

[14] 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)*, pp. 10–14, IEEE, 2014.

[15] C. Latotzke and T. Gemmeke, "Efficiency versus accuracy: a review of design techniques for dnn hardware accelerators," *IEEE Access*, vol. 9,

pp. 9785–9799, 2021.

[16] S. Mittal, "A survey of fpga-based accelerators for convolutional neural networks," *Neural computing and applications*, vol. 32, no. 4, pp. 1109–1139, 2020.

[17] R. Machupalli, M. Hossain, and M. Mandal, "Review of asic accelerators for deep neural network," *Microprocessors and Microsystems*, vol. 89, p. 104441, 2022.

[18] H. Valavi, P. J. Ramadge, E. Nestler, and N. Verma, "A 64-tile 2.4-mb in-memory-computing cnn accelerator employing charge-domain compute," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1789–1799, 2019.

[19] S. Yin, Z. Jiang, J.-S. Seo, and M. Seok, "Xnor-sram: In-memory computing sram macro for binary/ternary deep neural networks," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 6, pp. 1733–1743, 2020.

[20] J. Dean, "1.1 the deep learning revolution and its implications for computer architecture and chip design," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 8–14, IEEE, 2020.

[21] R. Nägele, J. Finkbeiner, V. Stadtlander, M. Grözing, and M. Berroth, "Analog multiply-accumulate cell with multi-bit resolution for all-analog ai inference accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[22] B. Murmann, "Mixed-signal computing for deep neural network inference," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 3–13, 2020.

[23] B.-E. Verhoef, N. Laubeuf, S. Cosemans, P. Debacker, I. Papistas,

A. Mallik, and D. Verkest, "Fq-conv: Fully quantized convolution for efficient and accurate inference," *arXiv preprint arXiv:1912.09356*, 2019.

[24] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, pp. 1–6, IEEE, 2013.

[25] M. S. Ansari, V. Mrazek, B. F. Cockburn, L. Sekanina, Z. Vasicek, and J. Han, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 2, pp. 317–328, 2019.

[26] H. Graf, L. Jackel, R. Howard, B. Straughn, J. Denker, W. Hubbard, D. Tennant, and D. Schwartz, "Vlsi implementation of a neural network memory with several hundreds of neurons," in *AIP Conference Proceedings 151 on Neural Networks for Computing*, pp. 182–187, 1987.

[27] A. Muthuramalingam, S. Himavathi, and E. Srinivasan, "Neural network implementation using fpga: issues and application," *International Journal of Electrical and Computer Engineering*, vol. 2, no. 12, pp. 2802–2808, 2008.

[28] A. Jain, A. Phanishayee, J. Mars, L. Tang, and G. Pekhimenko, "Gist: Efficient data encoding for deep neural network training," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 776–789, IEEE, 2018.

[29] M. Rhu, M. O'Connor, N. Chatterjee, J. Pool, Y. Kwon, and S. W. Keckler, "Compressing dma engine: Leveraging activation sparsity for training deep neural networks," in *2018 IEEE International Symposium on High*

Performance Computer Architecture (HPCA), pp. 78–91, IEEE, 2018.

[30] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, *et al.*, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 609–622, IEEE, 2014.

[31] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

[32] I. A. Papistas, S. Cosemans, B. Rooseleer, J. Doevenspeck, M.-H. Na, A. Mallik, P. Debacker, and D. Verkest, "A 22 nm, 1540 top/s/w, 12.1 top/s/mm 2 in-memory analog matrix-vector-multiplier for dnn acceleration," in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, pp. 1–2, IEEE, 2021.

[33] X. Guo, F. M. Bayat, M. Bavandpour, M. Klachko, M. Mahmoodi, M. Prezioso, K. Likharev, and D. Strukov, "Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded nor flash memory technology," in *2017 IEEE International Electron Devices Meeting (IEDM)*, pp. 6–5, IEEE, 2017.

[34] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaaauw, and R. Das, "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *2018 ACM/IEEE 45Th annual international symposium on computer architecture (ISCA)*, pp. 383–396, IEEE, 2018.

[35] S. Li, A. O. Glova, X. Hu, P. Gu, D. Niu, K. T. Malladi, H. Zheng, B. Bren-

nan, and Y. Xie, "Scope: A stochastic computing engine for dram-based in-situ accelerator," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 696–709, IEEE, 2018.

[36] A. X. M. Chang, B. Martini, and E. Culurciello, "Recurrent neural networks hardware implementation on fpga," *arXiv preprint arXiv:1511.05552*, 2015.

[37] S. Draghici, "Neural networks in analog hardware—design and implementation issues," *International journal of neural systems*, vol. 10, no. 01, pp. 19–42, 2000.

[38] S. Jain, A. Ankit, I. Chakraborty, T. Gokmen, M. Rasch, W. Haensch, K. Roy, and A. Raghunathan, "Neural network accelerator design with resistive crossbars: Opportunities and challenges," *IBM Journal of Research and Development*, vol. 63, no. 6, pp. 10–1, 2019.

[39] H. El-Rewini and M. Abd-El-Barr, *Advanced computer architecture and parallel processing*. John Wiley & Sons, 2005.

[40] F. Bistouni and M. Jahanshahi, "Scalable crossbar network: a non-blocking interconnection network for large-scale systems," *The Journal of Supercomputing*, vol. 71, pp. 697–728, 2015.

[41] F. Aguirre, A. Sebastian, M. Le Gallo, W. Song, T. Wang, J. J. Yang, W. Lu, M.-F. Chang, D. Ielmini, Y. Yang, *et al.*, "Hardware implementation of memristor-based artificial neural networks," *Nature Communications*, vol. 15, no. 1, p. 1974, 2024.

[42] J. H. Yoon, Z. Wang, K. M. Kim, H. Wu, V. Ravichandran, Q. Xia, C. S. Hwang, and J. J. Yang, "An artificial nociceptor based on a diffusive mem-

ristor," *Nature communications*, vol. 9, no. 1, p. 417, 2018.

[43] E. H. Lee and S. S. Wong, "Analysis and design of a passive switched-capacitor matrix multiplier for approximate computing," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 261–271, 2016.

[44] Y. Zhang and D. El-Damak, "A reconfigurable passive switched-capacitor multiply-and-accumulate unit for approximate computing," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 921–924, IEEE, 2020.

[45] Z. Wang, M. Rao, J.-W. Han, J. Zhang, P. Lin, Y. Li, C. Li, W. Song, S. Asapu, R. Midya, *et al.*, "Capacitive neural network with neuro-transistors," *Nature communications*, vol. 9, no. 1, p. 3208, 2018.

[46] K.-U. Demasius, A. Kirschen, and S. Parkin, "Energy-efficient memcapacitor devices for neuromorphic computing," *Nature Electronics*, vol. 4, no. 10, pp. 748–756, 2021.

[47] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, no. 1-3, pp. 239–255, 2010.

[48] S. Hwang, J. Yu, G. H. Lee, M. S. Song, J. Chang, K. K. Min, T. Jang, J.-H. Lee, B.-G. Park, and H. Kim, "Capacitor-based synaptic devices for hardware spiking neural networks," *IEEE Electron Device Letters*, vol. 43, no. 4, pp. 549–552, 2022.

[49] C. Rasche and R. Douglas, "An improved silicon neuron," *Analog integrated circuits and signal processing*, vol. 23, no. 3, pp. 227–236, 2000.

[50] A. Van Schaik, "Building blocks for electronic spiking neural networks,"

Neural networks, vol. 14, no. 6-7, pp. 617–628, 2001.

[51] M.-K. Park, W.-M. Kang, R.-H. Koo, J.-H. Kim, J. Hwang, J.-H. Bae, J.-J. Kim, and J.-H. Lee, "Cointegration of the tft-type and flash synaptic array and cmos circuits for a hardware-based neural network," *IEEE Transactions on Electron Devices*, vol. 70, no. 1, pp. 93–98, 2023.

[52] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[53] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[54] J. Wu, X. Lin, Y. Guo, J. Liu, L. Fang, S. Jiao, and Q. Dai, "Analog optical computing for artificial intelligence," *Engineering*, vol. 10, pp. 133–145, 2022.

[55] Y. Toyama, K. Yoshioka, K. Ban, S. Maya, A. Sai, and K. Onizuka, "An 8 bit 12.4 tops/w phase-domain mac circuit for energy-constrained deep learning accelerators," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 10, pp. 2730–2742, 2019.

[56] Y. Van De Burgt, E. Lubberman, E. J. Fuller, S. T. Keene, G. C. Faria, S. Agarwal, M. J. Marinella, A. Alec Talin, and A. Salleo, "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing," *Nature materials*, vol. 16, no. 4, pp. 414–418, 2017.

[57] S. Kuninti and S. Rooban, "Backpropagation algorithm and its hardware implementations: A review," in *Journal of Physics: Conference Series*, vol. 1804, p. 012169, IOP Publishing, 2021.

[58] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, "A survey of fpga-based neural network accelerator," *arXiv preprint arXiv:1712.08934*, 2017.

[59] T. Hirtzlin, M. Bocquet, B. Penkovsky, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz, "Digital biologically plausible implementation of binarized neural networks with differential hafnium oxide resistive memory arrays," *Frontiers in neuroscience*, vol. 13, p. 1383, 2020.

[60] S. Yin, X. Sun, S. Yu, and J.-s. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated rram and 90-nm cmos," *IEEE Transactions on Electron Devices*, vol. 67, no. 10, pp. 4185–4192, 2020.

[61] Z. Wang, S. Joshi, S. Savel'Ev, W. Song, R. Midya, Y. Li, M. Rao, P. Yan, S. Asapu, Y. Zhuo, *et al.*, "Fully memristive neural networks for pattern classification with unsupervised learning," *Nature Electronics*, vol. 1, no. 2, pp. 137–145, 2018.

[62] F. Kiani, J. Yin, Z. Wang, J. J. Yang, and Q. Xia, "A fully hardware-based memristive multilayer neural network," *Science advances*, vol. 7, no. 48, p. eabj4801, 2021.

[63] S. Oh, Y. Shi, J. Del Valle, P. Salev, Y. Lu, Z. Huang, Y. Kalcheim, I. K. Schuller, and D. Kuzum, "Energy-efficient mott activation neuron for full-hardware implementation of neural networks," *Nature nanotechnology*, vol. 16, no. 6, pp. 680–687, 2021.

[64] B. Li and G. Shi, "A CMOS rectified linear unit operating in weak inversion for memristive neuromorphic circuits," *Integration*, vol. 87, pp. 24–28, 2022.

[65] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao, *et al.*, "A compute-in-memory chip based on resistive random-access memory," *Nature*, vol. 608, no. 7923, pp. 504–512, 2022.

[66] L. Li, S. Zhang, and J. Wu, "An efficient hardware architecture for activation function in deep learning processor," in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 911–918, IEEE, 2018.

[67] E. van Keulen, S. Colak, H. Withagen, and H. Hegt, "Neural network hardware performance criteria," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 3, pp. 1885–1888, IEEE, 1994.

[68] T. Cornu and P. Ienne, "Performance of digital neuro-computers," in *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 87–93, IEEE, 1994.

[69] A. Krishna, S. R. Nudurupati, D. Chandana, P. Dwivedi, A. van Schaik, M. Mehendale, and C. S. Thakur, "Raman: A re-configurable and sparse tinyml accelerator for inference on edge," *IEEE Internet of Things Journal*, 2024.

[70] M. Liu, B. Zhou, Z. Zhao, H. Hong, H. Kim, S. Suh, V. F. Rey, and P. Lukowicz, "Fieldhar: A fully integrated end-to-end rtl framework for human activity recognition with neural networks from heterogeneous sensors," in *2023 IEEE 34th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 110–118, IEEE, 2023.

[71] P. Toupas, C.-S. Bouganis, and D. Tzovaras, "Fmm-x3d: Fpga-based modeling and mapping of x3d for human action recognition," in *2023 IEEE 34th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 119–126, IEEE, 2023.

[72] P. Toupas, C.-S. Bouganis, and D. Tzovaras, "fpgahart: A toolflow for throughput-oriented acceleration of 3d cnns for har onto fpgas," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 86–92, IEEE, 2023.

[73] Z. Que, S. Liu, M. Rognlien, C. Guo, J. G. Coutinho, and W. Luk, "Metaml: Automating customizable cross-stage design-flow for deep learning acceleration," in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 248–252, IEEE, 2023.

[74] M. T. L. Aung, D. Gerlinghoff, C. Qu, L. Yang, T. Huang, R. S. M. Goh, T. Luo, and W.-F. Wong, "Deepfire2: A convolutional spiking neural network accelerator on fpgas," *IEEE Transactions on Computers*, vol. 72, no. 10, pp. 2847–2857, 2023.

[75] S. Dey, P. Dasgupta, and P. P. Chakrabarti, "Dietcnn: Multiplication-free inference for quantized cnns," in *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2023.

[76] B. Noory and V. Groza, "A reconfigurable approach to hardware implementation of neural networks," in *CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No. 03CH37436)*, vol. 3, pp. 1861–1864, IEEE, 2003.

[77] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, "Piecewise linear ap-

proximation applied to nonlinear function of a neural network," *IEE Proceedings-Circuits, Devices and Systems*, vol. 144, no. 6, pp. 313–317, 1997.

[78] A. Agrawal, M. Kar, K.-H. Kim, S. Rylov, J. Jung, S. Munetoh, K. Ho-Sokawa, X. Zhang, B. Hekmatshoartabari, F. Carta, *et al.*, "A switched-capacitor integer compute unit with decoupled storage and arithmetic for cloud ai inference in 5nm cmos," in *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pp. 1–2, IEEE, 2023.

[79] 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, *et al.*, "15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*, pp. 244–246, IEEE, 2020.

[80] Y. Chen, L. Lu, B. Kim, and T. T.-H. Kim, "A reconfigurable 4t2r reram computing in-memory macro for efficient edge applications," *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 210–222, 2021.

[81] H. Jiang, W. Li, S. Huang, and S. Yu, "A 40nm analog-input adc-free compute-in-memory rram macro with pulse-width modulation between sub-arrays," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pp. 266–267, IEEE, 2022.

[82] K. Prabhu, A. Gural, Z. F. Khan, R. M. Radway, M. Giordano, K. Koul, R. Doshi, J. W. Kustin, T. Liu, G. B. Lopes, *et al.*, "Chimera: A 0.92-tops, 2.2-tops/w edge ai accelerator with 2-mbyte on-chip foundry resistive ram for efficient training and inference," *IEEE Journal of Solid-State Circuits*,

vol. 57, no. 4, pp. 1013–1026, 2022.

[83] 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.-Y. Huang, T.-W. Chang, *et al.*, "Embedded 1-mb reram-based computing-in-memory macro with multibit input and weight for cnn-based ai edge processors," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 203–215, 2019.

[84] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen, *et al.*, "33.2 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)*, pp. 500–502, IEEE, 2020.

[85] B. J. MacLennan, "A review of analog computing," *Department of Electrical Engineering & Computer Science, University of Tennessee, Technical Report UT-CS-07-601 (September)*, pp. 19798–19807, 2007.

[86] A. Dembo and T. Kailath, "Model-free distributed learning," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 58–70, 1990.

[87] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard, "The robustness of deep networks: A geometrical perspective," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 50–62, 2017.

[88] 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*, pp. 631–640, 2020.

[89] K. Cao, M. Liu, H. Su, J. Wu, J. Zhu, and S. Liu, "Analyzing the noise

robustness of deep neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 7, pp. 3289–3304, 2020.

[90] S. Chaudhury and T. Yamasaki, "Robustness of adaptive neural network optimization under training noise," *IEEE Access*, vol. 9, pp. 37039–37053, 2021.

[91] J. L. Holt and J.-N. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 281–290, 1993.

[92] P. D. Moerland and E. Fiesler, "Neural network adaptations to hardware implementations," in *Handbook of neural computation*, pp. E1–2, CRC Press, 2020.

[93] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[94] 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.

[95] T.-W. Weng, P. Zhao, S. Liu, P.-Y. Chen, X. Lin, and L. Daniel, "Towards certificated model robustness against weight perturbations," in *Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.

[96] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597, IEEE, 2016.

[97] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pp. 3–29, Springer, 2017.

[98] G. Li, S. K. S. Hari, M. Sullivan, T. Tsai, K. Pattabiraman, J. Emer, and S. W. Keckler, "Understanding error propagation in deep learning neural network (dnn) accelerators and applications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2017.

[99] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in neural information processing systems*, vol. 33, pp. 2958–2969, 2020.

[100] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[101] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in neural information processing systems*, vol. 31, 2018.

[102] M. D. Norton and J. O. Royset, "Diametrical risk minimization: Theory and computations," *Machine Learning*, pp. 1–19, 2021.

[103] 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*, pp. 2574–2582, 2016.

[104] P. Zhao, S. Wang, C. Gongye, Y. Wang, Y. Fei, and X. Lin, "Fault sneaking attack: A stealthy framework for misleading deep neural networks," in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6, 2019.

[105] X. Zeng and D. S. Yeung, "Sensitivity analysis of multilayer perceptron to input and weight perturbations," *IEEE Transactions on neural networks*, vol. 12, no. 6, pp. 1358–1366, 2001.

[106] S. Kwon, K. Lee, Y. Kim, K. Kim, C. Lee, and W. W. Ro, "Measuring error-tolerance in sram architecture on hardware accelerated neural network," in *2016 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, pp. 1–4, IEEE, 2016.

[107] C. Zhou, P. Kadambi, M. Mattina, and P. N. Whatmough, "Noisy machines: Understanding noisy neural networks and enhancing robustness to analog hardware errors using distillation," *arXiv preprint arXiv:2001.04974*, 2020.

[108] 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, p. 2473, 2020.

[109] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw)*, pp. 1–5, IEEE, 2015.

[110] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, *et al.*, "Going deeper with embedded fpga platform for convolu-

tional neural network," in *Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays*, pp. 26–35, 2016.

[111] J. Wang, Q. Lou, X. Zhang, C. Zhu, Y. Lin, and D. Chen, "Design flow of accelerating hybrid extremely low bit-width neural network in embedded fpga," in *2018 28th international conference on field programmable logic and applications (FPL)*, pp. 163–1636, IEEE, 2018.

[112] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 37, no. 1, pp. 35–47, 2017.

[113] F. Li, B. Liu, X. Wang, B. Zhang, and J. Yan, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[114] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *International Conference on Learning Representations*, 2016.

[115] 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.

[116] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[117] N. Cheney, M. Schrimpf, and G. Kreiman, "On the robustness of convolutional neural networks to internal architecture and weight perturbations," *arXiv preprint arXiv:1703.08245*, 2017.

[118] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connec-

tions for efficient neural network," *Advances in neural information processing systems*, vol. 28, 2015.

[119] A. P. Arechiga and A. J. Michaels, "The robustness of modern deep learning architectures against single event upset errors," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*, pp. 1–6, IEEE, 2018.

[120] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pp. 97–117, Springer, 2017.

[121] L. Pulina and A. Tacchella, "Challenging smt solvers to verify neural networks," *Ai Communications*, vol. 25, no. 2, pp. 117–135, 2012.

[122] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedforward neural networks to weight errors," *IEEE transactions on neural networks*, vol. 1, no. 1, pp. 71–80, 1990.

[123] A. Y. Cheng and D. S. Yeung, "Sensitivity analysis of neocognitron," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 29, no. 2, pp. 238–249, 1999.

[124] L. Xiang, X. Zeng, Y. Niu, and Y. Liu, "Study of sensitivity to weight perturbation for convolution neural network," *IEEE Access*, vol. 7, pp. 93898–93908, 2019.

[125] V. Zantedeschi, M.-I. Nicolae, and A. Rawat, "Efficient defenses against adversarial attacks," in *Proceedings of the 10th ACM workshop on artifi-*

cial intelligence and security, pp. 39–49, 2017.

[126] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, vol. 56, no. Suppl 1, pp. 1513–1589, 2023.

[127] V. Gabrel, C. Murat, and A. Thiele, "Recent advances in robust optimization: An overview," *European journal of operational research*, vol. 235, no. 3, pp. 471–483, 2014.

[128] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," *Mathematics of operations research*, vol. 23, no. 4, pp. 769–805, 1998.

[129] U. Shaham, Y. Yamada, and S. Negahban, "Understanding adversarial training: Increasing local stability of neural nets through robust optimization," *arXiv preprint arXiv:1511.05432*, 2015.

[130] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah, "In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning," *arXiv preprint arXiv:2101.06329*, 2021.

[131] E. Okewu, S. Misra, and F.-S. Lius, "Parameter tuning using adaptive moment estimation in deep learning neural networks," in *Computational Science and Its Applications–ICCSA 2020: 20th International Conference, Cagliari, Italy, July 1–4, 2020, Proceedings, Part VI 20*, pp. 261–272, Springer, 2020.

[132] T. Kavzoglu, "Increasing the accuracy of neural network classification using refined training data," *Environmental Modelling & Software*, vol. 24, no. 7, pp. 850–858, 2009.

[133] S. Chaudhury and T. Yamasaki, "Investigating generalization in neural networks under optimally evolved training perturbations," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3612, IEEE, 2020.

[134] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 ieee symposium on security and privacy (sp)*, pp. 39–57, Ieee, 2017.

[135] 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/CVF Conference on Computer Vision and Pattern Recognition*, pp. 588–597, 2019.

[136] O. Bastani, Y. Ioannou, L. Lampropoulos, D. Vytiniotis, A. Nori, and A. Criminisi, "Measuring neural net robustness with constraints," *Advances in neural information processing systems*, vol. 29, 2016.

[137] S. Taheri, M. Salem, and J.-S. Yuan, "Razornet: Adversarial training and noise training on a deep neural network fooled by a shallow neural network," *Big Data and Cognitive Computing*, vol. 3, no. 3, p. 43, 2019.

[138] F. Juefei-Xu, V. Naresh Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 19–28, 2017.

[139] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Perturbative neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3310–3318, 2018.

[140] A. Neelakantan, L. Vilnis, Q. V. Le, I. Sutskever, L. Kaiser, K. Kurach,

and J. Martens, "Adding gradient noise improves learning for very deep networks," *arXiv preprint arXiv:1511.06807*, 2015.

[141] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *arXiv preprint arXiv:1706.03825*, 2017.

[142] E. Dai, C. Aggarwal, and S. Wang, "Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 227–236, 2021.

[143] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*, pp. 1613–1622, PMLR, 2015.

[144] N. Pawlowski, A. Brock, M. C. Lee, M. Rajchl, and B. Glocker, "Implicit weight uncertainty in neural networks," *arXiv preprint arXiv:1711.01297*, 2017.

[145] X. He, L. Ke, W. Lu, G. Yan, and X. Zhang, "Axtrain: Hardware-oriented neural network training for approximate inference," in *Proceedings of the international symposium on low power electronics and design*, pp. 1–6, 2018.

[146] C. Wang, L. Xiong, J. Sun, and W. Yao, "Memristor-based neural networks with weight simultaneous perturbation training," *Nonlinear Dynamics*, vol. 95, no. 4, pp. 2893–2906, 2019.

[147] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78,

no. 9, pp. 1415–1442, 1990.

[148] M. Jabri and B. Flower, "Weight perturbation: An optimal architecture and learning technique for analog vlsi feedforward and recurrent multilayer networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 154–157, 1992.

[149] D. Psaltis and Y. Qiao, "Iv adaptive multilayer optical networks," in *Progress in optics*, vol. 31, pp. 227–261, Elsevier, 1993.

[150] R. D. Brandt and F. Lin, "Supervised learning in neural networks without feedback network," in *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, pp. 86–90, IEEE, 1998.

[151] P. Ienne, P. Thiran, and N. Vassilas, "Modified self-organizing feature map algorithms for efficient digital hardware implementation," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 315–330, 1997.

[152] H. Pujol, J.-O. Klein, E. Belhaire, and P. Garda, "Ra: An analog neurocomputer for the synchronous boltzmann machine," in *Proceedings of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 449–455, IEEE, 1994.

[153] B. Mittelstadt, C. Russell, and S. Wachter, "Explaining explanations in ai," in *Proceedings of the conference on fairness, accountability, and transparency*, pp. 279–288, 2019.

[154] E. Dai, T. Zhao, H. Zhu, J. Xu, Z. Guo, H. Liu, J. Tang, and S. Wang, "A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability," *arXiv preprint arXiv:2204.08570*, 2022.

[155] J. Pizarroso, J. Portela, and A. Muñoz, "Neuralsens: sensitivity analysis of neural networks," *arXiv preprint arXiv:2002.11423*, 2020.

[156] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[157] P. V. de Campos Souza, "Fuzzy neural networks and neuro-fuzzy networks: A review the main techniques and applications used in the literature," *Applied soft computing*, vol. 92, p. 106275, 2020.

[158] A. Schmid and Y. Leblebici, "Robust circuit and system design methodologies for nanometer-scale devices and single-electron transistors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 11, pp. 1156–1166, 2004.

[159] S. Mitra, H. Cho, T. Hong, Y. M. Kim, H.-H. K. Lee, L. Leem, Y. Li, D. Lin, E. Mintarno, D. Mui, *et al.*, "Robust system design," *IPSJ Transactions on System and LSI Design Methodology*, vol. 4, pp. 2–30, 2011.

[160] T.-C. Huang, J.-L. Huang, and K.-T. Cheng, "Robust circuit design for flexible electronics," *IEEE Design & Test of Computers*, vol. 28, no. 6, pp. 8–15, 2011.

[161] Y. Pan, Z. He, N. Guo, and Z. Zhang, "Distributionally robust circuit design optimization under variation shifts," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pp. 1–8, IEEE, 2023.

[162] H.-G. Beyer and B. Sendhoff, "Robust optimization–a comprehensive survey," *Computer methods in applied mechanics and engineering*, vol. 196, no. 33-34, pp. 3190–3218, 2007.

[163] B. L. Gorissen, İ. Yanıkoğlu, and D. Den Hertog, "A practical guide to robust optimization," *Omega*, vol. 53, pp. 124–137, 2015.

[164] D. Telen, M. Vallerio, L. Cabianca, B. Houska, J. Van Impe, and F. Logist, "Approximate robust optimization of nonlinear systems under parametric uncertainty and process noise," *Journal of Process Control*, vol. 33, pp. 140–154, 2015.

[165] S. Xu, Q. Wang, X. Wang, S. Wang, and T. T. Ye, "Multiplication through a single look-up-table (lut) in cnn inference computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 6, pp. 1916–1928, 2021.

[166] F. M. Dias, A. Antunes, and A. M. Mota, "Artificial neural networks: a review of commercial hardware," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 8, pp. 945–952, 2004.

[167] G. Serpen and Z. Gao, "Complexity analysis of multilayer perceptron neural network embedded into a wireless sensor network," *Procedia Computer Science*, vol. 36, pp. 192–197, 2014.

[168] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.

[169] R. Lee and I.-Y. Chen, "The time complexity analysis of neural network model configurations," in *2020 International conference on mathematics and computers in science and engineering (MACISE)*, pp. 178–183, IEEE, 2020.

[170] J. Šíma, "Energy complexity of recurrent neural networks," *Neural Com-*

putation, vol. 26, no. 5, pp. 953–973, 2014.

[171] J. W. Carr III, "Error analysis in floating point arithmetic," *Communications of the ACM*, vol. 2, no. 5, pp. 10–15, 1959.

[172] K. Kalliojarvi and J. Astola, "Roundoff errors in block-floating-point systems," *IEEE transactions on signal processing*, vol. 44, no. 4, pp. 783–790, 1996.

[173] A. Sanchez-Stern, P. Panchekha, S. Lerner, and Z. Tatlock, "Finding root causes of floating point error," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 256–269, 2018.

[174] D. A. Medler, M. R. Dawson, *et al.*, "Using redundancy to improve the performance of artificial neural networks," in *Proceedings of the Biennial Conference-Canadian Society for Computational Studies of Intelligence*, pp. 131–138, CANADIAN INFORMATION PROCESSING SOCIETY, 1994.

[175] F. Libano, P. Rech, B. Neuman, J. Leavitt, M. Wirthlin, and J. Brunhaver, "How reduced data precision and degree of parallelism impact the reliability of convolutional neural networks on fpgas," *IEEE Transactions on Nuclear Science*, vol. 68, no. 5, pp. 865–872, 2021.

[176] G. Ras, N. Xie, M. Van Gerven, and D. Doran, "Explainable deep learning: A field guide for the uninitiated," *Journal of Artificial Intelligence Research*, vol. 73, pp. 329–396, 2022.

[177] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning mod-

els," *arXiv preprint arXiv:1708.08296*, 2017.

[178] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson, "Explainable artificial intelligence: an analytical review," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 5, p. e1424, 2021.

[179] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[180] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Adaptive Computation and Machine Learning, The MIT Press, 2016.

[181] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.

[182] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Netw.*, vol. 3, no. 5, pp. 551–560, 1990.

[183] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[184] D. Hernandez and T. B. Brown, "Measuring the algorithmic efficiency of neural networks," *arXiv preprint arXiv:2005.04305*, 2020.

[185] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Compute and energy consumption trends in deep learning inference," *arXiv preprint arXiv:2109.05472*, 2021.

[186] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Di-*

gest of Technical Papers (ISSCC), pp. 10–14, 2014.

[187] H. P. Graf and L. D. Jackel, "Analog electronic neural network circuits," *IEEE Circuits and Devices magazine*, vol. 5, no. 4, pp. 44–49, 1989.

[188] I. Aleksander, W. Thomas, and P. Bowden, "WISARD· a radical step forward in image recognition," *Sensor review*, vol. 4, no. 3, pp. 120–124, 1984.

[189] M. Weeks, M. Freeman, A. Moulds, and J. Austin, "Developing hardware-based applications using PRESENCE-2," in *Perspectives in pervasive computing*, pp. 107–114, IET, 2005.

[190] D. Wang and D. Terman, "Image segmentation based on oscillatory correlation," *Neural Computation*, vol. 9, no. 4, pp. 805–836, 1997.

[191] R. Harrison, "A low-power analog VLSI visual collision detector," *Advances in Neural Information Processing Systems*, vol. 16, 2003.

[192] S. Bellis, K. M. Razeeb, C. Saha, K. Delaney, C. O'Mathuna, A. Pounds-Cornish, G. de Souza, M. Colley, H. Hagras, G. Clarke, *et al.*, "FPGA implementation of spiking neural networks-an initial step towards building tangible collaborative autonomous agents," in *Proceedings. 2004 IEEE International Conference on Field-Programmable Technology (IEEE Cat. No. 04EX921)*, pp. 449–452, IEEE, 2004.

[193] H. Li, D. Zhang, and S. Y. Foo, "A stochastic digital implementation of a neural network controller for small wind turbine systems," *IEEE Transactions on Power Electronics*, vol. 21, no. 5, pp. 1502–1507, 2006.

[194] L. M. Reyneri, M. Chiaberge, and L. Zocca, "CINTIA: A neuro-fuzzy real time controller for low power embedded systems," in *Proceedings of the*

Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, pp. 392–403, IEEE, 1994.

[195] M. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," *IEE Proceedings-Computers and Digital Techniques*, vol. 150, no. 6, pp. 403–411, 2003.

[196] K. Basterretxea, J. M. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," *IEE Proceedings-Circuits, Devices and Systems*, vol. 151, no. 1, pp. 18–24, 2004.

[197] A. H. Namin, K. Leboeuf, R. Muscedere, H. Wu, and M. Ahmadi, "Efficient hardware implementation of the hyperbolic tangent sigmoid function," in *2009 IEEE International Symposium on Circuits and Systems*, pp. 2117–2120, IEEE, 2009.

[198] L. Gatet, H. Tap-Béteille, and M. Lescure, "Analog neural network implementation for a real-time surface classification application," *IEEE Sensors Journal*, vol. 8, no. 8, pp. 1413–1421, 2008.

[199] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Analog implementation of a novel resistive-type sigmoidal neuron," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 4, pp. 750–754, 2011.

[200] J.-M. Sallese, M. Bucher, F. Krummenacher, and P. Fazan, "Inversion charge linearization in MOSFET modeling and rigorous derivation of the EKV compact model," *Solid-State Electronics*, vol. 47, no. 4, pp. 677–683, 2003.

[201] C. Galup-Montoro, M. C. Schneider, H. Klimach, and A. Arnaud, "A compact model of MOSFET mismatch for circuit design," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 8, pp. 1649–1657, 2005.

[202] A. Khakifirooz, O. M. Nayfeh, and D. Antoniadis, "A simple semiempirical short-channel MOSFET current–voltage model continuous across all regions of operation and employing only physical parameters," *IEEE Transactions on Electron Devices*, vol. 56, no. 8, pp. 1674–1680, 2009.

[203] J. P. Duarte, S. Khandelwal, A. Medury, C. Hu, P. Kushwaha, H. Agarwal, A. Dasgupta, and Y. S. Chauhan, "BSIM-CMG: Standard FinFET compact model for advanced circuit design," in *ESSCIRC Conference 2015-41st European Solid-State Circuits Conference (ESSCIRC)*, pp. 196–201, IEEE, 2015.

[204] F. A. Khanday, N. A. Kant, M. R. Dar, T. Z. A. Zulkifli, and C. Psychalinos, "Low-voltage low-power integrable cmos circuit implementation of integer- and fractional–order fitzhugh–nagumo neuron model," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 7, pp. 2108–2122, 2019.

[205] B. Razavi, *Design of analog CMOS integrated circuits*. Tsinghua University Press, 2005.

[206] M. Alioto, "Understanding DC behaviour of subthreshold CMOS logic through closed-form analysis," *IEEE Transactions on Circuits and Systems*, vol. 57, pp. 1597–1607, 2010.

[207] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreeto, and A. H., "Mobilenets: Efficient convolutional neural net-

works for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2016.

[208] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[209] H. H. Aghdam and E. J. Heravi, "Guide to convolutional neural networks," *New York, NY: Springer*, vol. 10, no. 978-973, p. 51, 2017.

[210] Nanoscale Integration and Modeling Group, Arizona State University, "Predictive Technology Model." http://ptm.asu.edu/, 2008.

[211] P. Daponte, D. Grimaldi, and L. Michaeli, "A full neural gray-code-based adc," *IEEE transactions on instrumentation and measurement*, vol. 45, no. 2, pp. 634–639, 1996.

[212] P. Pace, D. Styer, and I. Akin, "A folding adc preprocessing architecture employing a robust symmetrical number system with gray-code properties," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 462–467, 2000.

[213] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "Kan: Kolmogorov-arnold networks," 2024.

[214] Q. Xia and J. J. Yang, "Memristive crossbar arrays for brain-inspired computing," *Nature materials*, vol. 18, no. 4, pp. 309–323, 2019.

[215] M. A. Hanif, A. Manglik, and M. Shafique, "Resistive crossbar-aware neural network design and optimization," *IEEE Access*, vol. 8, pp. 229066–229085, 2020.

[216] C. Yakopcic, M. Z. Alom, and T. M. Taha, "Memristor crossbar deep net-

work implementation based on a convolutional neural network," in *2016 International joint conference on neural networks (IJCNN)*, pp. 963–970, IEEE, 2016.

[217] 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, p. 2331, 2018.

[218] X. Liu and Z. Zeng, "Memristor crossbar architectures for implementing deep neural networks," *Complex & Intelligent Systems*, vol. 8, no. 2, pp. 787–802, 2022.

[219] L. Chen, J. Li, Y. Chen, Q. Deng, J. Shen, X. Liang, and L. Jiang, "Accelerator-friendly neural-network training: Learning variations and defects in rram crossbar," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pp. 19–24, IEEE, 2017.

[220] J. Hur, Y.-C. Luo, A. Lu, T.-H. Wang, S. Li, A. I. Khan, and S. Yu, "Non-volatile capacitive crossbar array for in-memory computing," *Advanced Intelligent Systems*, vol. 4, no. 8, p. 2100258, 2022.

[221] S. Hong, H. Kang, J. Kim, and K. Cho, "Low voltage time-based matrix multiplier-and-accumulator for neural computing system," *Electronics*, vol. 9, no. 12, p. 2138, 2020.

[222] Y. Wang, H. Tang, Y. Xie, X. Chen, S. Ma, Z. Sun, Q. Sun, L. Chen, H. Zhu, J. Wan, *et al.*, "An in-memory computing architecture based on two-dimensional semiconductors for multiply-accumulate operations," *Nature communications*, vol. 12, no. 1, p. 3347, 2021.

[223] R. Li, L. Petit, D. P. Franke, J. P. Dehollain, J. Helsen, M. Steudtner, N. K. Thomas, Z. R. Yoscovits, K. J. Singh, S. Wehner, *et al.*, "A crossbar network for silicon quantum dot qubits," *Science advances*, vol. 4, no. 7, p. eaar3960, 2018.

[224] D. Tran and C. Teuscher, "Memcapacitive devices in logic and crossbar applications," *arXiv preprint arXiv:1704.05921*, 2017.

[225] Y.-C. Luo, A. Lu, J. Hur, S. Li, and S. Yu, "Design of non-volatile capacitive crossbar array for in-memory computing," in *2021 IEEE International Memory Workshop (IMW)*, pp. 1–4, IEEE, 2021.

[226] W. Kahan, "Ieee standard 754 for binary floating-point arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, no. 94720-1776, p. 11, 1996.

[227] W. Cody *et al.*, "Ieee standards 754 and 854 for floating-point arithmetic," *The IEEE Magazine MICRO*, pp. 84–100, 1984.

[228] R. Tibshirani, "A comparison of some error estimates for neural network models," *Neural computation*, vol. 8, no. 1, pp. 152–163, 1996.

[229] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on mnist," in *Seventh International Conference on Learning Representations (ICLR 2019)*, pp. 1–16, 2019.

[230] E. M. El Mhamdi, R. Guerraoui, and S. L. A. Rouault, "On the robustness of a neural network," in *2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*, pp. 84–93, 2017.

[231] R. S. Chen, B. Lucier, Y. Singer, and V. Syrgkanis, "Robust optimization for non-convex objectives," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[232] S. Webb, T. Rainforth, Y. Teh, and P. Mudigonda, "A statistical approach to assessing neural network robustness," in *Seventh International Conference on Learning Representations (ICLR 2019)*, International Conferences on Learning Representations, 2019.

[233] Y. Ding, W. Jiang, Q. Lou, J. Liu, J. Xiong, X. S. Hu, X. Xu, and Y. Shi, "Hardware design and the competency awareness of a neural network," *Nature Electronics*, vol. 3, no. 9, pp. 514–523, 2020.

[234] R. Mangal, A. V. Nori, and A. Orso, "Robustness of neural networks: A probabilistic and practical approach," in *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 93–96, IEEE, 2019.

[235] S.-i. Yi, J. D. Kendall, R. S. Williams, and S. Kumar, "Activity-difference training of deep neural networks using memristor crossbars," *Nature Electronics*, vol. 6, no. 1, pp. 45–51, 2023.

[236] Y. Yamagishi, T. Kaneko, M. Akai-Kasaya, and T. Asai, "Holmes: A hardware-oriented optimizer using logarithms," *IEICE TRANSACTIONS on Information and Systems*, vol. 105, no. 12, pp. 2040–2047, 2022.

[237] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, no. 1, p. 13276, 2016.

[238] A. Nøkland, "Direct feedback alignment provides learning in deep neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[239] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simu-

lated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[240] L. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Procedia Computer Science*, vol. 72, pp. 137–144, 2015.

[241] M. Creutz, "Microcanonical monte carlo simulation," *Physical Review Letters*, vol. 50, no. 19, p. 1411, 1983.

[242] G. Dueck and T. Scheuer, "Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing," *Journal of computational physics*, vol. 90, no. 1, pp. 161–175, 1990.

[243] I. Charon and O. Hudry, "The noising method: a new method for combinatorial optimization," *Operations Research Letters*, vol. 14, no. 3, pp. 133–137, 1993.

[244] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[245] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information sciences*, vol. 237, pp. 82–117, 2013.

[246] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *Advances in neural information processing systems*, vol. 30, 2017.

[247] M. Stern and A. Murugan, "Learning without neurons in physical systems," *Annual Review of Condensed Matter Physics*, vol. 14, pp. 417–441, 2023.

[248] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology press, 2005.

[249] M. S. Thomas and J. L. McClelland, "Connectionist models of cognition," *The Cambridge handbook of computational psychology*, pp. 23–58, 2008.

[250] T. Miconi, "Hebbian learning with gradients: Hebbian convolutional neural networks with modern deep learning frameworks," *arXiv preprint arXiv:2107.01729*, 2021.

[251] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[252] J. L. McClelland, A. G. Thomas, B. D. McCandliss, and J. A. Fiez, "Understanding failures of learning: Hebbian learning, competition for representational space, and some preliminary experimental data," *Progress in brain research*, vol. 121, pp. 75–80, 1999.

[253] D. Zipser and R. A. Andersen, "A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons," *Nature*, vol. 331, no. 6158, pp. 679–684, 1988.

[254] J. C. Whittington and R. Bogacz, "Theories of error back-propagation in the brain," *Trends in cognitive sciences*, vol. 23, no. 3, pp. 235–250, 2019.

[255] R. C. O'Reilly, "Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm," *Neural computation*, vol. 8, no. 5, pp. 895–938, 1996.

[256] K. Friston, "The free-energy principle: a unified brain theory?," *Nature reviews neuroscience*, vol. 11, no. 2, pp. 127–138, 2010.

[257] J. C. Whittington and R. Bogacz, "An approximation of the error back-propagation algorithm in a predictive coding network with local hebbian

synaptic plasticity," *Neural computation*, vol. 29, no. 5, pp. 1229–1262, 2017.

[258] F. A. Mikulasch, L. Rudelt, M. Wibral, and V. Priesemann, "Where is the error? hierarchical predictive coding through dendritic error computation," *Trends in Neurosciences*, vol. 46, no. 1, pp. 45–59, 2023.

[259] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, 2022.

[260] Y. Abadade, A. Temouden, H. Bamoumen, N. Benamar, Y. Chtouki, and A. S. Hafid, "A comprehensive survey on tinyml," *IEEE Access*, 2023.

[261] R. Sanchez-Iborra and A. F. Skarmeta, "Tinyml-enabled frugal smart objects: Challenges and opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, no. 3, pp. 4–18, 2020.

[262] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, *et al.*, "Description and discussion on dcase2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," *arXiv preprint arXiv:2006.05822*, 2020.

[263] P. P. Jayaraman, A. Yavari, D. Georgakopoulos, A. Morshed, and A. Zaslavsky, "Internet of things platform for smart farming: Experiences and lessons learnt," *Sensors*, vol. 16, no. 11, p. 1884, 2016.

[264] S. O. Ooko, M. M. Ogore, J. Nsenga, and M. Zennaro, "Tinyml in africa: Opportunities and challenges," in *2021 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–6, IEEE, 2021.

[265] C. D. Căleanu, C. L. Sîrbu, and G. Simion, "Deep neural architectures for contrast enhanced ultrasound (ceus) focal liver lesions automated diagnosis," *Sensors*, vol. 21, no. 12, p. 4126, 2021.

[266] P. Randhawa, V. Shanthagiri, and A. Kumar, "A review on applied machine learning in wearable technology and its applications," in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, pp. 347–354, IEEE, 2017.

[267] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov, *et al.*, "Benchmarking tinyml systems: Challenges and direction," *arXiv preprint arXiv:2003.04821*, 2020.

[268] M. Quigley and M. Burke, "Low-cost internet of things digital technology adoption in smes," *International Journal of Management Practice*, vol. 6, no. 2, pp. 153–164, 2013.

[269] L. Dutta and S. Bharali, "Tinyml meets iot: A comprehensive survey," *Internet of Things*, vol. 16, p. 100461, 2021.

[270] W. Niu, X. Ma, S. Lin, S. Wang, X. Qian, X. Lin, Y. Wang, and B. Ren, "Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning," in *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 907–922, 2020.

[271] G. Meijer, K. Makinwa, and M. Pertijs, *Smart sensor systems: Emerging technologies and applications*. John Wiley & Sons, 2014.

[272] Z. Xu, T. Zhou, M. Ma, C. Deng, Q. Dai, and L. Fang, "Large-scale pho-

tonic chiplet taichi empowers 160-tops/w artificial general intelligence," *Science*, vol. 384, no. 6692, pp. 202–209, 2024.

[273] V. G. Costa and C. E. Pedreira, "Recent advances in decision trees: An updated survey," *Artificial Intelligence Review*, vol. 56, no. 5, pp. 4765–4800, 2023.

[274] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[275] M. Elsisi, K. Mahmoud, M. Lehtonen, and M. M. Darwish, "Reliable industry 4.0 based on machine learning and iot for analyzing, monitoring, and securing smart meters," *Sensors*, vol. 21, no. 2, p. 487, 2021.

[276] T. Rusch and A. Zeileis, "Discussion of" 50 years of classification and regression trees"," *International Statistical Review*, vol. 82, pp. 361–367, 2014.

[277] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature machine intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[278] R. Balestriero, "Neural decision trees," *arXiv preprint arXiv:1702.07360*, 2017.

[279] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," *arXiv preprint arXiv:1806.06988*, 2018.

[280] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo, "Deep neural decision forests," in *Proceedings of the IEEE international conference on*

computer vision, pp. 1467–1475, 2015.

[281] S. Goswami, R. Pramanick, A. Patra, S. P. Rath, M. Foltin, A. Ariando, D. Thompson, T. Venkatesan, S. Goswami, and R. S. Williams, "Decision trees within a molecular memristor," *Nature*, vol. 597, no. 7874, pp. 51–56, 2021.

[282] H. D. Nguyen and F. Chamroukhi, "Practical and theoretical aspects of mixture-of-experts modeling: An overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1246, 2018.

[283] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: a literature survey," *Artificial Intelligence Review*, vol. 42, pp. 275–293, 2014.

List of terms

FLOPS

floating-point operation per second

MOSFET

metal oxide semiconductor field-effect transistor

OPS/W

operation per second per Watt

ReLU6

Rectified Linear Unit 6

ReLU

Rectified Linear Unit

activation function

A function that calculates the output of a node in an artificial neural network based on its individual inputs and their weights in machine learning

ADC

analog-digital converter

AFC

activation function circuit

ALU

arithmetic logic unit

ANN

artificial neural network

AR

augmented reality

artificial intelligence

A field of research in computer science that develops and studies methods and software which enable machines to perceive their environment and uses learning and intelligence to take actions that maximize their chances of achieving defined goals.

ASIC

application-specific integrated circuit

CE

cross-entropy

CMOS

complementary metal oxide semiconductor

CNN

convolutional neural network

CPPS

connection primitives per second

CPU

central processing unit

DRAM

dynamic random-access memory

DC

direct current

DIBL

drain induced barrier lowering

DNN

deep neural network

DPDT

double Pole double throw

DSP

digital signal processor

DT

decision tree

EMF

electromotive force

FET

field effect transistor

FLOP

floating-point operation

Fourier transform

A transform taking a function as input and a function as output describing the extent to which various frequencies are present in the origin function.

FPGA

field-programmable gate array

global Lipschitz constant

The bound of a given neural network's output on the maximum rate of change over its entire input space.

GNN

graph neural network

GPU

graphics processing unit

gradient descent

An optimization algorithm for finding the local minimum of a differentiable function.

HNN

hardware neural network

I/O

input / output

IC

integrated circuit

IoT

Internet of things

Kirchoff's current law

The algebraic sum of currents in a network of conductors meeting at a point is zero.

Kirchoff's voltage law

The directed total electromotive force (EMF) around any closed loop is zero.

Lagrange's mean value theorem

For a given planar arc between two endpoints, there is at least one point at which the tangent to the arc is parallel to the secant through its endpoints.

LBCNN

local binary convolutional neural network

loss function

A function that maps an event or values of one or more variables onto a real number intuitively representing some cost associated with the event in mathematical optimization.

LSB

least significant bit

LSTM

long short-term memory network

LUT

look-up table

MAC

multiply accumulate circuit

ML

machine learning

MLP

multi-layer perceptron

MNIST

Modified National Institute of Science and Technology

MoE

mixture of experts

Monte-Carlo method

An algorithm using randomness to solve problems that might be deterministic in principle rely on repeated random sampling to obtain numerical results.

MOS

metal oxide semiconductor

MSB

most significant bit

MSE

mean square error

MUX

multiplexer

N-MOS

N-Channel metal oxide semiconductor

NDT

neural decision tree

neural network

A model inspired by the neuron organization found in the biological nerve system in animal brains in machine learning.

NLP

natural language processing

NP

non-deterministic polynomial

NP-Hard

non-deterministic polynomial-hard

Op-Amp

operational amplifier

OPS

operation per second

P-MOS

P-Channel metal oxide semiconductor

PDF

probability density function

PIM

processing-in-memory

PTM

predictive technology model

PWL

piece-wise linear

RAM

random-access memory

ResNet

residual neural network

RMS

root mean square

RNN

recursive neural network

S/H

sample-and-hold

SRAM

static random-access memory

SGD

stochastic gradient descent

SI

Système International d'Unités

SNN

spiking neural network

SNR

signal-to-noise ratio

SoC

systems on a chip

SOI

silicon on insulator

SOTA

state-of-the-art

SPICE

simulation program with integrated circuit emphasis

Taylor expansion

A method to present a function by an infinite sum of terms expressing the function's derivations at a given point.

TinyML

tiny machine learning

transfer function

A function that models the output of a system for each possible input in engineering.

Turing complete

Having the ability to simulate any Turing Machines.

VMM

vector-matrix multiplication

Appendix A

Introduction of Neural Networks

A neural network represents a sophisticated computational framework that is inspired by the structure and function of the biological nervous system. It is comprised of a multitude of interconnected neurons (or nodes) that facilitate the transmission of signals through weighted connections. These neurons employ activation functions to non-linearly transform data, thereby emulating the human brain's information processing capabilities. Neural networks possess the ability to learn and adjust, enabling them to execute complex tasks encompassing classification, regression, clustering, and reinforcement learning.

A.1 Structural Composition of Neural Networks

Neural networks, known as a deep learning model, is typically composed of multiple interconnected layers that collaboratively extract features from data and execute specific tasks, such as classification, regression, or data generation.

Neural networks generally consist of three primary types of layers: the input layer, hidden layer, and output layer. The input layer serves to receive raw

data, the hidden layer is responsible for processing this data and extracting pertinent features, while the output layer generates the final prediction or decision. Each layer comprises multiple neurons interconnected via weights, which are progressively fine-tuned throughout the training process.

A.1.1 Input Layer

The input layer serves as the initial layer of the model responsible for receiving the raw input data. In image recognition applications, this layer is generally a two-dimensional convolutional layer configured to process image pixel data. Conversely, in natural language processing (NLP) tasks, the input layer may consist of an embedding layer that converts text into numerical vector representations.

A.1.2 Hidden Layers

Hidden layers are situated between the input layer and output layer, playing a crucial role in extracting high-level features from the data. These layers can be further classified into several types, including:

- Fully Connected Layer: Positioned between hidden layers or at the conclusion of the neural network, it utilises the activation values from the preceding layer as inputs for classification or other tasks.
- Convolutional Layer: utilises filters or kernels to slide over the input data, thereby extracting local features. This layer is extensively employed in computer vision applications.
- Pooling Layer: Operates to reduce the spatial dimensions of the feature

map, diminishes computational load while preserving essential information. Typical pooling operations include Max Pooling and Average Pooling.

- Activation Layer: Introduces non-linear transformations to enable the model to learn complex function mappings. Common activation functions include *Rectified Linear Unit* (*ReLU*), *Sigmoid*, and $\tanh$.

A.1.3 Output Layer

The output layer represents the final component of the model, and its architecture is tailored to the specific task at hand. In classification tasks, this layer typically incorporates **softmax** activation functions to generate a probability distribution. In regression tasks, the output layer may produce continuous value predictions directly.

A.1.4 Layer Stacking

Layer stacking pertains to the strategic arrangement of various types of layers in a specific order to form a comprehensive network architecture. The output generated by each layer typically serves as the input for the subsequent layer. During the stacking process, each layer extracts and transforms features based on the preceding layer, progressively constructing a complex feature representation.

In convolutional neural networks (CNNs), layer stacking generally follows a progression from low-level features to high-level abstractions. The convolutional and pooling layers alternate to systematically decrease the spatial dimen-

sions of the feature map while enhancing the feature abstraction level. The fully connected layer is customarily positioned at the network's end, integrating the features across all levels and facilitating the final decision-making process.

In recursive neural networks (RNNs) and Transformer models, layer stacking emphasizes the processing of sequential data, adeptly capturing dependencies within the sequence through cyclic connections or self-attention mechanisms.

In the design of deep learning models, the selection of appropriate layer types and stacking strategies is paramount for optimizing the model's performance. By methodically adjusting the network architecture, it is feasible to identify the optimal combination of layers and types to address a specific task effectively.

A.2 Principle of Model Fitting

The back-propagation algorithm serves as a foundational learning mechanism utilised for training artificial neural networks. This algorithm meticulously updates the network weights by computing the gradient of the loss function concerning these weights, with the objective of minimizing prediction errors.

A.2.1 Training Process of Neural Networks

The training process of neural networks encompasses the following pivotal steps:

1. Data Pre-processing: Prior to initiating training, data typically undergoes normalization or standardization procedures to ensure the network can ef-

fectively learn the essential features.

2. Network Parameter Initialization: Weights and biases are generally initialized randomly to disrupt symmetry and facilitate the network's ability to learn diverse features.
3. Forward Propagation: Input data traverses the network, with neurons at each layer calculating their activation values based on their activation functions and the outputs from the preceding layer.
4. Loss Calculation: The network's predicted output is compared against the actual label to derive the value of the loss function, which serves as an indicator of prediction accuracy.
5. Back-propagation: The error is back-propagated through the network layers based on the loss function's gradient, enabling the calculation of gradients for the weights and biases of neurons in each layer.
6. Parameter Update: Optimization algorithms such as gradient descent are utilised to dynamically update network parameters in alignment with the gradients to minimize losses.
7. Iterative Training: The cyclical sequence involving forward propagation, loss calculation, back-propagation, and parameter update is reiterated until the convergence conditions are satisfied or a predefined number of training cycles is achieved.
8. Model Evaluation: Validation or test sets are utilised to assess the network's performance, allowing for hyper-parameter adjustments aimed at model optimisation.
9. Model Tuning: Based on evaluation findings, modifications to the net-

work's structure, learning rate, batch size, and other hyper-parameters may be necessary to enhance the model's generalization capabilities.

In practical applications, training neural networks may also necessitate the implementation of strategies to mitigate over-fitting, which may include the adoption of regularization techniques, dropout methods, or data augmentation practices.

A.2.2 Fitting Principle of Neural Networks

The fitting principle of neural networks is fundamentally based on the error back-propagation algorithm. During the forward propagation phase, input data is transmitted through each layer of the network, where the neurons within each layer compute activation values derived from their respective inputs and weights. This process continues until the output layer generates a predicted result. Should there be a discrepancy between this output and the actual label, the resultant error is utilised to modify the weights and biases within the network.

The sequential workflow of the error back-propagation algorithm is delineated as follows:

1. Output Layer Error Calculation: Initially, the error between the predicted value from the output layer and the actual label must be computed, typically quantified using a loss function.
2. Gradient Calculation: The chain rule is applied to ascertain the error gradients corresponding to each neuron's weight and bias in the network. The gradient indicates the rate of error change relative to the respective weights and biases, thereby guiding the network in adjusting these parameters to

minimize the error.

3. Weight and Bias Update: The optimisation algorithms (such as gradient descent) are employed to update the network's weights and biases based on the previously calculated gradients. The update rule generally involves subtracting the product of the learning rate and the gradient from the current parameters to facilitate movement towards reduced error.
4. Iterative Optimization: This entire process is iteratively executed across the training set until the network's output aligns sufficiently with the actual label or until a pre-established maximum number of iterations is reached.

Central to the back-propagation algorithm is the error back-propagation mechanism, which initializes the output calculation via forward propagation and subsequently refines the network weights through gradient descent in response to the errors identified in the output.

The Taylor expansion represents a significant mathematical tool that facilitates the approximation of a function's value by expanding it into an infinite series around a designated point. In the context of optimisation challenges, the Taylor expansion is frequently deployed to estimate the objective function, thus aiding in identifying its minimum during iterative processes. The Newton iteration method, which is predicated on the Taylor expansion, employs both the first and second derivatives of a function—the gradient and Hessian matrix—to formulate a quadratic function. The vertices of this quadratic representation are then leveraged as approximations for the optimal solution in subsequent iterations.

The relationship between the back-propagation algorithm and both the Taylor expansion and Newton iteration method lies in their utilization of deriva-

tive information to streamline the search for optimal solutions. Specifically, in the back-propagation algorithm, weight updates are achieved through the calculation of the gradient of the loss function, akin to the first-order approximations found in the Taylor expansion. The Newton iteration method enhances this process by incorporating the Hessian matrix, allowing for the establishment of second-order approximations. This refinement facilitates a more rapid convergence towards the optimal solution by considering the function's curvature. In various optimisation strategies, notably quasi-Newton methods, direct computation of the Hessian matrix is not performed; however, these methods still contribute to expediting convergence through matrix approximation updates, demonstrating notable efficacy in the training of neural networks.

In practical implementations, the back-propagation algorithm is extensively adopted for neural network training due to its comparatively straightforward and user-friendly execution. While both the Newton iteration method and quasi-Newton approaches exhibit rapid convergence, they are not as commonly employed as the back-propagation algorithm within the realm of neural network training. Nonetheless, these methodologies may prove more advantageous for large-scale optimisation problems, particularly in scenarios where the Hessian matrix is sparse or nearly positive definite.

A.3 Development of Neural Networks

A.3.1 Perceptron

Perceptrons represent the foundational structure of neural networks, initially introduced by Frank Rosenblatt in 1957. They consist of one or multiple input nodes, a single output node, and a threshold component. Mathematically, the perceptron can be articulated as a linear function augmented by an activation functions, typically represented as a step function. These models are adept at learning linearly separable decision boundaries, making them appropriate for straightforward binary classification tasks.

A.3.2 Multi-layer Perceptron

The multi-layer perceptron (MLP) builds upon the perceptron framework by incorporating at least one hidden layer, each containing multiple neurons. MLPs are distinguished by their use of non-linear activation functions, allowing the network to learn and approximate complex non-linear relationships effectively. The architecture of an MLP comprises an input layer, one or more hidden layers, and an output layer, with neurons interlinked by weighted connections across layers.

A.3.3 Convolutional Neural Network

Convolutional neural networks (CNNs) are advanced deep learning models specifically designed for the analysis of two-dimensional data, prominently used in image and video processing applications. CNNs extract image features

through local connections and weight sharing within convolutional layers, complemented by spatial reduction techniques and pooling operations. These networks proficiently process the spatial hierarchies inherent in images, reducing model parameters while enhancing computational efficiency and generalization capabilities.

A.3.4 Recursive Neural Network

Recursive neural networks (RNNs) represent a unique class of architectures designed for sequential data processing, such as text and time series. RNNs are characterized by their internal feedback loops, enabling the capture of temporal dynamics within sequential data. However, traditional RNNs can be hindered by issues of vanishing and exploding gradients, which impede their performance on long-distance dependencies.

A.3.5 Long Short-Term Memory Network and Gated Recurrent Unit

long short-term memory networks (LSTMs) and Gated Recurrent Units are innovative adaptations of RNNs that address the limitations of traditional configurations through the introduction of gating mechanisms. LSTMs manage information flow via three distinct gates—forget gate, input gate, and output gate—while GRUs streamline the architecture with only two gates: update gate and reset gate. These models are efficient in learning and retaining long-range dependent information.

A.4 Range of Applications

Neural networks encompass a diverse spectrum of applications, including but not limited to:

- Computer Vision: Encompasses image recognition, object detection, facial recognition, etc.
- Natural Language Processing: Encompasses machine translation, sentiment analysis, text generation, etc.
- Speech Recognition: Refers to automatic speech recognition systems.
- Gaming: Involves the application of reinforcement learning in sectors such as Go, video gaming, etc.
- Medical Diagnosis: Encompasses disease prediction and medical image analysis.
- Autonomous Driving: Pertains to environmental perception and decision-making.
- Recommendation Systems: Involves personalized content recommendations.

For instance, GoogleNet (Inception v1) exemplifies a deep CNN that achieves multi-scale feature extraction through the innovative design of the Inception module. This module encompasses convolutional kernels of varying sizes and employs 1×1 convolutions to minimize parameter count while enhancing non-linearity. These modules can be stacked to create a profound network structure that effectively captures diverse features of an image.

In the Transformer model, the multi-head self-attention mechanism permits the model to consider global dependencies among individual elements within

a sequence during processing. This model facilitates efficient sequence processing by stacking multiple identical encoder and decoder layers, each comprising self-attention and feed-forward neural networks.

As technological advancements continue, the application domains of neural networks are rapidly expanding, accompanied by the emergence of novel architectures and training methodologies.

Appendix B

Low Power *45nm* MOS FET PTM

* PTM Low Power 45nm Metal Gate / High-K / Strained-Si

* nominal Vdd = 1.1V

.model nmos nmos level = 54			
+version = 4.0	binunit = 1	paramchk= 1	mobmod = 0
+capmod = 2	igcmod = 1	igbmod = 1	geomod = 1
+diomod = 1	rdsmod = 0	rbodymod= 1	rgatemod= 1
+permod = 1	acnqsmod= 0	trnqsmod= 0	
+tnom = 27	toxe = 1.8e-009	toxp = 1.5e-009	toxm = 1.8e-009
+dtox = 3e-010	epsrox = 3.9	wint = 5e-009	lint = 0
+ll = 0	wl = 0	lln = 1	wln = 1
+lw = 0	ww = 0	lwn = 1	wwn = 1
+lwl = 0	wwl = 0	xpart = 0	toxref = 1.8e-009
+vth0 = 0.42261	k1 = 0.4	k2 = 0	k3 = 0
+k3b = 0	w0 = 2.5e-006	dvt0 = 1	dvt1 = 2
+dvt2 = 0	dvt0w = 0	dvt1w = 0	dvt2w = 0

+dsub = 0.1	minv = 0.05	voffl = 0	dvtp0 = 1e-010
+dvtp1 = 0.1	lpe0 = 0	lpeb = 0	xj = 1.4e-008
+ngate = 1e+023	ndep = 3.24e+018	nsd = 2e+020	phin = 0
+cdsc = 0	cdscb = 0	cdscd = 0	cit = 0
+voff = -0.13	nfactor = 1.6	eta0 = 0.0125	etab = 0
+vfb = -0.55	u0 = 0.049	ua = 6e-010	ub = 1.2e-018
+uc = 0	vsat = 130000	a0 = 1	ags = 0
+a1 = 0	a2 = 1	b0 = 0	b1 = 0
+keta = 0.04	dwg = 0	dwb = 0	pclm = 0.02
+pdiblc1 = 0.001	pdiblc2 = 0.001	pdiblcb = -0.005	drout = 0.5
+pvag = 1e-020	delta = 0.01	pscbe1 = 8.14e+008	pscbe2 = 1e-007
+fprout = 0.2	pdits = 0.08	pditsd = 0.23	pditsl = 2300000
+rsh = 5	rdsw = 210	rsw = 80	rdw = 80
+rdswmin = 0	rdwmin = 0	rswmin = 0	prwg = 0
+prwb = 0	wr = 1	alpha0 = 0.074	alpha1 = 0.005
+beta0 = 30	agidl = 0.0002	bgidl = 2.1e+009	cgidl = 0.0002
+egidl = 0.8	aigbacc = 0.012	bigbacc = 0.0028	cigbacc = 0.002
+nigbacc = 1	aigbinv = 0.014	bigbinv = 0.004	cigbinv = 0.004
+eigbinv = 1.1	nigbinv = 3	aigc = 0.015211	bigc = 0.0027432
+cigc = 0.002	aigsd = 0.015211	bigsd = 0.0027432	cigsd = 0.002
+nigc = 1	poxedge = 1	pigcd = 1	ntox = 1
+xrcrg1 = 12	xrcrg2 = 5		

+cgso = 1.1e-010	cgdo = 1.1e-010	cgbo = 2.56e-011	cgdl = 2.653e-010
+cgsl = 2.653e-010	ckappas = 0.03	ckappad = 0.03	acde = 1

+moin = 15	noff = 0.9	voffcv = 0.02	
+kt1 = -0.11	kt1l = 0	kt2 = 0.022	ute = -1.5
+ua1 = 4.31e-009	ub1 = 7.61e-018	uc1 = -5.6e-011	prt = 0
+at = 33000			
+fnoimod = 1	tnoimod = 0		
+jss = 0.0001	jsws = 1e-011	jswgs = 1e-010	njs = 1
+ijthsfwd= 0.01	ijthsrev= 0.001	bvs = 10	xjbvs = 1
+jsd = 0.0001	jswd = 1e-011	jswgd = 1e-010	njd = 1
+ijthdfwd= 0.01	ijthdrev= 0.001	bvd = 10	xjbvd = 1
+pbs = 1	cjs = 0.0005	mjs = 0.5	pbsws = 1
+cjsws = 5e-010	mjsws = 0.33	pbswgs = 1	cjswgs = 3e-010
+mjswgs = 0.33	pbd = 1	cjd = 0.0005	mjd = 0.5
+pbswd = 1	cjswd = 5e-010	mjswd = 0.33	pbswgd = 1
+cjswgd = 5e-010	mjswgd = 0.33	tpb = 0.005	tcj = 0.001
+tpbsw = 0.005	tcjsw = 0.001	tpbswg = 0.005	tcjswg = 0.001
+xtis = 3	xtid = 3		
+dmcg = 0	dmci = 0	dmdg = 0	dmcgt = 0
+dwj = 0	xgw = 0	xgl = 0	
+rshg = 0.4	gbmin = 1e-010	rbpb = 5	rbpd = 15
+rbps = 15	rbdb = 15	rbsb = 15	ngcon = 1

.model pmos pmos level = 54			
+version = 4.0	binunit = 1	paramchk= 1	mobmod = 0
+capmod = 2	igcmod = 1	igbmod = 1	geomod = 1

+diomod = 1	rdsmod = 0	rbodymod= 1	rgatemod= 1
+permod = 1	acnqsmod= 0	trnqsmod= 0	
+tnom = 27	toxe = 1.82e-009	toxp = 1.5e-009	toxm = 1.82e-009
+dtox = 3.2e-010	epsrox = 3.9	wint = 5e-009	lint = 0
+ll = 0	wl = 0	lln = 1	wln = 1
+lw = 0	ww = 0	lwn = 1	wwn = 1
+lwl = 0	wwl = 0	xpart = 0	toxref = 1.82e-009
+vth0 = -0.42661	k1 = 0.4	k2 = -0.01	k3 = 0
+k3b = 0	w0 = 2.5e-006	dvt0 = 1	dvt1 = 2
+dvt2 = -0.032	dvt0w = 0	dvt1w = 0	dvt2w = 0
+dsub = 0.1	minv = 0.05	voffl = 0	dvtp0 = 1e-011
+dvtp1 = 0.05	lpe0 = 0	lpeb = 0	xj = 1.4e-008
+ngate = 1e+023	ndep = 2.44e+018	nsd = 2e+020	phin = 0
+cdsc = 0	cdscb = 0	cdscd = 0	cit = 0
+voff = -0.126	nfactor = 1.8	eta0 = 0.0125	etab = 0
+vfb = 0.55	u0 = 0.021	ua = 2e-009	ub = 5e-019
+uc = 0	vsat = 90000	a0 = 1	ags = 1e-020
+a1 = 0	a2 = 1	b0 = 0	b1 = 0
+keta = -0.047	dwg = 0	dwb = 0	pclm = 0.12
+pdiblc1 = 0.001	pdiblc2 = 0.001	pdiblcb = 3.4e-008	drout = 0.56
+pvag = 1e-020	delta = 0.01	pscbe1 = 8.14e+008	pscbe2 = 9.58e-007
+fprout = 0.2	pdits = 0.08	pditsd = 0.23	pditsl = 2300000
+rsh = 5	rdsw = 250	rsw = 75	rdw = 75
+rdswmin = 0	rdwmin = 0	rswmin = 0	prwg = 0

+prwb = 0	wr = 1	alpha0 = 0.074	alpha1 = 0.005
+beta0 = 30	agidl = 0.0002	bgidl = 2.1e+009	cgidl = 0.0002
+egidl = 0.8	aigbacc = 0.012	bigbacc = 0.0028	cigbacc = 0.002
+nigbacc = 1	aigbinv = 0.014	bigbinv = 0.004	cigbinv = 0.004
+eigbinv = 1.1	nigbinv = 3	aigc = 0.0097	bigc = 0.00125
+cigc = 0.0008	aigsd = 0.0097	bigsd = 0.00125	cigsd = 0.0008
+nigc = 1	poxedge = 1	pigcd = 1	ntox = 1
+xrcrg1 = 12	xrcrg2 = 5		
+cgso = 1.1e-010	cgdo = 1.1e-010	cgbo = 2.56e-011	cgdl = 2.653e-010
+cgsl = 2.653e-010	ckappas = 0.03	ckappad = 0.03	acde = 1
+moin = 15	noff = 0.9	voffcv = 0.02	
+kt1 = -0.11	kt1l = 0	kt2 = 0.022	ute = -1.5
+ua1 = 4.31e-009	ub1 = 7.61e-018	uc1 = -5.6e-011	prt = 0
+at = 33000			
+fnoimod = 1	tnoimod = 0		
+jss = 0.0001	jsws = 1e-011	jswgs = 1e-010	njs = 1
+ijthsfwd= 0.01	ijthsrev= 0.001	bvs = 10	xjbvs = 1
+jsd = 0.0001	jswd = 1e-011	jswgd = 1e-010	njd = 1
+ijthdfwd= 0.01	ijthdrev= 0.001	bvd = 10	xjbvd = 1
+pbs = 1	cjs = 0.0005	mjs = 0.5	pbsws = 1
+cjsws = 5e-010	mjsws = 0.33	pbswgs = 1	cjswgs = 3e-010
+mjswgs = 0.33	pbd = 1	cjd = 0.0005	mjd = 0.5
+pbswd = 1	cjswd = 5e-010	mjswd = 0.33	pbswgd = 1
+cjswgd = 5e-010	mjswgd = 0.33	tpb = 0.005	tcj = 0.001

+tpbsw = 0.005	tcjsw = 0.001	tpbswg = 0.005	tcjswg = 0.001
+xtis = 3	xtid = 3		
+dmcg = 0	dmci = 0	dmdg = 0	dmcgt = 0
+dwj = 0	xgw = 0	xgl = 0	
+rshg = 0.4	gbmin = 1e-010	rbpb = 5	rbpd = 15
+rbps = 15	rbdb = 15	rbsb = 15	ngcon = 1