UNIVERSIDADE FEDERAL DO AMAZONAS – UFAM

INSTITUTO DE COMPUTAÇÃO – ICOMP

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA – PPGI

An Automated Method for Neural Network Quantization Refinement

João Batista Pereira Matos Júnior

Manaus, Brasil

2025

João Batista Pereira Matos Júnior

An Automated Method for Neural Network Quantization Refinement

Tese de doutorado apresentada ao Programa de Pós-Graduação em Informática, da Universidade Federal do Amazonas, como requisito parcial para a obtenção do título de Doutor em Informática, na Área de concentração em Sistemas Embarcados e Engenharia de Software.

Supervisor: Lucas Carvalho Cordeiro

Co-supervisor: Eddie Batista de Lima Filho

Manaus, Brasil

2025

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

M433a Matos Júnior, João Batista Pereira
An Automated Method for Neural Network Quantization Refinement / João Batista Pereira Matos Júnior. - 2025.
130 f. : il., color. ; 31 cm.

Orientador(a): Lucas Carvalho Cordeiro.
Coorientador(a): Eddie Batista de Lima Filho.
Tese (doutorado) - Universidade Federal do Amazonas, Programa de Pós-Graduação em Informática, Manaus, 2025.

1. Redes Neurais. 2. Quantização. 3. Verificação Formal de Equivalência. 4. Equivalência Funcional. 5. Sistemas Críticos para Segurança. I. Cordeiro, Lucas Carvalho. II. Batista de, Eddie. III. Universidade Federal do Amazonas. Programa de Pós-Graduação em Informática. IV. Título

---

Ministério da Educação
Universidade Federal do Amazonas
Coordenação do Programa de Pós-Graduação em Informática

FOLHA DE APROVAÇÃO

"AN AUTOMATED METHOD FOR NEURAL NETWORK QUANTIZATION REFINEMENT"

JOÃO BATISTA PEREIRA MATOS

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Lucas Carvalho Cordeiro - **Presidente**

Prof. Dr. Thierson Couto Rosa - **Membro Externo**

Prof. Dr. Leandro Buss Becker - **Membro Externo**

Prof. Dr. Raimundo da Silva Barreto - **Membro Interno**

Dr. Eddie Batista de Lima Filho - **Membro Externo**

Manaus, 18 de fevereiro de 2025.

Documento assinado eletronicamente por **Eddie Batista de Lima Filho**, **Usuário Externo**, em 19/02/2025, às 13:36, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Leandro Buss Becker**, **Usuário Externo**, em 19/02/2025, às 15:26, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Lucas Carvalho Cordeiro**, **Professor do Magistério Superior**, em 19/02/2025, às 15:37, conforme horário oficial de Manaus, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufam.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2462510** e o código CRC **7133BC05**.

Avenida General Rodrigo Octávio, 6200 - Bairro Coroado I Campus Universitário Senador Arthur Virgílio Filho, Setor Norte - Telefone: (92) 3305-1181 / Ramal 1193
CEP 69080-900, Manaus/AM, coordenadorppgi@icomp.ufam.edu.br

Referência: Processo nº 23105.005926/2025-85 SEI nº 2462510

This work is dedicated to my wife Geovanna, my family, and my friends for their support and understanding throughout this journey of study.

Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work was partially supported by Amazonas State Research Support Foundation - FAPEAM - through the POSGRAD project.

"Everything should be made as simple as possible, but no simpler."
(Albert Einstein)

Abstract

In the rapidly evolving field of artificial intelligence, deep neural networks (DNNs) have become integral to numerous applications, from autonomous vehicles to healthcare systems. However, the increasing size and complexity of DNNs pose significant challenges for deployment in resource-constrained and safety-critical environments due to high computational and memory demands. This dissertation addresses the critical problem of neural network quantization, which aims to reduce the precision of network parameters to enhance efficiency while preserving functional behavior. The primary contribution is the development of the Counterexample-Guided Quantization for Neural Networks (CEG4N) framework, which integrates formal equivalence verification (FEV)—a technique that rigorously verifies that two models exhibit identical behavior—into the quantization process. Methodologically, CEG4N employs an iterative optimization loop that leverages formal verification tools to systematically detect and correct discrepancies between the original and quantized models, thus ensuring their functional equivalence (FE). Experimental evaluations conducted on benchmark datasets, including Iris, Seeds, MNIST, and CIFAR-10, demonstrate that CEG4N effectively maintains model accuracy and behavioral integrity, outperforming traditional quantization methods such as the greedy path following quantization (GPFQ), particularly in smaller and sparse networks. While scalability to larger models remains challenging due to increased computational overhead associated with formal verification methods, the proposed framework significantly advances the robustness and reliability of quantized neural networks. Moreover, this research underscores the importance of integrating formal verification into quantization to optimally balance efficiency and accuracy. Ultimately, these contributions facilitate the deployment of neural networks in environments with stringent resource and reliability constraints, promoting safer and more efficient artificial intelligence systems.

Keywords: Neural networks, quantization, formal equivalence verification, functional equivalence, resource-constrained environments, safety critical systems.

Resumo

No campo em rápida evolução da inteligência artificial, redes neurais profundas (DNNs) tornaram-se essenciais em diversas aplicações, desde veículos autônomos até sistemas de saúde. No entanto, o aumento no tamanho e na complexidade dessas redes apresenta desafios significativos para sua implementação em ambientes com restrições de recursos e críticos para segurança, devido às altas demandas computacionais e de memória. Esta dissertação aborda o problema crítico da quantização de redes neurais, cujo objetivo é reduzir a precisão dos parâmetros das redes para aumentar a eficiência sem comprometer seu comportamento funcional. A principal contribuição é o desenvolvimento do arcabouço Quantização Guiada por Contraexemplos para Redes Neurais (CEG4N), que integra a Verificação Formal de Equivalência (FEV)—uma técnica que rigorosamente verifica se dois modelos apresentam comportamento idêntico—ao processo de quantização. Metodologicamente, o CEG4N emprega um ciclo iterativo de otimização que utiliza ferramentas de verificação formal para detectar e corrigir sistematicamente discrepâncias entre os modelos original e quantizado, garantindo assim sua equivalência funcional (FE). Avaliações experimentais realizadas em benchmarks como Iris, Seeds, MNIST e CIFAR-10 demonstram que o CEG4N mantém efetivamente a precisão e integridade comportamental dos modelos, superando métodos tradicionais de quantização como quantização por seguimento de caminho guloso (GPFQ), especialmente em redes menores e esparsas. Embora a escalabilidade para modelos maiores permaneça um desafio devido ao alto custo computacional associado aos métodos de verificação formal, o arcabouço proposto avança significativamente na robustez e confiabilidade das redes neurais quantizadas. Além disso, esta pesquisa ressalta a importância da integração da verificação formal com a quantização para equilibrar de maneira ótima eficiência e precisão. Em última análise, estas contribuições facilitam a implementação de redes neurais em ambientes com rigorosas restrições de recursos e confiabilidade, promovendo sistemas de inteligência artificial mais seguros e eficientes.

Palavras-chave: Redes Neurais, Quantização, Verificação Formal de Equivalência, Equivalência Funcional, Ambientes com Recursos Limitados, Sistemas Críticos para Segurança.

List of Figures

List of Tables

List of abbreviations and acronyms

ACAS Xu	Airborne Collision Avoidance System Xu Dataset
AI	Artificial Intelligence
AM	Abstractions Module
ASIC	Application-Specific Integrated Circuit
BNN	Binarized Neural Network
BMC	Bounded Model Checking
BSM	Bits Search Module
CEG4N	Counterexample-guided neural network quantization framework
CESBMC	CEG4N in combination with ESBMC tool
CNN	Convolutional Neural Network
CNNEQUIV	CEG4N in combination with NNEquiv tool
DNN	Deep Neural Network
DL	Deep Learning
EC	Equivalence Checking
EG1, EG2, ...	Evaluation Goals
ESBMC	Efficient SMT-Based Model Checker
EV	Equivalence Verification
FEV	Formal Equivalence Verification
FE	Functional equivalence
FPGA	Field-Programmable Gate Array
FPA	Floating-Point Arithmetic
FV	Formal Verification
GAN	Generative Adversarial Network

GA	Genetic Algorithm
GPE	Geometric Path Enumeration
GPFQ	Gradient-based Post-Training Quantization
GPU	Graphics Processing Unit
HAQ	Hardware-Accelerated Quantization
IoT	Internet of Things
MNIST	Modified National Institute of Standards and Technology Dataset
MSE	Mean Squared Error
NN	Neural Network
NNE	Neural Network Equivalence
NNEV	Neural Network Equivalence Verification
NNs	Neural Networks
NNEQUIV	Neural Network Equivalence Verifier
N. CEs	Number of counterexamples
O.Ac.	Original Accuracy
ONNX	Open Neural Network Exchange
PTQ	Post-Training Quantization
Q.Ac.	Quantized Accuracy
QAT	Quantization-Aware Training
QNN	Quantized Neural Network
ReLU	Rectified Linear Unit Activation Function
RNN	Recurrent Neural Network
SAT	Boolean Satisfiability
Sigm	Sigmoid activation function
SMT	Satisfiability Modulo Theories
TanH	Hyperbolic Tangent activation function
VM	Verifier Module

List of symbols

$\%$	Percentage
δ	Input Space Bounds
f	Original Neural Network function
f'	Quantized Neural Network function
f''	De-quantized Neural Network function
$\mathcal{Q}$	Quantization function
$\mathcal{J}(\mathbf{b})$	Cost function for quantization parameters
$\mathbf{b}$	Bit width vector for NN layers
$b_{\min}, b_{\max}$	Minimum and maximum bit width for quantization
$\mathcal{R}^{\delta}$	Critical input regions
$\mathcal{R}^{\delta=0}$	Critical data points
$\mathcal{G}$	Equivalence verification function
$\mathcal{X}^{\delta}$	Set of counterexamples for equivalence verification
$\backsimeq$	Equivalence relation between $[f]$ and $[f']$
$\mathcal{I}$	Input domain
$\mathcal{O}$	Output domain
L	Number of layers in a neural network
$\mathbf{W}^{(l)}$	Weight matrix for layer $[l]$
$\mathbf{b}^{(l)}$	Bias vector for layer $[l]$
$\mathbf{z}^{(l)}$	Non-activated output for layer $[l]$
$\mathbf{a}^{(l)}$	Activated output for layer $[l]$
σ	Activation function
b	Bit-width for quantization

s	Scale factor
z	Zero-point
α, β	Quantization range for real values
α', β'	Quantization range for quantized values
$\mathcal{Q}_b$	General quantization function
$\mathcal{Q}'_b$	Quantization mapping function
$\mathcal{Q}''_b$	Dequantization function
$\|\| \cdot \|\|_p$	$[L_p]$-norm
ϵ	Tolerance level in norm-based equivalence
$\backsimeq$	General equivalence relation
$\approx_1$	Classification-based equivalence relation
$\backsim_{p,\epsilon}$	Norm-based equivalence relation

Contents

1 Introduction

1.1 Background

Over recent decades, artificial intelligence (AI) has significantly evolved from simple computational models to complex algorithms capable of intricate tasks (LECUN; BENGIO; HINTON, 2015; RUSSELL; NORVIG, 2021). Central to this progress are deep neural networks (DNNs), sophisticated computational models inspired by the human brain's neural structure, enabling advanced data processing (KRUSE et al., 2016). These advancements in deep learning (DL) have resulted in powerful neural networks (NNs), excelling in various applications, including image recognition, natural language processing, and strategic decision-making (KRIZHEVSKY; SUTSKEVER; HINTON, 2012; GOODFELLOW; BENGIO; COURVILLE, 2016; SILVER et al., 2016).

The widespread adoption of DNNs across industries—from smartphones to autonomous vehicles—has led to increasingly complex models that are computationally intensive and resource-demanding (SZE et al., 2017; CHEN et al., 2020). This complexity poses substantial challenges in resource-constrained environments, necessitating more efficient computational methods to ensure universal applicability.

Quantization techniques have emerged as a promising solution to these challenges, reducing the precision of NN computations from high-precision floating-point arithmetic (FPA) to lower-precision formats, such as fixed-point arithmetic (JACOB et al., 2017; LIN; TALATHI; ANNAPUREDDY, 2016). Such techniques significantly decrease model size, computational requirements, and energy consumption, thereby enabling efficient deployment in resource-limited environments (HAN; MAO; DALLY, 2016; NAGEL et al., 2021). However, quantization introduces new challenges, particularly regarding maintaining accuracy and ensuring the reliability of NNs, especially in safety-critical applications, where small errors can lead to significant negative outcomes (JACOB et al., 2017; HUBARA et al., 2017).

Reliability, in this context, refers to a quantized neural network's ability to consistently perform as expected across varying operational conditions, whereas accuracy specifically measures the correctness of predictions against ground truth data. Thus, preserving both accuracy and reliability during quantization is critical, motivating the exploration of rigorous methodologies to ensure these properties are maintained in quantized models.

1.2 Motivation

Ensuring a proper balance between efficiency and accuracy in quantized neural networks (QNNs) has become a vital research priority, particularly for safety-critical applications such as autonomous vehicles and healthcare devices. In these domains, robustness—the capacity to perform reliably despite input variations—and reliability—the consistency of correct operation over time—are critical requirements.

Existing quantization techniques typically measure performance degradation using statistical methods based on accuracy and loss metrics. While informative, these methods fail to comprehensively address subtle behavioral changes caused by quantization, which could be crucial in safety-critical contexts. These limitations underscore the need for more rigorous verification frameworks that can ensure QNNs retain their original functional behavior.

Formal verification (FV) techniques, which mathematically prove system properties, provide a promising approach to addressing this gap. However, traditional FV methods are often decoupled from the quantization process, leading to inefficient iterative cycles of quantization and verification (JACOB et al., 2017; HUANG et al., 2017). Integrating FV into the quantization process could streamline this workflow, allowing for immediate feedback and adjustment of quantization parameters, thus ensuring functional equivalence between original NNs and their quantized counterparts.

1.3 Problem Statement

This thesis addresses the critical challenge of ensuring that neural network quantization does not alter the original network's functional behavior. Current quantization approaches primarily focus on preserving accuracy metrics but overlook detailed behavioral equivalence. Such oversight is problematic in safety-critical systems, where minor behavioral discrepancies can have significant real-world consequences.

While several quantization methods demonstrate effectiveness in general benchmarks, they do not guarantee reliability or robustness against all potential inputs, particularly adversarial scenarios (GHOLAMI et al., 2022; LOHAR et al., 2023). Statistical accuracy measures alone are insufficient as they do not adequately reflect the vulnerabilities and nuanced behavioral changes introduced by quantization.

To robustly address these issues, this thesis proposes leveraging equivalence checking (EC), a formal verification approach explicitly designed to assess the behavioral equivalence between two neural networks (BÜNING; KERN; SINZ, 2020; ELEFTHERIADIS et al., 2022). EC ensures quantized models replicate the original model's decision-making processes precisely, thereby providing a rigorous methodology suitable for safety-critical applications.

1.4 Objectives

The goal of this thesis is to develop a robust framework for neural network quantization that ensures functional behavioral equivalence between original and quantized models. To systematically achieve this goal, the following specific objectives have been identified:

- **Investigate the behavioral impact of NN quantization:** Analyze how quantization affects neural network behavior, specifically addressing accuracy, robustness, reliability, and vulnerability to adversarial inputs.
- **Develop a modular, iterative quantization framework:** Design and implement a quantization methodology integrating formal verification to ensure functional behavior preservation.
- **Integrate formal equivalence verification:** Establish equivalence checking as a core component of the quantization process to rigorously verify behavioral equivalence between original and quantized models.
- **Propose optimization-based quantization methods:** Create optimization strategies within the quantization framework aimed at balancing efficiency gains with strict preservation of original neural network behavior.
- **Evaluate and validate the proposed framework:** Empirically validate the developed methodologies using various benchmarks and neural network architectures, demonstrating their effectiveness compared to existing techniques.

Achieving these objectives will significantly advance the field, providing both theoretical insights and practical tools to safely and efficiently deploy quantized neural networks in constrained environments.

1.5 Publications

The methodologies and results developed in this thesis have been disseminated through peer-reviewed publications, underscoring their relevance and contributions to the scientific community.

Our research introduced the Counterexample Guided Neural Network Quantization Refinement (CEG4N) framework, a novel technique integrating search-based quantization with formal equivalence checking. This method ensures that neural networks retain their functional behavior after quantization, crucial for deploying reliable and efficient neural

networks in resource-constrained environments. The initial concept and preliminary evaluations of CEG4N demonstrated its effectiveness on various benchmarks, highlighting up to 72% accuracy improvement over state-of-the-art quantization techniques (MATOS et al., 2022).

Further development and extensive experimental validation of the CEG4N framework extended its capabilities and applications. We refined the equivalence checking process, enhancing the precision of behavioral equivalence guarantees between original and quantized neural networks. The improved version of the CEG4N framework was validated using diverse benchmarks, including deep convolutional neural networks and safety-critical applications like the ACAS Xu system, achieving models with up to 163% better accuracy compared to existing techniques (MATOS et al., 2024).

These contributions have advanced the intersection of neural network quantization and formal verification, providing robust methodologies for maintaining functional correctness in quantized models.

1.6 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 presents the theoretical foundations of neural networks, quantization methods, formal verification, and relevant optimization approaches, clearly defining the research problem.

Chapter 3 details the proposed iterative quantization refinement framework, explicitly integrating equivalence checking and optimization methods, along with underlying assumptions and limitations.

Chapter 4 describes empirical validation processes, presenting comparative analyses against existing quantization and verification methods.

Chapter 5 critically analyzes the results, discusses implications for deployment in resource-constrained systems, and addresses potential limitations and future improvements.

Chapter 6 surveys existing literature on quantization techniques, formal verification methodologies, and current challenges in preserving functional behavior.

Chapter 7 summarizes the thesis contributions, highlights its significance, and outlines directions for future research.

2 Theoretical Foundations

In this chapter we present the background theory that are fundamental for the understanding the research conducted in this thesis. It covers the basics of NNs, quantization processes, and the principles of formal equivalence verification. It also explores optimization techniques relevant to NN quantization. The mathematical formulation of the research problem is presented, providing a clear and precise statement of what problem this thesis aims to solve.

2.1 Neural Networks (NNs)

An NN consists of multiple layers of interconnected nodes that resemble a direct graph, and each node acts as an artificial neuron that emulates the functions of a biological neuron. NNs are structured with layers of nodes, or neurons, linked by edges that symbolize weights. As illustrated in Figure Figure 1, NN consists of three types of layers: an input layer, an output layer, and one or more hidden layers.

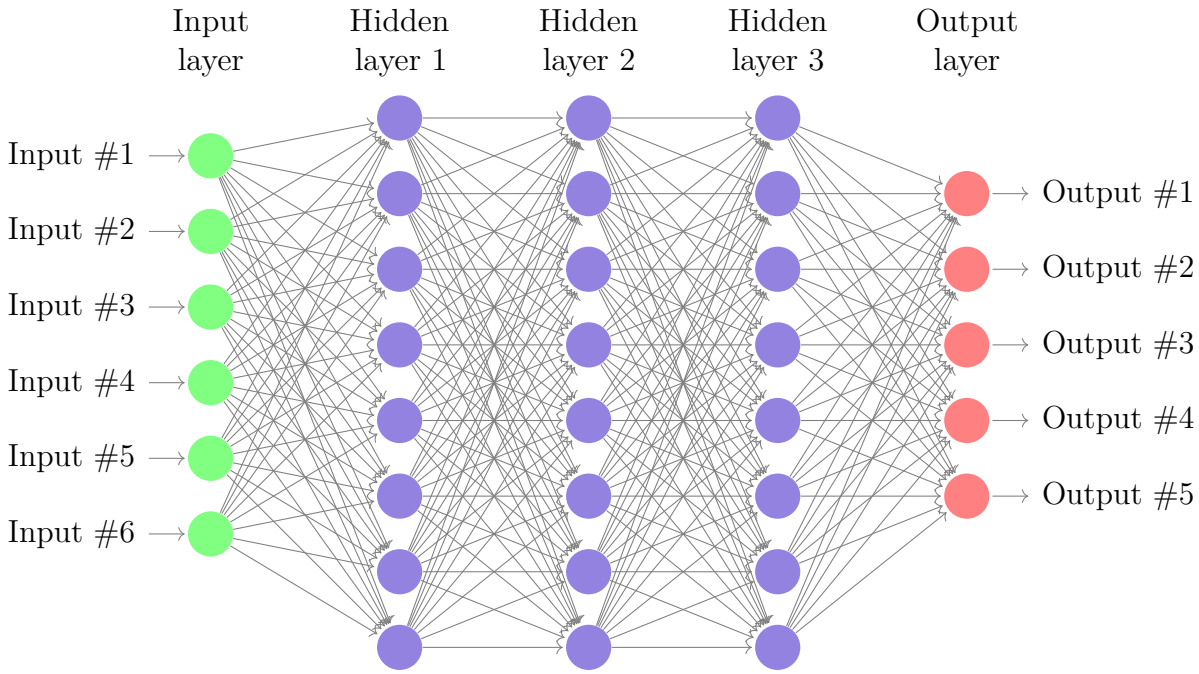

Figure 1 – A simple artificial NN.

Each of the layers plays a key role in the functionality of the NN. The input layer is the first layer of the NN, which receives the initial data for processing. Each neuron in the input layer represents a feature of the input dataset. The hidden layers, in turn,

lie between the input and output layers and are termed "hidden" because they do not directly interact with the external environment, i.e., they neither observe the input data nor produce the final output. The complexity and capacity of the NN increases with the number of hidden layers and neurons within them, enabling the NN to capture more complex patterns and relationships in the data. Finally, the output layer produces the final result of the NN, delivering the prediction or classification based on the input data. The structure of the output layer varies depending on the specific task, such as regression, binary classification, or multi-class classification.

2.2 Background on NNs

AI encompasses a wide range of technologies capable of performing tasks that require human intelligence. These technologies include reasoning, learning, problem-solving, perception, and understanding language (ANANTRASIRICHAI; BULL, 2020). Machine Learning (ML), a subset of AI, focuses on algorithms that allow computers to learn from and make predictions or decisions based on data. ML has advanced various fields, including predictive maintenance in Industry 4.0 (BERTOLINI et al., 2021). Deep Learning (DL), a further subset of ML, uses NNs with many layers (deep NNs) to analyze various forms of data. Its capabilities extend to image recognition, natural language processing, and more complex data interpretation tasks (LECUN; BENGIO; HINTON, 2015).

NNs, inspired by the neuronal structure of the human brain, consist of interconnected nodes or neurons that process and transmit signals (KRUSE et al., 2016). These networks can learn and adapt, making them versatile tools for various computational tasks (RUSSELL; NORVIG, 2021). NNs have been juxtaposed with traditional statistical methods, often in a comparative context, to understand their predictive and classification capabilities across different applications (PALIWAL; KUMAR, 2009).

NNs represent a foundational technology in AI, offering unparalleled capabilities in pattern recognition, data analysis, and predictive modeling. Their diverse architectures enable them to tackle various tasks, from simple classifications to complex problem-solving across various domains.

NN architectures define the structure and function of these systems (KRUSE et al., 2016), including the arrangement of neurons and layers, connections between neurons, and the methods of data processing and signal transmission within the network (KRUSE et al., 2016). Typically, the architecture of an NN is tied to a particular application domain, indicating what sets of problems a particular NN is better-suited for (AHAMED, 2016; HAPPEL; MURRE, 1994). Notable architectures and key applications are discussed below.

Feedforward NNs (FFNNs) are the simplest NNs where connections between the nodes do not form a cycle. This architecture is particularly highlighted in comparative

studies with statistical techniques for prediction and classification (PALIWAL; KUMAR, 2009). An example of using FFNNs in radiology highlights this architecture's utility in pattern recognition and signal detection, offering better accuracy than human observers in certain diagnostic tasks (BOONE; SIGILLITO; SHABER, 1990).

Convolutional NNs (CNNs) are widely used in processing visual imagery, characterized by convolutional layers that automatically and adaptively learn spatial hierarchies of features from input images (LI et al., 2020). CNNs have revolutionized image processing and computer vision, offering superior performance in image classification, object detection, and more (EGMONT-PETERSEN; RIDDER; HANDELS, 2002).

Recurrent NNs (RNNs) (DAS et al., 2023) and long short-term memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) are used for sequential data or where the order and context of the data are critical, such as in time series analysis or natural language processing. NNs have significantly improved the performance of language models (ZHOU et al., 2020), enabling breakthroughs in machine translation (SENNRICH; HADDOW; BIRCH, 2015), sentiment analysis (ZHANG et al., 2023), and automated content creation (KABRA et al., 2022).

Indeed, NNs have found applications across various fields due to their versatility and powerful learning capabilities (PALIWAL; KUMAR, 2009; KRUSE et al., 2016). In healthcare and biomedicine, NN applications range from diagnostics and medical imaging analysis to predicting disease outbreaks, highlighting their potential to improve patient care and health outcomes (WEISS et al., 2022). In financial analysis and prediction, NNs have been utilized for stock market prediction, fraud detection, and customer data analysis, demonstrating their adaptability to analyze complex financial systems (WONG; SELVI, 1998). In environmental modeling and climate analysis, they have been employed in predicting weather patterns and analyzing climate change impacts and environmental monitoring, showcasing their ability to handle large-scale, complex data sets (MAIER; DANDY, 2000).

2.2.1 NN Notation

NNs (NNs) represent non-linear multivariate functions $f : \mathcal{I} \subset \mathbb{R}^n \rightarrow \mathcal{O} \subset \mathbb{R}^m$ where $\mathcal{I}$ and $\mathcal{O}$ denote the input and output domains, and n is the dimension of the input space and m is the dimension of the output space. Internally, an NN is structured as a directed graph comprising a set of L hidden layers. In a feedforward NN, the neurons in each layer $l = \{1, ..., L\}$ are connected to those in the preceding layer $l - 1$, each applying a linear transformation followed by a non-linear transformation. Additionally, the neurons in the first layer $l = 1$ serve as placeholders for the NN's input, while those in the last

layer $l = L$ represent the function f's output. Formally, for a network with L layers:

$$f(x) = f_L(f_{L-1}(...f_1(x))),$$

where x denotes an input sample from the input domain $\mathcal{I}$. The output of the last layer of the NN is usually considered the output of the function f on a specific input x, and is normally denoted as $y = f(x)$. Except for the last layer, the output $\mathbf{y}_{(\mathbf{l})}$ of each of the other layers (i.e., the hidden layers) is computed by applying both an affine and a non-linear transformation. The non-activated and activated outputs of layer L, denoted by $\mathbf{z}^{(\mathbf{l})}$ and $\mathbf{a}^{\mathbf{l})}$, respectively, are:

$$\mathbf{z}^{(\mathbf{l})} = \mathbf{W}^{(\mathbf{l})} \cdot \mathbf{a}^{(\mathbf{l-1})} + \mathbf{b}^{(\mathbf{l})}, \tag{2.1}$$

$$\mathbf{a}^{(\mathbf{l})} = \sigma(\mathbf{z}^{(\mathbf{l})}), \tag{2.2}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{m_{l-1} \times m_l}$ represents the weight matrix, $\mathbf{b}^{(l)} \in \mathbb{R}^{m_l}$ the bias matrix, and $\sigma : \mathbb{R}^{m_l} \rightarrow \mathbb{R}^{m_l}$ the non-linear activation function. And $\mathbf{a}^{(0)} = \mathbf{x}$ is the NN's input.

Popular activation functions include the Rectified Linear Unit (ReLU), the sigmoid (Sigm), and the hyperbolic tangent (TanH). ReLU is the most common and widely used activation function. The ReLU function can be defined as:

$$\text{ReLU}(\mathbf{z}^{(\mathbf{l})}) = \max\{0, \mathbf{z}^{(\mathbf{l})}\}. \tag{2.3}$$

This notation supports expressing the arithmetic for both fully-connected and convolutional layers. Note that the Equation 2.1 can be written in terms of matrix multiplication and addition as follows:

$$\begin{pmatrix} z_0^{(l)} \\ z_1^{(l)} \\ \vdots \\ z_m^{(l)} \end{pmatrix} = \begin{pmatrix} \mathbf{w}_{\mathbf{0,0}}^{(\mathbf{l})} & \cdots & \mathbf{w}_{\mathbf{0,n}}^{(\mathbf{l})} \\ \mathbf{w}_{\mathbf{1,0}}^{(\mathbf{l})} & \cdots & \mathbf{w}_{\mathbf{1,n}}^{(\mathbf{l})} \\ \vdots & \ddots & \vdots \\ \mathbf{w}_{\mathbf{m,0}}^{(\mathbf{l})} & \cdots & \mathbf{w}_{\mathbf{m,n}}^{(\mathbf{l})} \end{pmatrix} \begin{pmatrix} \mathbf{a}_{\mathbf{0}}^{(\mathbf{l-1})} \\ \mathbf{a}_{\mathbf{1}}^{(\mathbf{l-1})} \\ \vdots \\ \mathbf{a}_{\mathbf{m}}^{(\mathbf{l-1})} \end{pmatrix} + \begin{pmatrix} \mathbf{b}_{\mathbf{0}}^{(\mathbf{l})} \\ \mathbf{b}_{\mathbf{1}}^{(\mathbf{l})} \\ \vdots \\ \mathbf{b}_{\mathbf{m}}^{(\mathbf{l})} \end{pmatrix},$$

where the matrix $\mathbf{W}^{(\mathbf{l})}$ is dense in fully-connected layers and sparse in convolutional ones (BISHOP, 2006).

2.3 Quantization

In this Section, we present the fundamental concepts of NN quantization and fixed-point hardware accelerators that support the algebraic operations of QNNs. We

begin by outlining the hardware rationale and explaining typical quantization methods and their characteristics. Subsequently, we dive into the practical aspects concerning layers frequently encountered in contemporary NNs and how they impact fixed-point accelerators.

2.3.1 Introduction

Quantization refers to limiting high-precision values, such as single-precision floating-point numbers, to a finite set of lower-precision values, such as integers. This procedure is characterized by three key parameters: bit-width b, scale factor s, and zero point z. The bit-width b, which represents the number of bits available (or the desired number of bits to represent the quantized data), sets the limits for the quantization range. That is, it determines both the quantization process's precision and the quantized data's size. The scale factor is responsible for aligning the quantized value range with the input data range and setting the increment size between successive quantization levels. The zero-point moves the quantization grid to accurately represent a specific value, which may not be zero.

As mentioned above, *quantization* refers to mapping a set of real number values to a set of discrete integer values. However, it can also involve the reverse process, in which these discrete quantized values are mapped back to a floating-point representation, in the process called *de-quantization*. This dual nature of quantization is crucial for various applications. Having the capability to establish this two-way connection enables achieving a compromise between the effectiveness and conciseness of quantized information, enabling one to closely mimic the precision and range of the original data when required. Achieving this equilibrium is crucial for efficiently applying quantization in technology and computing.

The bit-width refers to the number of bits used to encode individual floating-point values within a finite, limited set of quantized values, determining the quantization resolution. That is the number of discrete levels or steps that can be represented. For example, an 8-bit quantization allows 2^8 different levels or values, while a 32-bit quantization allows for 2^{32} levels. Choosing an appropriate bit-width is crucial for balancing the quantized data's precision and the storage or computational resources required to handle the data. A lower bit-width results in smaller data sizes and lower computational complexity, but can lead to higher quantization error and loss of information. Conversely, a higher bit-width reduces the quantization error but at the cost of increased data size and computational requirements.

Quantization ranges setting refers to the process of defining the quantization range (also known as the *clipping range*), that is, the arbitrary definition of a set of minimum and maximum limit values for the input data (floating point data), in the form of the range $[\alpha, \beta]$. The process of selection of α and β is commonly known as *calibration*. One simple option is to utilize the *minimum/maximum* values of the input data for setting the

clipping range, that is, let x be a floating-point input value, and $\mathcal{X}$ be the set of all know input values, such that $x \in \mathcal{X}$, thus:

$$\begin{aligned} \alpha &= \min_{x \in \mathcal{X}} x \\ \beta &= \max_{x \in \mathcal{X}} x \end{aligned} \tag{2.4}$$

Another possibility for selecting the clipping range is to establish it according to the *minimum/maximum* values, as shown below:

$$\begin{aligned} x_{min} &= |\min_{x \in \mathcal{X}} x| \\ x_{max} &= |\min_{x \in \mathcal{X}} x| \\ -\alpha = \beta &= \max_{x \in \{x_{min}, x_{max}\}} x \end{aligned} \tag{2.5}$$

Similarly, another significant range is represented by $[\alpha', \beta']$, which establishes the lower and upper limits that the quantized format can encode. This other range can be seen as the clipping range for the quantized values (which will be further discussed in Section Section 2.3.1.1). However, this range is defined more simply based on the bit-width chosen for the quantization process. For example, in an 8-bit quantization system, quantized values can range from 0 to 255, meaning that any input value must be mapped or scaled to fall within this range $[\alpha' = 0, \beta' = 255]$. Furthermore, the definition of $[\alpha', \beta']$ is determined by whether signed or unsigned integers ranges desired for quantization. For example, the $[\alpha', \beta']$ range for a signed integer quantization is defined as follows. Let b be the bit-width, thus the minimum and maximum values (α', β') can be computed in relation to b as follows:

$$\begin{aligned} \alpha' &= -2^{b-1} \\ \beta' &= -2^{b-1} - 1, \end{aligned} \tag{2.6}$$

while the $[\alpha', \beta']$ range for an unsigned integer quantization is defined as follows. Let b the bit-width, thus the minimum and maximum values (α', β') can be computed in relation to b as follow:

$$\begin{aligned} \alpha' &= 0 \\ \beta' &= 2^{b-1} \end{aligned} \tag{2.7}$$

The selection of quantization intervals directly impacts the level of detail and precision of the quantized values, which in turn influences the faithfulness of the data representation (GHOLAMI et al., 2022). Optimal quantization intervals can help reduce data loss and maintain the fundamental features of the initial data in quantized form.

The main consideration when setting intervals is to find a balance between clipping and rounding errors and how they affect information loss during quantization (NAGEL et al., 2021; JACOB et al., 2017).

In quantization, the scale factor is a crucial multiplier employed to modify the range of input data values. It plays a vital role in ensuring that the original input data range is effectively represented in the quantized format by appropriately scaling the data up or down to align with the desired range, thereby minimizing any substantial loss of information. Thus, to compute the scaling factor, one must first define the desired bit-width and then compute the quantization ranges (as discussed in Section Section 2.3.1.1). With the computed quantization ranges,the scaling factor is computed as:

$$s = \mathcal{S}(\alpha, \beta, \alpha', \beta') = \frac{\beta - \alpha}{\beta' - \alpha'} \tag{2.8}$$

The zero-point refers to a specific value representing the real number zero within the quantized data range. It acts as an offset in the quantization formula, enabling the mapping of real-valued inputs, which may include negative numbers, to positive integers in the quantized output. The zero-point integration into the quantization formula can be seen in Equations Equation 2.10 and Equation 2.12 in Section 2.3.1.1, respectively.

In quantization, not all values can be perfectly aligned with their real-number equivalents due to the limited precision imposed by the bit-width. The zero-point ensures that the quantization process can accurately represent zero and, by extension, values around zero, which is critical for maintaining the integrity of input data with positive and negative values. By carefully choosing the zero-point along with the scale factor, quantization can minimize information loss and preserve the relative distribution of the original data values. The zero point can be computed as follows:

$$z = \mathcal{Z}(\alpha, \beta, \alpha', \beta') = \left\lfloor \frac{\beta\alpha' - \alpha\beta'}{\beta - \alpha} \right\rceil \tag{2.9}$$

2.3.1.1 Quantization Mapping

Mathematically, quantization can be expressed as a mapping function $\mathcal{Q}'_b : \mathbb{R}^{m \times p} \rightarrow \mathbb{I}^{m \times p}$, which takes a floating point value $x \in \mathbb{R}^{m \times p} \mid \alpha \leq x \leq \beta$ and transforms it into a $\lfloor$-bit integer $x' \in \mathbb{I}^{m \times p} \mid \alpha' \leq x' \leq \beta'$. Given a floating point value x, one can transform it into an integer value x' within the grid of integers $\{\alpha', ..., \beta'\}$, as defined below:

$$x' = \mathcal{Q}'_b(x, z, \alpha', \beta') = clip\left(\left\lfloor \frac{x}{s} \right\rceil + z, \alpha', \beta'\right), \tag{2.10}$$

where x represents the high-precision value to be quantized, b denotes the bit-width for quantization, and s is the scaling factor for x relative to b, and z is the zero-point. The *clip*

term denotes the clipping function that ensures that the quantized values remain within the specified lower and upper bounds α' and β', respectively. The details of the clipping function are presented in Section Section 2.3.1.1. Lastly, $\lfloor \cdot \rceil$ denotes the rounding of the values to the nearest integer.

Dequantization involves converting quantized values back into a more continuous format or of higher precision, typically involving floating-point numbers. In quantization, values are transformed from a larger, often continuous set to a smaller, discrete set to simplify complexity and storage demands. Therefore, dequantization acts as the inverse process of quantization. By utilizing the same parameters employed during quantization, such as bit-width $\lfloor$, scale factor s, and zero-point z - dequantization enables the recreation of the original data from their quantized form as accurately as possible. This is achieved by approximating the low-precision (quantized) value to its original high-precision counterpart. Essentially, dequantization closes the gap between the efficiency of quantization and the necessity for precise data representation.

Mathematically, dequantization can be expressed as a mapping function $\mathcal{Q}_b'' : \mathbb{I}^{m \times p} \rightarrow \mathbb{I}^{m \times p}$, which takes a $\lfloor$-bit integer value $x' \in \mathbb{I}^{m \times p} \mid \alpha' \leq x \leq \beta'$ and transforms it into a floating-point value $x'' \in \mathbb{R}^{m \times p} \mid \alpha' \leq x \leq \beta'$. Given a integer value x', one can transform it into an floating-point value $x'' \mid x'' \approx x$, as follows:

$$\begin{aligned} x'' &= s(x' - z) \\ &= s(Q_b'(x, z, \alpha', \beta') - z) \end{aligned} \tag{2.11}$$

Combining the definitions of quantization (defined in Section Equation 2.10), and dequantization defined above, one can formalize a general function for uniform quantization, in the form of $\mathcal{Q}_b : \mathbb{R}^{m \times p} \rightarrow \mathbb{R}^{m \times p}$, defined as follows:

$$x'' = \mathcal{Q}_b(x) = s \left[clip \left(\left\lfloor \frac{x}{s} \right\rceil + z, \alpha, \beta \right) - z \right] \tag{2.12}$$

Notice that until now the formalization of quantization and de-quantization has been using a real value x, this serves the purpose of simplifying the notation being used. However, the formalization of $\mathcal{Q}_b$ is flexible enough to allow the real value x to be replaced by a vector $\mathbf{x}$ of floating-point values and also a matrix $\mathbf{X}$ of floating-point values.

The formalization provided for the quantization mapping function $\mathcal{Q}_b$ is indeed generalized and versatile enough to be extended naturally from scalar real values to vectors $\mathbf{x} \in \mathbb{R}^m$ and matrices $\mathbf{X} \in \mathbb{R}^{m \times p}$. The notation employed initially utilized scalar values x primarily for clarity and simplicity. Nonetheless, the mathematical expressions defined in equations (2.10), (2.11), and particularly (2.12), remain valid and applicable when generalized to multidimensional arrays.

Explicitly, when generalizing from a scalar x to a vector $\mathbf{x}$ or a matrix $\mathbf{X}$, each operation such as scaling, rounding, clipping, and offsetting (via zero-point z) should

be applied element-wise. Therefore, for a vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]$, the quantization-dequantization function $\mathcal{Q}_b$ is computed independently for each element, as follows:

$$\mathbf{x}'' = \mathcal{Q}_b(\mathbf{x}) = s\left[clip\left(\left\lfloor\frac{\mathbf{x}}{s}\right\rceil + z, \alpha', \beta'\right) - z\right],$$

where the operations *clip*, *round*, and arithmetic are understood as applied element-wise.

Similarly, for matrices, $\mathbf{X} = [x_{ij}] \in \mathbb{R}^{m\times p}$, the quantization-dequantization mapping is also applied element-wise:

$$\mathbf{X}'' = \mathcal{Q}_b(\mathbf{X}) = s\left[clip\left(\left\lfloor\frac{\mathbf{X}}{s}\right\rceil + z, \alpha', \beta'\right) - z\right].$$

In this generalized context, each element of the input vector or matrix undergoes the same quantization-dequantization process independently, ensuring consistency and simplicity of implementation, while maintaining mathematical rigor and clarity.

In practice, the quantization process will have the possibility of having values outside the range of α, β, therefore, the quantized value x'' will also be outside the range of α', β'. If the integer grid is signed, then a typical choice is to define $(\alpha', \beta') = (-2^{b-1}, 2^{b-1}-1)$. On the other hand, if the integer grid is unsigned, a typical choice is to define $(\alpha', \beta') = (0, 2^b - 1)$. To ensure no integer or floating-point overflows or unintended quantization errors, the quantization process requires the clipping step. That is, values of x that fall outside of this interval are coerced into its limits, resulting in a clipping error. To minimize this error, one can define a larger quantization range by increasing the scale factor. However, increasing the scale factor results in higher rounding errors. As noted by Nagel et al. (NAGEL et al., 2021), the rounding error lies in the range $[-\frac{1}{2}s, \frac{1}{2}s]$. This is a trade-off an important aspect to consider in the quantization mapping and a justification to why we can only approximate the original floating point value x, therefore, quantization errors must be assumed.

In symmetric quantization, the clipping range is defined following Equation Equation 2.6 in Section Section 2.3.1, and, in addition, the zero-point is fixed at zero, that is, $z = \mathcal{Z}(\alpha, \beta, \alpha', \beta') = 0$. The name symmetric quantization comes from the symmetric relation between the parameters α and β, that is, $\alpha = -\beta$. In the asymmetric quantization, the clipping range may not be symmetric with respect to the origin (GHOLAMI et al., 2022), i.e., $-\alpha \neq \beta$, hence the name asymmetric quantization. For example, one way to achieve asymmetric quantization would be to define the quantization ranges accordingly to Equation Equation 2.4 in Section Section 2.3.1. This approach is not the only way to achieve asymmetric quantization, but it provides a rather suitable example.

Asymmetric quantization often results in a tighter clipping range compared to symmetric quantization (GHOLAMI et al., 2022), as it allows for more control over the zero-point offset. That is, by moving the zero-point towards the sparser region of the

input data, consequently, more quantization levels are made available for the denser region. Symmetric quantization, on the other hand, can be seen as a specific instance of asymmetric quantization; that is, it is a simplified version of the general asymmetric case. It has two advantages: (1) it is a good choice for distributions that are roughly symmetric around zero, and (2) it allows the complete removal of the zero-point as a term in the the quantization and de-quantization formulas, therefore simplifying them.

2.3.2 Quantization of NNs

The are a number of different strategies that can be followed to quantize NNs (JACOB et al., 2017; LIN; TALATHI; ANNAPUREDDY, 2016), that can result in many combinations of parameters being quantized or not. In this section, for the sake of simplification, the focus will be to provide a basic notation for the quantization of NNs. Consider a FFNN f as defined in Equation 2.1 and Equation 2.2. Call B_f the set whose elements $b_{(l)} \in B_f$ are the bit widths for each layer h in f. Given the de-quantization process in Equation 2.11 and the non-activated output in Equation 2.1, we can describe the non-activated output $z''_{(l)}$ of a quantized layer l as:

$$\begin{aligned} \mathbf{z}'' &= \mathcal{Q}(\mathbf{z}_{(\mathbf{l})}, b_{(l)}) \\ &= \mathcal{Q}(\mathbf{b}_{(\mathbf{l})}, b_{(l)}) + \mathcal{Q}(\mathbf{W}_{(\mathbf{l})}, b_{(l)})\mathcal{Q}(\mathbf{a}_{(\mathbf{l})}, b_{(l)}) \\ &= \mathbf{b}''_{(\mathbf{l})} + \mathbf{W}''_{(\mathbf{l})}\mathbf{a}''_{(\mathbf{l})} \end{aligned} \tag{2.13}$$

Notice that in the notation above the input, the weight, and the bias are quantized using the same bit-width $b_{(l)}$. This is a valid quantization approach, however it is not the only one. There other approaches that allow the input, the weight, and the bias to have a bit-widths associated with them. However, the notation to express that is quite heavy and does not bring clarity. Also notice that the clipping ranges were omitted from this notation.

2.3.3 Practical considerations

When quantizing neural networks (NNs) with multiple layers, there exists a wide range of quantization options, including the quantization method, granularity, and bit-width. Granularity, in this context, refers to the level at which quantization is applied, such as per-layer, per-channel, or per-tensor quantization, each offering different trade-offs between precision, computational efficiency, and hardware implementation complexity. This section delves into and examines some practical factors that can help narrow down the search space for selecting appropriate quantization strategies.

2.3.3.1 Mixed-precision quantization

Homogeneous bit-width, such as single precision quantization, is widely supported by hardware. Nevertheless, employing single precision for all weights and quantizations restricts the ability to identify the best quantization parameters for individual layers, or even for specific weights and quantizations within each layer.

Recent studies have also delved into incorporating heterogeneous bit-width or mixed-precision techniques. Mixed-precision quantization involves using different levels of precision for various parts of a model (such as layers and activations), providing a customized way to optimize performance while managing computational requirements. Nonetheless, this approach increases the intricacy of the quantized model since additional arithmetic operations are needed to handle conversions between different precisions within the NN graph.

2.3.3.2 Weight quantization

Weight quantization aims to reduce the precision of the weights of NN layers, typically from floating-point to lower-bit-width formats like integer representations. This process significantly reduces the size and computational complexity of the model. Since weight parameters account for the majority of the memory requirements of a NN, and are involved in the most expensive computations and data transfer operations during runtime. The benefits of weight quantization include improved model efficiency, with faster inference times and lower power consumption.

However, there are several considerations and relevant aspects to consider when implementing weight quantization. One of the main challenges is managing the trade-off between model size/computational efficiency and the accuracy of the model. Quantization can lead to a loss in precision, which, in some cases, may result in a decrease in model performance. This is especially critical for the quantization of weights because weights are where NNs store their learned knowledge, and each layer may have different data distributions and information density. Therefore, the choice of quantization schemes and parameters may affect layers differently.

Weights are typically quantized without requiring calibration data (GHOLAMI et al., 2022). Symmetric quantization is commonly used to quantize weights because setting the zero point to zero can decrease the computational cost during inference (WU et al., 2020b).

2.3.3.3 Bias quantization

The bias account for only a small fraction of the parameters in a NN. However, as each entry in a quantized bias vector is added to many of these output activations,

meaning that even small errors in the quantized bias vector can have widespread effects on the NN's outputs (JACOB et al., 2017). Quantization errors do not just introduce random noise but can systematically skew the output activations in one direction or another, i.e., they introduce an overall bias or an error component with a non-zero average (JACOB et al., 2017). So is imperative to evaluate and apply the right quantization scheme for bias. Another thing to consider is that, when quantizing both weights and activations (inputs to the layers), it's crucial to align their scale factors to ensure that the quantized dot product (between weights and activations) remains accurate. This alignment affects how biases are quantized and re-scaled, making it necessary to carefully manage the scale factors for biases to maintain overall NN performance.

There are several approaches and recommendations for bias quantization. Bias is often quantized and stored with higher precision, such as a 32-bit (NAGEL et al., 2021; JACOB et al., 2017). Moreover, employing an increase in precision for bias vectors serves a crucial purpose. It helps to prevent errors in the quantization of the bias vector from skewing activations output (JACOB et al., 2017). And also facilitates compatibility with the MAC operations in fixed-point arithmetic. In another direction, asymmetric quantization can be more beneficial, since it can better accommodate the range of bias values without loss of information.

The approaches and recommendations discussed above inherently add complexity to the bias quantization scheme. As they add many more factors to consideration. A simpler approach also supported by the literature is called bias absorption (JACOB et al., 2017; BHALGAT et al., 2020). In this approach, the bias is absorbed into the weights matrix, allowing for a more uniform treatment of the quantized parameters. That is, weights and bias have the quantization parameters. It should be noted that this approach may require adjustment of the subsequent layers to account for the absorbed bias(NAGEL et al., 2019).

2.3.3.4 Activation quantization

Usually, an overall recommendation is that unsigned symmetric quantization is well suited for one-tailed distributions, such as ReLU activations. In contrast, hyperbolic tangent activations should rely on signed symmetric quantization. Anyway, both strategies may be applied if an activation function is suitably modified.

2.4 Neural network equivalence

Neural network equivalence (NNE) refers to the concept that different NN architectures or configurations can produce the same output for a given input. Understanding equivalence in NNs is crucial for several reasons, including model optimization, simplifi-

cation, and interpretability. In this section we will explore the various aspects of NNE, including mathematical foundations, types of equivalence, and implications for ML.

At its core, NNE is rooted in the mathematical properties of functions and transformations. A NN can be viewed as a complex function composed of simpler, parameterized functions (e.g., activation functions, weight matrices). Mathematically, let $f : \mathcal{I} \to \mathcal{O}$ and $f' : \mathcal{I} \to \mathcal{O}$ be two arbitrary NNs, where $\mathcal{I} \in \mathbb{R}^n$ and $\mathcal{O} \in \mathbb{R}^m$ are their common input and output spaces, respectively. Thus, the two NNs f and f' are considered equivalent if:

$$f(x) \backsimeq f'(x)$$

, for all inputs x in the input domain of the function. Here, $\backsimeq$ defines a proper equivalence relation.

In this thesis we focus on functional equivalence which extends the concept of NNE by emphasizing the identical output behavior of different NNs under various transformations. This concept is particularly important in theoretical computer science and ML for understanding the foundational aspects of NN behavior and transformations. Research often explores how small changes in NN structure or weights can lead to functionally equivalent NNs.

Research on NN optimization and transformations frequently investigates conditions under which functionally equivalent models can be derived. Notable studies by LeCun et al. (CUN; DENKER; SOLLA, 1990) and Han et al. (HAN et al., 2015) have significantly contributed to understanding and achieving functional equivalence in NNs. These studies highlight methods to reduce model complexity while retaining the same functional behavior, which is crucial for efficient deployment and scalability of NN models.

Next, we will provide the definition and discuss some types of NNE.

2.4.1 Types of functional equivalence

There are several types of equivalence in NNs, each with distinct implications and applications. Currently, the literature reports the following definitions of types of equivalence (ELEFTHERIADIS et al., 2022; TEUBER et al., 2021; BÜNING; KERN; SINZ, 2020; PAULSEN et al., 2020; PAULSEN; WANG; WANG, 2020):

Strict equivalence

Strict equivalence (SE) occurs when two NNs produce the exact same output for every possible input. This requires that for any given input, the output values are identical across both NNs. This is the strongest form of equivalence, implying that the NNs implement the same function, despite potentially having different internal structures or parameter values. SE is considered a type of functional equivalence because it pertains to the exact

numerical match of the outputs, ensuring that both networks behave identically for all inputs.

Definition 2.4.1 (Strict equivalence) *Consider two NNs $f : \mathcal{I} \to \mathcal{O}$ and $f' : \mathcal{I} \to \mathcal{O}$. Then, f and f' are strictly equivalent, i.e., $f \equiv f'$, if and only if the following holds:*

$$\forall\ x \in \mathcal{I}, f(x) = f'(x). \tag{2.14}$$

SE is crucial in scenarios where any deviation in outputs is unacceptable. This form of equivalence is necessary when the exact output values are critical to the application, and any discrepancy could lead to significant errors or failures. While SE is a rigorous form of verification that ensures two NNs produce identical outputs for every input, it often proves impractical in real-world applications for several reasons. For example, achieving and verifying SE can be computationally expensive and time-consuming which may not be practical. For such applications, approximate forms of equivalence may be necessary and useful.

Classification based equivalence

Classification based equivalence (CBE) occurs when two NNs produce the same class prediction for every possible input. This means that, for each input, the index of the maximum value in the output layer (i.e., the class label with the highest probability) is identical for both networks. This is a more relaxed type of equivalence, that is, it is an approximate form of equivalence as it does not require the actual output values to be the same, only that they lead to the same classification decision.

Definition 2.4.2 (Classification based equivalence) *Consider two NNs $f : \mathcal{I} \to \mathcal{O}$ and $f' : \mathcal{I} \to \mathcal{O}$. Then, f and f' are equivalent, i.e., $f \approx_1 f'$, if and only if the following holds:*

$$\forall\ x \in \mathcal{I}, argmax_1 f(x) = argmax_1 f'(x). \tag{2.15}$$

CBE is particularly useful, as the name suggests, in classification tasks where the exact output values are less important than the final classification decision. In many applications, requiring exact output matches is unnecessarily stringent. Often, allowing the outputs to be close enough instead of exact matches is sufficient to ensure the networks perform the desired task effectively. For instance, in image recognition tasks, it is often sufficient to ensure that two different NNs identify the same object in an image. This is crucial in applications like medical image analysis, where different models might be used to ensure consistency in diagnosis, or in transfer learning scenarios, where a model fine-tuned on a specific dataset should produce the same classifications as the original model.

Norm based equivalence

Norm based equivalence (NBE) occurs when the outputs of two NNs are close to each other within a specified tolerance ϵ. This means that, for every input, the difference between the outputs of the two networks, measured by a chosen norm (typically the L_p norm), does not exceed ϵ. NBE is a more flexible form of equivalence compared to SE and allows for minor variations in the outputs, thus also bein an approximate form of equivalence. It is considered a type of functional equivalence because it pertains to the numerical similarity of the functions represented by the networks. Common choices of norms are: the Manhattan norm L_1 where $p = 1$; the Euclidean norm L_2 where $p = 2$; and the infinity norm L_∞ where $p = \infty$.

Definition 2.4.3 (NBE) *Consider two NNs $f : \mathcal{I} \to \mathcal{O}$ and $f' : \mathcal{I} \to \mathcal{O}$, $p \in \{1, 2, \infty\}$, and an $\epsilon > 0$. Then, f and f' are equivalent, i.e., $f \backsim_{p,\epsilon} f'$, if and only if the following holds:*

$$\forall\ x \in \mathcal{I}, ||f(x) - f'(x)||_p \leq \epsilon. \tag{2.16}$$

NBE is useful in scenarios where minor differences in output values are acceptable and do not significantly impact the overall performance or behavior of the network. This is common in regression tasks, generative models, and scenarios involving numerical approximations. For example, in weather prediction models, slight differences in output temperatures might be tolerable as long as they fall within a certain acceptable range.

In addition, CBE, although being a relaxed form of SE, also imposes a hard requirement but not as stringent or inflexible as in the case of the latter. NBE, in turn, is a flexible form of equivalence and often does not imposes such a hard requirement. As noted by Eleftheriadis et al.(ELEFTHERIADIS et al., 2022), SE is a true equivalence relation, that is, it is reflexive ($f \equiv f$ for any NN f), symmetric ($f \equiv f'$ iff $f' \equiv f$), and transitive ($f \equiv f'$ and $f' \equiv f''$ implies $f \equiv f''$). However, approximate forms of equivalence are only reflexive and symmetric and may not always satisfy the transitivity property.

2.4.2 Formal verification of NN equivalence

Verifying NN equivalence using formal verification techniques involves defining the NNE verification problem, where the objective is determining if two NNs, denoted as f and f', produce outputs that are equivalent according to a specified equivalence relation, for all inputs within a given input set I. The equivalence relation, denoted by $\backsimeq$, can represent SE ($\equiv$), as CBE ($\approx_1$), or NBE with a norm parameter p and within a tolerance ϵ ($\backsim_{p,\epsilon}$). The task is to verify whether $f(x) \backsimeq f'(x)$ holds true for every x in the input set I. Then, definition of formal verification of NN equivalence is as follows.

Definition 2.4.4 (NN equivalence verification problem) *Given two NNs f and f', an equivalence relation $\backsimeq \in \{\equiv, \approx_1, \backsim_{p,\epsilon}\}$, and the parameters p and ϵ, the NN equivalence verification problem consists in checking if $f(x) \backsimeq f'(x), \forall x \in \mathcal{I}$.*

The definition of equivalence can vary based on different criteria and parameters. And in practice, checking the equivalence for all possible inputs in the input domain $\mathcal{I}$ is infeasible. Often, an input space sampling strategy is employed to generate a representative set of inputs for which the outputs will be compared. This can be done through uniform sampling, importance sampling, or other techniques. In addition, it is common practice to express the input space in terms of input regions defined as bounded regions around a specific input. The effective manner this is achieved depends on the specific semantics of each formal method and will be explained in the following sections.

In this thesis, we use two formal methods to solve this equivalence verification problem: bounded model checking (BMC) and geometric path enumeration (GPE). BMC is combined with satifiability modulo theory (SMT) and GPE is combined with reachability analysis (RA). With SMT, the equivalence property and the NN model are encoded as a first-order logic formula. SMT restricts the full expressive power of first-order logic to a decidable fragment. With GPE, the property to be checked and the model are encoded as linear constraints.

2.4.3 SMT Encoding

The following steps show how to reduce NN equivalence problem to a logical satisfiability problem: (1) encoding f into an SMT formula ϕ; (2) encoding f' into an SMT formula ϕ'; (3) encoding the relation $f \backsimeq f'$ into an SMT formula Φ, such that $f \backsimeq f'$ iff Φ is not satisfiable; and (4) checking, via SMT solver, whether Φ is satisfiable. If the latter is true, f and f' are not equivalent, and the solver provides a counterexample. Otherwise, f and f' are equivalent.

Checking NN equivalence problem becomes possible by relying on the negation of $f \backsimeq f'$, i.e., by encoding it as a formula that asserts the existence of an input $x \in \mathcal{I}$ and two outputs $y, y' \in \mathcal{O}$ where ($y = f(x)$ and $y' = f'(x)$) such that they do not satisfy the conditions imposed by $\backsimeq$. Indeed, we check if

$$\exists x \in \mathcal{I}; y, y' \in \mathcal{O}; y = f(x) \wedge y' = f'(x) \wedge y \neq y',$$

in the SE case. For NBE, we check if

$$\exists x \in \mathcal{I}; y, y' \in \mathcal{O}; y = f(x) \wedge y' = f'(x) \wedge ||y - y'||_p > \epsilon,$$

and for the CBE we check if

$$\exists x \in \mathcal{I}; y, y' \in \mathcal{O}; y = f(x) \wedge y' = f'(x) \wedge \text{argmax}_1\ y \neq \text{argmax}_1\ y',$$

In summary, checking whether the formula $y = f(x) \wedge y' = f'(x) \wedge y \neq y'$ is unsatisfiable can be further expressed as

$$\begin{aligned} \phi &:= y = f(x) \\ \phi' &:= y' = f'(x) \\ \Phi &:= \phi \wedge \phi' \wedge y \neq y'. \end{aligned} \tag{2.17}$$

Similarly, checking whether the formula $y = f(x) \wedge y' = f'(x) \wedge ||y - y'||_p > \epsilon$ is unsatisfiable can be expressed as

$$\begin{aligned} \phi &:= y = f(x) \\ \phi' &:= y' = f'(x) \\ \Phi &:= \phi \wedge \phi' \wedge ||y - y'||_p > \epsilon. \end{aligned} \tag{2.18}$$

In addition, the input of an NN f is a vector $\mathbf{x} = \{x_1, ..., x_n\} \in \mathbb{R}^n$, and some limitation regarding it may be necessary. This constraint can then be added as

$$\bigwedge_{j=1}^{n} x_j - \delta \leq x_j \leq x_j + \delta. \tag{2.19}$$

In practice, this defines a limiting region between $x_j - \delta$ and $x_j + \delta$ around every point $x_j \in \mathbf{x}$, where equivalence is more likely. It works as another relaxation factor for equivalence because the associated properties should hold only for a restricted input domain. It is also corroborated by the notion that we expect the same behavior from a close neighbor of an input. In addition, it can also be linked to real conditions of a given application, such as its equivalence method and input deviation and magnitude. However, this does not mean that our technique is limited to small input ranges. Instead, it is important to choose a range that preserves the relationship between x and y and is also according to a specific application.

2.4.4 GPE encoding

Tran et al. (TRAN et al., 2019a) proposed GPE, a formal method for verifying NNs' safety properties and later expanded by Teuber et al. (TEUBER et al., 2021) for the verification of NN equivalence properties. This section provides a brief description how it encodes NNs and equivalence property into a verification problem. The complete explanation regarding GPE can be found in its original paper (TEUBER et al., 2021).

Definition 2.4.5 (Generalized Star Set (TRAN et al., 2019a)) *A generalized start set Θ is a tuple $\langle c, G, P \rangle$ where $c \in \mathbb{R}^n$ is the center, $G = (g_1, ..., g_m) \in \mathbb{R}^{n \times m}$ is the generator matrix, and $P \subseteq \mathbb{R}^m$ is a polytope defining a conjunction of linear constraints. The set represented by Θ is then defined as*

$$\Theta = \{x \in \mathbb{R}^n \mid \exists \alpha \in P : x = c + G\alpha\}.$$

Assume there are two NNs f and g representing piecewise linear functions. Furthermore, assume that the goals is to verify whether $f(x) = g(x)$ for the input domain $\mathcal{I} \equiv \langle\ c, G, P\rangle$. Since f and g are piecewise linear, there exist a tiling $\mathcal{T} \equiv \{P'\}$ of the input domain $\mathcal{I}$ such that $\mathcal{I} = \{x \in \mathbb{R}^n \mid \exists P' \in \mathcal{T} \wedge \alpha \in P' : x = c + G\alpha\}$. Moreover, for each tile $P' \in \mathcal{T}$, we require both f and g to be linear, i.e., $f(x) = c_f + G_f\alpha$ and $g(x) = c_g + G_g\alpha$ for $\alpha \in P'$, where (c_f, c_g, G_f, G_g) are specific to each tile P'. The NNs f and g are equivalent if $c_f = c_g$ and $G_f = G_g$ for each tile P'.

The GPE method computes the tiling $\mathcal{T}$ via repeated reachability analysis as follows. First, the input domain $\mathcal{I}$ is propagated through NN f, outputting a union of star sets. Then, each set is projected back onto the input space and propagated through NN g, leading to a further union of star sets whose elements, once projected onto the input domain, represent the tiles $P' \in \mathcal{T}$.

2.5 Summary

In this chapter, we provided the foundational background theory necessary to understand the research conducted in the dissertation. We covered the basics of neural networks (NNs), the process and mathematical framework of quantization, and the principles of neural network equivalence verification. Additionally, optimization techniques relevant to NN quantization were discussed, and concluded with a formal definition of the research problem.

Our discussion on NNs began with an overview of their structure, which includes input, hidden, and output layers, and explains their role in modeling complex, non-linear relationships. NNs were mathematically formalized as multi-layered functions, with detailed explanations of layer transformations and activation functions such as ReLU, sigmoid, and tanh.

Quantization, a process of mapping high-precision values to lower-precision representations, was explored in detail, with focus on key parameters such as bit-width, scale factor, and zero-point. We presented various quantization methods, including symmetric and asymmetric quantization, and discussed their trade-offs in terms of precision and computational efficiency. The role of quantization in NNs was described, with attention to weights, biases, and activations, and strategies like mixed-precision and bias absorption are highlighted.

Finally, neural network equivalence (NNE), which examines when different NNs produce the same output for given inputs, was introduced. We defined strict equivalence, classification-based equivalence, and norm-based equivalence, providing formal mathematical definitions and examples. Techniques for verifying equivalence, such as satisfiability modulo theories (SMT) and geometric path enumeration (GPE), were presented as formal

methods used to solve the NNE verification problem.

3 Counter-Example Guided Neural Network Quantization Refinement (CEG4N)

3.1 Introduction

Techniques for NNs quantization have become a crucial method for deploying advanced ML models on devices with limited resources, such as mobile devices, embedded systems, and Internet of Things (IoT) devices. Although quantization reduces the memory usage and computational demands of NNs, it introduces challenges in maintaining the functional performance of the NN, particularly in fields where accuracy and correct behavior are critical. For example, many embedded systems have safety requirements which makes NNs with erratic or unreliable behavior unacceptable. Therefore, ensuring that the quantized network retains the same functionality as the original network is essential.

NN behavior, as described in this thesis, refers to the specific operational patterns or responses exhibited by a NN f when given a particular set of inputs x. This behavior is identified by the relationship that maps the input data points to their respective outputs, where each output y is represented as $y = f(x)$. To maintain consistent behavior, a NN must generate the same or nearly identical outputs for the same inputs, even after undergoing transformations such as quantization. Thus, in the context of quantization, the behavior of a NN remains intact if, for every input x within a specified vicinity around a critical data point x', both the original NN f and its quantized counterpart f' produce equivalent outputs, meeting the condition $f(x) = f'(x)$. This parity ensures that the fundamental operational characteristics of the NN are preserved, notwithstanding the decrease in precision resulting from quantization. The deployment of NNs in resource-constrained environments necessitates the application of quantization techniques to reduce the model's precision from floating-point (e.g., 32 bits) to lower-bit representations. However, this transformation is known to introduce discrepancies to the numerical operations of an NN, altering its functional behavior, and potentially leading to deviations in output for given inputs. In domains where the specific behavior of a NN is crucial, such as medical diagnostics, autonomous driving, and financial forecasting, such deviations are unacceptable. Therefore, there is a need for methods that allow quantization without affecting the network's essential functionality. The challenge, therefore, lies in quantizing the NN (denoted as f) into a lower-precision version f' without altering its behavior for a defined set of critical inputs and their surroundings.

In this chapter, we are going to discuss the problem of guaranteeing the behavioral equivalence between a full-precision NN and its quantized version, and how to address it.

Specifically, we are going to formalize the problem and the quantization method that offers formal guarantees on behavioral equivalence, discuss the our fundamental assumptions, and the important details and aspects of our proposed method.

3.2 Equivalent Quantization Problem

Given a NN (NN) $f : \mathbb{R}^n \to \mathbb{R}^m$ with L layers, and its quantized version $f' : \mathbb{R}^n \to \mathbb{R}^m$, our objective is to find a quantization scheme that ensures the outputs of f and f' are equivalent over a specified set of critical input regions, while minimizing the cost associated with the quantization (e.g., reducing the bit widths of its parameters).

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_c\} \subset \mathbb{R}^n$ denote a set with C critical data points. Around each critical point $x_c = x_i, 1 \leq i \leq c$, we define a region $\mathcal{R}_{c,\delta}$ as the set of points within a radius δ:

$$\mathcal{R}_{c,\delta} = \{x \in \mathbb{R}^n \mid \|x - x_c\| \leq \delta\}.$$

Let $\mathcal{R}_\delta = \bigcup_{c=1}^{C} \mathcal{R}_{c,\delta}$ be the union of all such regions. Our goal is to ensure that the outputs of f and f' are equivalent for all inputs in $\mathcal{R}_\delta$. We define equivalence in terms of an equivalence relation $\backsimeq$ between the outputs of two NNs, which could represent exact equality ($\equiv$) or approximate equality within a tolerance ($\sim_{p,\epsilon}$). Specifically, for all $x \in \mathcal{R}_\delta$, we require satisfy the following constraint:

$$f(x) \backsimeq f'(x).$$

Let $\mathbf{b} = (b_1, b_2, \ldots, b_L)$ be a vector where each b_l denotes the bit width used to quantize the parameters of layer l in the network f. Additionally, consider the constants b_{min} and b_{max} to be the minimum and maximum bit widths allowed for quantization. They define the lower and upper bounds for the quantization bit width, which guarantee correctness for the quantization process and the generation of valid quantized models. They can also be regarded as initialization and termination criteria; as a feasible $\mathbf{b}$ can not be such that every b_l in it are equal to either the upper or the lower bounds, the optimization process must stop, as no valid quantized model can be generated outside them. The quantization process is represented by a function $\mathcal{Q}$ that maps the original network f and the bit width vector $\mathbf{b}$ to the quantized network f':

$$f' = \mathcal{Q}(f, \mathbf{b}).$$

We introduce a cost function $\mathcal{J}(\mathbf{b})$ that quantifies the cost associated with the quantization parameters $\mathbf{b}$. This cost could represent some metric related to the number of bits to represent the quantized NN weights or a combination of other factors such as memory usage, computational efficiency, and energy consumption.

To formally ensure the equivalence between f and f' over the critical regions $\mathcal{R}_\delta$, we define a constraint function $\mathcal{G}(f, f', \mathcal{R}_\delta)$. This function has an underlying formal technique that solves the NN equivalence problem defined as $f(x) \backsimeq f'(x)$ for all $x \in \mathcal{R}_\delta$, and returns a set of counterexamples $\mathcal{X}_\delta$. If the equivalence constraint is satisfied, $\mathcal{G}$ returns an empty set (i.e., $\mathcal{X}_\delta = \emptyset$), and $\mathcal{X}_\delta \neq \emptyset$ otherwise. The constraint in the optimization can be defined as:

$$\mathcal{G}(f, f', \mathcal{R}_\delta) = \begin{cases} \mathcal{X}_\delta = \emptyset, & \text{if } f(x) \backsimeq f'(x) \text{ for all } x \in \mathcal{R}_\delta, \\ \mathcal{X}_\delta \neq \emptyset, & \text{otherwise.} \end{cases}$$

Our optimization problem is thus formulated as:

$$\begin{aligned} \text{minimize} \quad & \mathcal{J}(\mathbf{b}) \\ \text{subject to} \quad & \mathcal{G}(f, f', \mathcal{R}_\delta) = \mathcal{X}_\delta = \emptyset \\ & f' = \mathcal{Q}(f, \mathbf{b}) \\ & \mathbf{b}_o = \{b_l | b_{\min} \leq b_l \leq b_{\max}, \forall l = 1, 2, \ldots, L\} \end{aligned} \tag{3.1}$$

The goal is to find a bit width vector $\mathbf{b}$ that minimizes the cost $\mathcal{J}(\mathbf{b})$ while ensuring that the quantized network f' remains equivalent to the original network f over the critical regions $\mathcal{R}_\delta$. The constraint $\|f(x) - f'(x)\| \leq \epsilon$ for all $x \in \mathcal{R}_\delta$ ensures that the quantized network f' approximates the original network f within a specified tolerance over the regions of interest. The function $\mathcal{J}(\mathbf{b})$ captures the trade-off between quantization precision and cost, encouraging the use of lower bit widths to reduce resource consumption while maintaining acceptable performance. The quantization function $\mathcal{Q}$ applies quantization to the weights and biases of each layer in f. The bit widths $\mathbf{b}$ determine the precision of quantization for each layer, influencing both the approximation error and the resource usage of the quantized network f'.

This formalization provides a precise mathematical framework for the problem of EQ of NNs. We defined it as an optimization problem, where the task is minimizing the quantization bit widths $\mathbf{b}$ under the constraint that the quantized network f' behaves equivalently to f within specified regions of interest. Solving this optimization problem involves determining the bit widths $\mathbf{b}$ that achieve the desired trade-off between resource efficiency and functional equivalence.

3.2.1 Discussion on the assumptions

This method relies on having a comprehensive and representative set of critical points $\mathcal{X}$ that reflects the fundamental behavioral traits of the NN f. It also supposes that for every data point in $\mathcal{X}$, there is a designated area where the NN's behavior remains stable. These assumptions are crucial as they suggest a thorough understanding of the specific domain and the NN's behavior, which may not always be entirely achievable.

The optimization problem described here is inherently complex due to several factors. Firstly, the quantization process involves discrete decision-making, leading to a non-convex optimization landscape. Secondly, the formal equivalence checking function $\mathcal{G}$ adds a layer of computational complexity, as it requires the verification the behaviors of f and f' across potentially infinite number of points within the defined set of critical input regions $\mathcal{R}_\delta$. In other words, equivalence verification must cover a large search space. Lastly, the need to balance the trade-off between the quantization granularity (determined by $\mathbf{b}$) and the preservation of behavioral integrity makes the optimization problem challenging to solve efficiently. The complexity of this optimization problem may require advanced algorithms, possibly involving heuristic or approximation methods, to find a solution that is practically acceptable within reasonable computational times.

The implication is that quantizing NNs while maintaining their functional characteristics poses a complex challenge that involves mathematical, computational, and domain-specific factors. Framing this issue as an optimization problem highlights the complexities of achieving quantization without sacrificing the behavior integrity of the quantized NNs. The assumptions emphasize the importance of expertise in the field and the thoughtful choice of essential data points and their respective vicinity areas, while the intricacy of the optimization process underscores the requirement for advanced algorithms that can handle the trade-off between efficient resource utilization and preserving the network's behavior.

3.2.2 Formal equivalence verification in neural networks quantization

The equivalence verification problem, represented by the function $\mathcal{G}$, plays a pivotal role in ensuring that a quantized neural network f' retains the functional behavior of the original full-precision network f. The formal equivalence verification task involves checking whether $f(x) \backsimeq f'(x)$ holds for all $x \in \mathcal{R}_\delta$, a requirement that translates into a formal specification suitable for computational solvers.

Formal methods provide a rigorous framework for analyzing the equivalence problem by transforming it into a set of specifications that a solver can verify. The key elements in this transformation include:

- A neural network f that operates in floating-point precision, i.e., the original model, where $f(x)$ denotes the output for input x;
- The network f transformed into a quantized version f', i.e., the quantized model, using the quantization function $\mathcal{Q}$;
- The input space around critical points, i.e., the critical Regions $\mathcal{R}_\delta$, where equivalence must be verified;

- The equivalence relation $\backsimeq$ that defines the acceptable level of agreement, ranging from exact equality ($f(x) = f'(x)$) to approximate equivalence within a tolerance ($|f(x) - f'(x)| \leq \epsilon$).

These elements are encoded as formal specifications, which solvers use to exhaustively check the behavior of f and f' over $\mathcal{R}_\delta$. A failure to meet the equivalence criteria results in counterexamples where the behavior diverges.

Two prominent formal verification techniques, Bounded Model Checking (BMC) and Geometric Path Enumeration, have been employed in the context of neural network verification. BMC translates the verification problem into a satisfiability problem, represented as a set of logical formulas over a bounded domain. Solvers, such as SAT/SMT solvers, explore this domain to identify counterexamples. BMC has been widely used to verify neural network properties, but its application to verifying equivalence between networks with different precisions remains unexplored. Geometric path enumeration, in turn, focuses on decomposing the input space into regions defined by the activation patterns of neurons. Each region is treated as a geometric polytope where the network behaves as a linear function. While effective for exact equivalence checking between networks, it does not extend to networks with mixed or differing precisions, such as floating-point and quantized networks.

Applying FEV to NNs with different precisions introduces significant challenges. First we have to consider the possibility of not being able to correctly encode and express quantization semantics in the verification model. Quantized models often involve non-linear operations like clipping, rounding, and re-scaling, which are difficult to model mathematically. Often requiring approximation to be considered, which increases the chances of the inconsistencies between the concrete quantized model and the model being verified. Another point is the complexity of certain quantization approaches, for example, mixed-precision quantization where the quantized networks can have layers of varying precisions (e.g., 8-bit weights and floating-point activations). This would require encoding diverse arithmetic behaviors within a unified verification framework. And we must consider the error propagation in quantized models. The quantization errors propagate through layers, complicating the analysis of cumulative effects on the network's output. This become even more challenging if the quantization schemes apply some technique to correct and minimized the errors being propagated.

To address these challenges, we propose a novel approach: once the original network f is quantized to f', we de-quantize f' into a floating-point network f'' and perform equivalence checking between f and f''. De-quantization involves translating quantized weights and operations back into their floating-point representations. This approach simplifies verification by ensuring that the equivalence problem is framed entirely within the

domain of floating-point arithmetic. The validity of this strategy relies on the preservation of quantization semantics during de-quantization. Specifically:

- quantized weights are mapped to floating-point values while retaining the same numerical representation within their quantization range;
- any inaccuracies introduced during quantization are captured in f'', ensuring that f'' faithfully reflects the behavior of f' under the original quantization constraints;
- the process guarantees that re-quantizing f'' reproduces f' without additional errors.

By using f'' as a proxy for f' during verification, we sidestep the complexities of quantized arithmetic while maintaining the integrity of the equivalence problem. Henceforth, references to quantized networks in the context of equivalence verification pertain to the de-quantized network representation.

De-quantization can be rigorously designed to ensure that each quantized value is mapped back to its exact floating-point equivalent within the quantization range. This process relies on deterministic mappings, where the boundaries of quantization levels are precisely defined. Any discrepancies arising during quantization (e.g., rounding errors) are inherently embedded in f'. By directly translating these effects into f'', we ensure that f'' mirrors f' in terms of behavior. A carefully designed de-quantization step, validated against edge cases (e.g., extreme weights or activations), can minimize the risk of numerical instability. Empirical tests on standard benchmarks can further substantiate this claim. By design, de-quantization does not alter the underlying computation but translates it into a domain (floating-point arithmetic) that enables equivalence checking. This ensures that errors are neither corrected nor amplified. Techniques such as interval arithmetic or sensitivity analysis can be used to formally verify that f'' captures the same error profile as f'. These methods provide a robust mathematical framework for quantifying error preservation. The proposed method can be tested on networks with known quantization artifacts. Validation of consistent behavior across f' and f'' would demonstrate the efficacy of the approach.

Uniform symmetric post-training quantization, in simple terms, is rounding to the nearest point of an evenly-spaced lattice. Hence the only discrepancy between a stored integer weight and its reconstructed floating-point value is the rounding residue.

Lemma 3.2.1 *Given a uniform symmetric quantiser with scale s and zero-point z, the mapping*

$$Q^{-1}(Q(w)) \;=\; w + \varepsilon_w$$

satisfies $|\varepsilon_w| \leq s/2$ for every weight w. Consequently, for any layer the de-quantised weight matrix differs from the realised integer matrix by at most $s/2$ in each entry.

Corollary 3.2.2 (Layer-wise propagation error) *Let a linear layer act on an input vector x as $y = Wx + b$. Assume the integer implementation stores $\hat{W} = Q(W)$ and de-quantises it to $W' = Q^{-1}(\hat{W}) = W + E$, where every entry of E obeys $|E_{ij}| \le s/2$ by Lemma 3.2.1. Then the output produced by W' differs from the exact-real-weight output by*

$$\|Ex\|_\infty \;\le\; \frac{s}{2}\,\|x\|_1,$$

so the discrepancy is at most $(s/2)\sum_j |x_j|$. Because the quantiser scale s is a calibrated constant, this bound can be propagated layer-by-layer (via each layer's operator norm or by additive accumulation for residual links) to obtain a closed-form network-level error that is typically many orders of magnitude smaller than the numerical tolerances assumed in existing safety-verification tools. Thus verifying the floating-point surrogate f'' is sound: every concrete execution of the integer circuit f' lies within the norm-ball proven for f''.

A similar "half-step" reconstruction bound is explicitly invoked for neural networks by Jacob et al. 2018 (JACOB et al., 2017) and Banner et al. 2019 (BANNER; NAHSHAN; SOUDRY, 2019). Layer-wise error-propagation arguments in the same spirit are presented by Lee et al. (LEE; JEONG; BAE, 2019) and by Li et al.(LI et al., 2022).

For every input $x \in \mathcal{R}_\delta$ we wish to guarantee

$$\|f(x) - f'(x)\| \;\le\; \varepsilon.$$

This is ensured if the following two inequalities hold:

$$\|f(x) - f''(x)\| \;\le\; \varepsilon - \alpha \quad \text{and} \quad \|f'(x) - f''(x)\| \;\le\; \alpha,$$

because their conjunction implies the desired bound by the triangle inequality.

The constant

$$\alpha \;=\; \sup_{x \in \mathcal{R}_\delta} \|f'(x) - f''(x)\|$$

is the *network-level half-step reconstruction error*. It is obtained analytically—without any solver intervention—by propagating Lemma 3.2.1 and Corollary 3.2.2 through the network: each linear or convolutional layer ℓ with quantiser scale s_ℓ contributes at most $(s_\ell/2)\,\|x\|_1$ at its input, and these contributions are composed along the forward path (and added at residual junctions) to yield a closed-form, data-independent bound α.

Since α is fixed once the network architecture and quantiser scales are fixed, the verification problem presented to the solver reduces to proving the tighter condition $\|f(x) - f''(x)\| \le \varepsilon - \alpha$. The second inequality, $\|f'(x) - f''(x)\| \le \alpha$, is already guaranteed by the analytical error bound, thereby justifying the sound use of the floating-point surrogate f'' in place of the integer circuit f' during formal verification.

The reversibility of the mapping can be established through formal verification methods, ensuring that every floating-point representation in f'' maps directly back to

its quantized counterpart in f' within the same quantization range. Special cases such as saturation or clipping can be explicitly modeled in the de-quantization step to ensure that they are preserved. For instance, introducing flags or metadata for saturated values can help maintain consistency during re-quantization. Extensive testing on diverse network architectures and quantization schemes (e.g., uniform, non-uniform) would reinforce confidence in the reversibility claim. For instance, experiments could confirm that re-quantizing f'' consistently yields f' across different datasets and configurations.

Tools and algorithms for verifying floating-point operations are well-established and highly optimized. By framing the problem entirely within this domain, the proposed method leverages these mature resources. Directly verifying f'' against f' often requires reasoning across two different arithmetic domains (floating-point and quantized). This cross-domain complexity can lead to inefficiencies and increased verification times. Introducing f'' creates a structured intermediate step that aligns well with existing verification tools. Empirical comparisons of verification times and resource usage for the proposed method versus direct approaches can substantiate its practicality.

The de-quantization process can be extended to handle non-uniform quantization by incorporating the specific quantization parameters (e.g., non-linear scaling factors) into the translation step. The method is not tied to a particular quantization scheme but rather to the principles of reversible mapping and error preservation. These principles can be adapted to diverse quantization methods through appropriate design. Testing the approach on networks employing non-uniform or dynamic quantization can demonstrate its generality. Metrics such as fidelity, error consistency, and verification success rates can provide quantitative evidence of robustness.

This discussion establishes the foundation for ensuring equivalence between full-precision and quantized networks through de-quantization. The next section will delve into the specifics of the quantization scheme itself, outlining the methodology for scaling and parameterizing the quantization process to align with equivalence verification requirements.

3.2.3 Importance of quantization scheme choices

The proposed quantization scheme begins with a floating-point neural network (f) operating in 32-bit precision and systematically reduces the precision of its weights, biases, and activations to integers ranging from 2 to 32 bits. This reduction is achieved using uniform and symmetric quantization for weights and biases and uniform and asymmetric quantization for activations.

Quantization involves the following steps:

- precision assignment, where, for each NN layer, weights, biases, and activations share the same bit width, denoted by b_l, ensuring consistent precision throughout the layer;

- scaling, where each quantization operation uses scaling factors α and β, which depend on the target bit width and the range of values in the original network;
- input-layer quantization, where the input layer is quantized to minimize the risk of overflows and ensure bounded arithmetic in subsequent layers.

Quantization introduces challenges, particularly in the input-layer quantization step. While quantizing the input-layer ensures consistency across the remaining layers, it risks altering the representation of input features, which may affect the network's interpretability and performance. Careful calibration of scaling parameters and quantization ranges is necessary to mitigate these risks.

By aligning the quantization process with the equivalence verification framework, we ensure that the resulting quantized (and de-quantized) networks remain amenable to formal analysis. The next section will explore how this quantization scheme is integrated into an iterative optimization process to achieve behavioral equivalence while minimizing resource usage.

3.3 Iterative Quantization Framework

The complex task of quantizing NNs while preserving their functional behavior in resource-constrained environments presents significant computational challenges. Given the intricacies involved, an iterative quantization approach offers a viable strategy to address this challenge by decomposing the overarching quantization problem into two distinct, yet interrelated, sub-problems. In this section we discuss how to reformulate the optimization problem defined in (3.1) into an iterative quantization problem. This methodology entails, first, the optimization problem, focusing on identifying feasible quantization parameters, and second, the equivalence verification problem, which involves verifying the equivalence of the original and quantized networks over a set of critical input region $\mathcal{R}^b$. By iterating between these two problems, one can systematically navigate the optimization landscape, addressing both the efficiency of quantization and the preservation of functional behavior.

The iterative framework conceptualizes the quantization process as a cycle of optimization and formal equivalence verification. Initially, the optimization problem is tackled to find the quantization parameters $\mathbf{b}$ that reduce the size of the model and the computational requirements without having to consider the equivalence constraint. This step involves optimizing an objective function $\mathcal{J}(\mathbf{b})$, which measures the effectiveness of quantization in terms of resource efficiency. The challenge lies in navigating the parameter space to find a quantization strategy that minimizes resource consumption while potentially preserving the network's functional behavior intact.

Following the optimization phase, the focus shifts to the constraint problem, where the equivalence of the original and quantized networks is assessed. Given the set of quantization parameters $\mathbf{b}$ derived from the optimization phase, this step involves verifying whether the quantized network f' behaves equivalently to the original network f for all critical data points within $\mathcal{X}$ and their neighboring data points. This verification process is computationally intensive and constitutes significant complexity in the optimization problem. The difficulty of verifying equivalence across the entire set of critical data points underscores the necessity of an iterative approach, where each cycle refines the quantization parameters based on concrete results of equivalence checks.

3.3.1 Simplification of the NN equivalence constraints

To mitigate the computational burden of NN equivalence constraints in the optimization step, the iterative optimization framework incorporates a strategy to use concrete critical data inputs derived from the verification process to further constrain the optimization problem. Recalling the original optimization problem and the discussion in Section 3.1, the fundamental assumption is that there exists a collection of critical data points that represent the functional characteristics of the original NN f and that must be maintained in the quantized NN f'. Another assumption is the presence of a surrounding area near these critical points where the desired functional behavior is also evident.

Therefore, the goal of the NN equivalence verification process is to prove that two NNs share the same functional behavior for all critical data points and their surrounding area. This can ultimately be reduced to proving that there is no such data point in the surrounding area of a critical point for which the functional behavior of the two NNs is not the same. If such a data point exists, it serves as a proof of the falsehood of the equivalence. Thus, it is considered to be a counterexample. Having a counterexample is extremely important and relevant, since checking whether a counterexample is valid or not is significantly less computationally expensive than finding it. To the potential limitations of this simplification, such as the risk of not covering all critical input regions entirely. The idea is to leverage on existing formal equivalence verification techniques. Such techniques offer guarantees of soundness and completeness on the solution they provide. What gives assurance that all critical input regions entirely.

If the solver cannot prove or disprove the equivalence property, several conclusions can be drawn depending on the specific circumstances. One common issue is that the solver might run out of time or computational resources, resulting in an inconclusive result. To address this, additional computational resources or alternative verification methods might be considered.

The simplification of the NN equivalence constraints acknowledges that comparing the outputs of two NNs for specific inputs is substantially simpler than verifying their

equivalence for all possible inputs within the surrounding region of a critical data point. By integrating feedback from the verification process—specifically, the outcomes of comparing network outputs for selected critical inputs, that is, the counterexamples—into the optimization problem, the framework narrows the search space for the quantization parameters. In other words, the constraints imposed to the optimization become simpler and tractable. This feedback loop allows for a more focused and potentially more efficient optimization process, where the parameters are tuned not only for resource efficiency but also for behavioral preservation based on actual equivalence outcomes (STURSBERG et al., 2004).

The foundation of the proposed simplification is rooted in the field of formal methods. It is widely acknowledged that solving a verification problem is much more challenging than confirming the correctness of a concrete solution (OBERKAMPF; TRUCANO, 2002). Formal methods involve various approaches and resources to design, implement, and validate systems with the aim of demonstrating correctness with respect to a defined set of properties (HOFER-SCHMITZ; STOJANOVIć, 2020; URBAN; MIN'E, 2021). The contrast between the complexity of solving a verification problem and the simplicity of validating the correctness of a concrete solution can be understood through multiple conceptual and computational structures (MENDIAS et al., 2002). Let's re-define the simplification in the equivalence constraints as:

$$\mathcal{G}(f, f', \mathcal{R}_{\delta=0}) = \begin{cases} \mathcal{X}_{\delta=0} = \emptyset, & \text{if } f(x) \backsimeq f'(x) \text{ for all } x \in \mathcal{R}_{\delta=0}, \\ \mathcal{X}_{\delta=0} \neq \emptyset, & \text{otherwise.} \end{cases}$$

Note that we can simplify the equivalence constraints by simply forcing the δ to 0. Thus we can define $\mathcal{R}_{\delta=0}$ as:

$$\mathcal{R}_{\delta=0} = \bigcup_{c=1}^{C} \mathcal{R}_{c,\delta=0},$$

and since $\delta = 0$ we can define $\mathcal{R}_{c,\delta}$ as:

$$\mathcal{R}_{c,\delta} = \{x \mid x = x_c \wedge x_c \in \mathcal{X}\}.$$

That is, we made the set of critical regions $R^{\delta=0}$ be exactly the set of critical points $\mathcal{X}$. From a formal verification perspective, this is still valid verification problem, and way easier to solve. Another implication is that, if a formal method allows one to construct formal abstract models of the NNs f and f' with a high degree of fidelity, one could replace the need for a formal method to solve the equivalence problem in function $\mathcal{G}$, and work directly with the concrete models f and f'. Since the equivalence problem can now simply be solved by checking if the outputs of f and f' satisfy the equivalence relation. This way the optimization problem won't need to absorb and deal with the semantics of the formal equivalence verification method. Likewise, the formal equivalence verification method won't need to absorb and deal with the complexities of the optimization and quantization.

This is another advantage of our iterative quantization method, the quantization (i.e., the optimization problem in our method) and the equivalence verification can be solved by completely independent techniques and tools. As long as we are able to effectively construct and express the mathematical models of the NNs f and f' for both of the problems. That is, ensure the quantization and equivalence verification are working with the same NNs, to minimize discrepancies and imprecision.

3.3.2 Formalization

In this section, we reformulate the problem of equivalent quantization of neural networks (NNs) as an iterative optimization problem. The objective remains to find a quantization scheme that ensures the outputs of the original NN f and its quantized version f' are equivalent over specified critical input regions while minimizing the cost associated with the quantization. The iterative approach divides the problem into two complementary subproblems at each iteration. The firs, the **Bit Search Problem** defines the problem of finding a feasible bit width vector that satisfies the original optimization problem's objective and constraints. And second, the **Equivalence Verification Problem**, defines the problem of verifying the equivalence of f and f' with the bit width vector found, and update the critical input regions based on the verification results.

The formalization of our iterative method follows the notation provided in Section 3.2. Given a neural network $f : \mathbb{R}^n \to \mathbb{R}^m$ with L layers, we aim to find a quantization bit width vector $\mathbf{b}^o = (b_1^o, b_2^o, \ldots, b_L^o)$ that minimizes the quantization cost while ensuring equivalence over a set of critical input points $\mathcal{R}_{\delta=0}^o$ at iteration o. At each iteration o, the set of critical input regions $\mathcal{R}_{\delta=0}^o$ is defined as:

$$\mathcal{R}_{\delta=0}^o = \bigcup_{i=1}^{o-1} \mathcal{X}_{\delta=0}^i,$$

where $\mathcal{X}_{\delta=0}^i$ is the set of counterexamples generated from the Equivalence Verification Problem at iteration i. Initially, $\mathcal{R}_{\delta=0}^{o=0}$ may be an empty set or contain predefined critical points.

The Bit Search Problem is formulated as:

$$\begin{aligned} \min_{\mathbf{b}^o} \quad & \mathcal{J}(\mathbf{b^o}) \\ \text{s.t.} \quad & \mathcal{G}(f, f'_o, \mathcal{R}_{\delta=0}^o) \\ & f'_o = \mathcal{Q}(f, \mathbf{b}^o), \\ & b_{\min} \leq b_l^o \leq b_{\max}, \quad \forall l = 1, 2, \ldots, L. \end{aligned} \tag{3.2}$$

Here, $\mathcal{J}(\mathbf{b}^o)$ is the cost function associated with the bit widths $\mathbf{b}^o$, $\backsimeq$ denotes the equivalence relation between f and f'_o outputs, $\mathcal{Q}$ is the quantization function mapping f and $\mathbf{b}^o$ to f'_o, and $b_{\min}$ and $b_{\max}$ are the minimum and maximum allowable bit widths.

After obtaining $\mathbf{b}^o$ from the Bit Search Problem, the next step is the Equivalence Verification Problem is solved, whith the goal of formally verifying the equivalence of f and f'_o over the critical regions $\mathcal{R}_\delta$, where δ is a positive constant. Notice that $\delta > 0$, in the Equivalence Verification Problem the equivalence constraints are define over input region set, not concrete input sets as in Bit Search Problem. The critical regions $\mathcal{R}^0_\delta$ are defined as:

$$\mathcal{R}_\delta = \bigcup_{c=1}^{C} \mathcal{R}_{c,\delta},$$

with each region $\mathcal{R}_{c,\delta}$ centered at a critical point x_c:

$$\mathcal{R}_{c,\delta} = \{x \in \mathbb{R}^n \mid \|x - x_c\| \leq \delta\}.$$

The equivalence verification function $\mathcal{G}$ remains the same but we now label its outputs based on the iteration o, and is defined as:

$$\mathcal{G}(f, f'_o, \mathcal{R}_\delta) = \begin{cases} \emptyset, & \text{if } f(x) \backsimeq f'_o(x), \quad \forall x \in \mathcal{R}_\delta, \\ \mathcal{X}^o_\delta, & \text{otherwise,} \end{cases}$$

where $\mathcal{X}^o_\delta$ is the set of counterexamples found. The iterative process proceeds as follows:

Initialize $t \leftarrow 1$, $\mathcal{R}^{o=0}_\delta \leftarrow \emptyset$
Repeat `// Bit Search Problem`
Solve

$$\mathbf{b}^t \leftarrow \arg\min_{\mathbf{b}^t} \mathcal{J}(\mathbf{b}^t)$$

$$\text{s.t.} \quad f(x) \backsimeq f'(x), \quad \forall x \in \mathcal{R}^t_{\delta=0},$$

$$f' = \mathcal{Q}(f, \mathbf{b}), \quad b_{\min} \leq b_l \leq b_{\max}, \quad \forall l.$$

`// Equivalence Verification Problem`
Quantize f using $\mathbf{b}$ to obtain f'.
Compute $\mathcal{X}^t_{\delta=0} = \mathcal{G}(f, f', \mathcal{R}_\delta)$.
Update $\mathcal{R}^{t+1}_{\delta=0} \leftarrow \mathcal{R}^t_{\delta=0} \cup \mathcal{X}^t_{\delta=0}$;
$t \leftarrow t + 1$;
Until $\mathcal{X}^{t-1}_{\delta=0} = \emptyset$ ***or*** *termination criteria met.*
Algorithm 1: Counterexample Guided Quantization Refinement Algorithm

In the **Bit Search Problem**, the equivalence constraint is applied to the updated set of critical data points $\mathcal{R}^{o=t}_{\delta=0}$, which changes at each iteration based on the counterexamples found. The optimization focuses on ensuring equivalence at these specific points, allowing for efficient computation. In the **Equivalence Verification Problem**, the verification is performed over the constant set of critical regions $\mathcal{R}_\delta$. This distinction ensures that while the bit search adapts to new counterexamples, the verification consistently assesses equivalence over the broader input space of interest.

This iterative method assumes that: 1) the initial critical regions $\mathcal{R}_\delta$ are representative of the NN's essential behaviors; 2) the equivalence verification process can effectively

identify counterexamples when equivalence does not hold; 3) he iterative process converges to a solution where no counterexamples are found within $\mathcal{R}_\delta$.

The optimization's complexity arises from: 1) he discrete nature of quantization bit widths, leading to a combinatorial search space; 2) he equivalence verification over continuous regions $\mathcal{R}_\delta$, which is computationally intensive; 3) he iterative updates to $\mathcal{R}_{\delta=0}^{o=t}$, potentially increasing the problem size at each iteration.

Advanced optimization techniques or heuristics may be required to find acceptable solutions within reasonable computational times. While convergence to an optimal solution is desirable, it may not always be achievable due to: 1) the potential for an infinite loop if new counterexamples are continually found; the possibility that the minimum allowable bit widths $b_{\min}$ are reached without achieving equivalence. Termination criteria, such as a maximum number of iterations or acceptable tolerance levels, should be defined to ensure practical applicability.

By reformulating the Equivalent Quantization Problem as an iterative process involving the Bit Search and Equivalence Verification subproblems, we create a dynamic framework that adapts to the NN's behavior. This approach allows for a targeted search for feasible bit widths while ensuring functional equivalence over critical input regions, balancing quantization efficiency and network performance.

3.4 Iterative Quantization Framework Implementation

The proposed framework, termed CEG4N, implements a counterexample-guided optimization approach to solve the equivalent quantization problem. It addresses the two sub-problems defined previously (Section 3.3.2): optimization of bit widths and formal verification of neural network equivalence. CEG4N is composed of three modules, each fulfilling a distinct role within the iterative quantization framework, as illustrated in Figure 2.

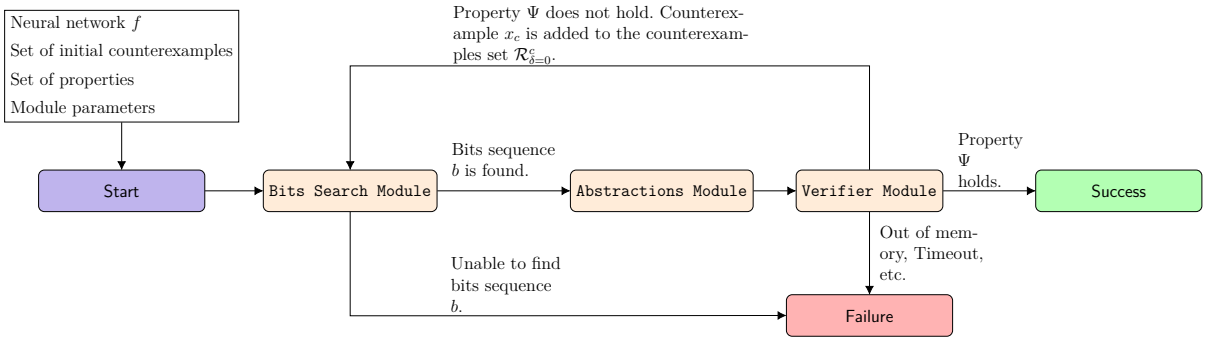

Figure 2 – An overview of CEG4N's architecture, highlighting the relationship between main modules and their inputs and outputs.

3.4.1 Bits Search Module (BSM)

The Bits Search Module (BSM) implements an optimization process using a genetic algorithm (GA). Genetic algorithms are heuristic optimization techniques inspired by biological evolution, employing mechanisms such as selection, crossover, and mutation to evolve populations of candidate solutions toward optimality. The BSM begins with an initial population of candidate bit-width configurations for each neural network (NN) layer, represented as vectors of integers constrained within user-defined bounds (2-32 bits).

Three distinct objective functions are employed within the BSM:

- **Single Objective:** Minimizes the overall bit-width uniformly across layers.
- **Multi-objective (Pareto-optimal):** Balances between minimizing bit widths and maintaining model accuracy, generating a Pareto front of candidate solutions.
- **Weighted Objective:** Incorporates neuron-count-based weighting, emphasizing quantization precision in layers with higher neuron density and thus greater computational impact.

The genetic algorithm configurations were empirically tuned (see Table 1), resulting in a recommended population size of 5 and approximately 100 generations per neural network layer. This choice balances computational feasibility with the likelihood of identifying near-optimal bit-width configurations. Explicit termination criteria based on generation count prevent indefinite computation.

Table 1 – Summary of experiments for tuning Genetic Algorithm Parameters.

Number of Layers	Generations	Population	Percentage of optimal solutions
7	800	5	100
7	750	5	100
7	700	5	98
7	50	5	0
2	250	5	100
2	200	5	100
2	150	5	96
2	50	5	30

In Table 1, we report a summary of experiments conducted to tune the parameters of the Genetic Algorithm, more precisely, the number of generations. For example, a NN with 2 layers would require a brute force algorithm to search for 52^2 combinations of bits widths for the quantization. Similarly, a NN with 7 layers would require a brute force algorithm to search for 52^7 combinations of bits widths. We conducted a set of experiments where we ran the GA one hundred times with a different number of generations options

ranging from 50 to 1000. In addition, we fixed the population size to 5. From our findings, the GA needs about 100 to 110 generations per layer to find the optimal bit width solution for each run.

3.4.2 Abstractions Module (AM)

The Abstractions Module (AM) translates the original and quantized neural networks into representations suitable for formal verification. Currently, two primary abstraction formats are supported:

- **ONNX (Open Neural Network Exchange):** Models trained in PyTorch (PASZKE et al., 2019) are exported to ONNX (BAI et al., 2019), enabling accurate reproduction of neural network behavior without loss in performance or accuracy.
- **C/C++ Abstraction:** Neural networks are abstracted explicitly into C-language models performing double-precision arithmetic. These abstractions precisely encode neural network computations, differing only in their precision of weights and biases between original (floating-point) and quantized (fixed-point) models.

The AM maintains two synchronized but distinct versions of the neural networks: an optimization-oriented Python implementation for use by the BSM, and a verification-oriented abstraction compatible with formal verification tools in the Verifier Module.

3.4.3 Verifier Module (VM)

The Verifier Module (VM) evaluates the equivalence of neural networks according to formally defined properties. Currently, the VM integrates two verification techniques:

- **Bounded Model Checking (BMC)** using ESBMC (GADELHA et al., 2018; MONTEIRO; GADELHA; CORDEIRO, 2022), a verification tool capable of generating explicit counterexamples when equivalence properties are violated.
- **Geometric Path Enumeration (GPE)** using NNEQUIV (TEUBER et al., 2021), focusing on enumerating activation patterns to verify neural network equivalence.

Counterexample management within the VM involves a two-step validation process to ensure robustness against spurious counterexamples. Initially identified counterexamples are subjected to a validation check directly against both the original and quantized networks. Only those inputs that definitively demonstrate functional divergence influence subsequent optimization iterations. Timeout values for verification were empirically determined, establishing a limit of 25 minutes per verification instance to balance computational practicality with verification depth.

Equivalence verification leverages two types of properties:

- **TOP equivalence:** Ensures identical class predictions for classification tasks.
- **EPSILON equivalence:** Ensures outputs differ by no more than a specified tolerance (e.g., $\epsilon = 0.05$), particularly suitable for regression tasks.

The equivalence properties are encoded into C language assertions compatible with ESBMC, using non-deterministic inputs to represent regions around critical data points, as described next. As an example, suppose a NN f, for which $x \in \mathcal{X}$ is a safe input and $y \in \mathcal{O}$ is the expected output of $f(x)$. We now show how one can specify the equivalence properties. For this example, consider that the function f can produce the outputs of f in floating-point arithmetic, while f' produces the outputs of f in fixed-point arithmetic (i.e., quantization). First, the concrete NN input x is replaced by a non-deterministic one, which is achieved using the command **nondet_float** from the ESBMC.

Listing 3.1 – Definition of concrete and symbolic input domain in *EBMC*.

```
float x0 = -1.0;
float x1 = 1.0;
float s0 = nondet_float();
float s1 = nondet_float();
```

Listing 3.2 – Definition of input constraints in *EBMC*.

```
const float EPS = 0.5;
__ESBMC_assume(x0 - EPS <= s0 && s0 <= x0 + EPS);
__ESBMC_assume(x1 - EPS <= s1 && s1 <= x1 + EPS);
```

Listing 3.3 – Definition of output constraints in *EBMC*.

```
__ESBMC_assert(f(s0, s1) == fq(s0, s1));
```

3.4.4 Clarification on Input Quantization vs. Input-layer Quantization

To avoid ambiguity, it is crucial to distinguish between input quantization and input-layer quantization explicitly:

- **Input Quantization:** Refers to directly quantizing the network inputs themselves, altering input feature values into lower precision representations. This process impacts data precision and potentially degrades feature quality.
- **Input-layer Quantization:** Refers specifically to quantizing weights, biases, and activations of the first computational layer after the inputs, ensuring stable arithmetic

operations without modifying the raw input data. CEG4N employs input-layer quantization to prevent arithmetic overflow and minimize representational inaccuracies, preserving the original input feature fidelity.

3.4.5 High-level overview of a CEG4N execution

An execution instance of CEG4N proceeds iteratively, structured as follows:

1. Initialization with the original neural network, initial counterexamples, BSM parameters, and specified equivalence properties.
2. Execution of the BSM to identify candidate bit-width configurations.
3. If no feasible configuration is found within the generation limit, CEG4N terminates unsuccessfully.
4. Otherwise, the AM translates neural networks into verification-ready abstractions.
5. The VM evaluates the abstractions against equivalence properties.
6. Upon successful verification, the quantized network solution is finalized.
7. If verification fails, new validated counterexamples are added to the optimization constraints, and the process returns to Step 2.
8. The iterative process continues until no new counterexamples are found or until a global timeout criterion is met.

3.4.6 Algorithmic choices and justifications

The employment of a genetic algorithm in the BSM was guided by the discrete and combinatorial nature of bit-width optimization. Heuristic optimization approaches, like genetic algorithms, effectively navigate complex optimization landscapes, offering practical solutions where exact optimization methods become computationally prohibitive. Empirical analyses (see Section 3.4.1) justified choosing a population size of 5 and a limit of approximately 100 generations per network layer as sufficient for reliably converging toward near-optimal solutions. The timeout and verification limits set in the VM balance thoroughness with computational resources, ensuring timely and actionable outcomes. Finally, rigorous two-step validation of counterexamples ensures that optimization iterations are robust, accurately guided by valid equivalence violations.

3.5 Summary

In this chapter, we discussed a framework for achieving EQ of NNs to ensure their functionality is preserved when reducing precision for deployment in resource-constrained environments and safety critical applications. The core challenges addressed include maintaining behavioral equivalence between the original and quantized NNs and optimizing quantization parameters to minimize resource usage. The key topics we covered in this chapter are as follows.

1. **Equivalent Quantization Problem Formalization:**
   - defines the task of quantizing an NN f into f' such that they produce equivalent outputs over a specified critical region $\mathcal{R}_\delta$;
   - behavioral equivalence is characterized by an equivalence relation ($\eqsim$) that can be exact ($\equiv$) or approximate ($\sim_{p,\epsilon}$);
   - optimization is formulated to minimize a cost function $\mathcal{J}(\mathbf{b})$ while satisfying constraints on equivalence and bit width ($b_{\min}, b_{\max}$).
2. **Iterative Quantization Framework:**
   - proposes an iterative method combining optimization (bit search) and formal equivalence verification;
   - counterexamples ($\mathcal{X}_\delta$) from failed equivalence checks refine the quantization process in subsequent iterations;
   - simplifies constraints by focusing on equivalence at critical data points ($\mathcal{R}_{\delta=0}$).
3. **Formal Equivalence Verification:**
   - techniques include Bounded Model Checking (BMC) and Geometric Path Enumeration (GPE);
   - introduces a novel approach of de-quantizing f' into f'' to leverage floating-point arithmetic for equivalence verification.
4. **Quantization Scheme:**
   - uses uniform quantization for weights and biases and asymmetric quantization for activations;
   - addresses trade-offs between precision reduction and functional accuracy through scaling parameters.
5. **Framework Implementation (CEG4N):**

- composed of three modules, i.e., **Bits Search Module (BSM)**, which uses genetic algorithms to optimize bit widths, **Abstractions Module (AM)**, which translates NNs into verification-ready formats (e.g., ONNX, C/C++), and **Verifier Module (VM)**, which ensures equivalence using tools like ESBMC and NNEQUIV.
- iteratively refines bit widths using counterexamples until equivalence is achieved or constraints are met.

In conclusion, we examined and discussed the challenges and computational complexity of EQ, and offered a systematic solution that integrates optimization and formal methods to preserve NN functionality in quantized settings.

4 Evaluation and Results

This chapter presents the evaluation of the CEG4N framework we proposed in Chapter 3. In the following sections, we state our evaluation goals, explain the benchmarks and metrics used in the evaluation and analysis, and explain how CEG4N was applied to different NN architectures and datasets. The results compare different configurations of CEG4N and also compare it with an existing quantization technique. Our evaluation aims to highlight the effectiveness of CEG4N in preserving the behavior integrity of NNs post-quantization. We also critically analyze the results in the context of the proposed evaluation goals.

4.1 Evaluation goals

In this section, we outline the primary objectives of our experimental evaluation of the CEG4N framework, designed to address the challenge of quantizing neural networks (NNs) to lower-precision versions without compromising their functional integrity. Our evaluation is structured around six specific goals: assessing the framework's effectiveness, analyzing performance metrics, evaluating equivalence-checking techniques, and examining the quality of quantized models. Additionally, we investigate the impact of maintaining equivalence constraints on accuracy and model size reduction. This comprehensive evaluation thoroughly explains CEG4N's capabilities and limitations in various quantization scenarios. Our experimental evaluation has the following list of goals:

EG1 - Assess the effectiveness of CEG4N
Demonstrate that CEG4N can generate quantized NNs that are verifiably equivalent to their original counterparts. This goal involves testing whether the integration of formal equivalence verification ensures the behavioral integrity of quantized NNs. It also verifies that performance metrics, such as accuracy and efficiency, remain consistent post-quantization.

EG2 - Analyze performance metrics
Identify the strengths and limitations of the CEG4N framework in solving the quantization problem. This goal aims to analyze the framework's performance, identifying key metrics that reflect its efficiency and reliability in different scenarios. It should help us pinpoint the areas where CEG4N excels and where it might need improvements, providing a comprehensive understanding of its capabilities and potential drawbacks in various quantization scenarios.

EG3 - Evaluate equivalence-checking techniques
Compare CEG4N's performance with different configurations, such as varying verification tools and constraints. Assess different FEV techniques and their impacts on CEG4N's performance. This goal seeks to understand how different settings and parameters affect the framework's performance, helping to identify the most effective configurations for various use cases.

EG4 - Perform quality assessment of quantized models
Evaluate the quality of the quantized models generated by CEG4N in terms of accuracy degradation and equivalence constraint satisfaction. This goal focuses on ensuring that the quantized models meet high standards of quality and performance, maintaining the functional integrity of the original models while undergoing quantization.

EG5 - Evaluate the impact of NNE on the QNNs accuracy
Investigate how maintaining equivalence constraints between the original and quantized NNs impacts the overall accuracy of the quantized models. This goal focuses on understanding whether equivalence guarantees can be achieved without significantly compromising performance metrics such as accuracy and how different levels of equivalence might influence model behavior.

EG6 - Evaluate the impact of NNE on the NN size reduction
Explore the balance between reducing the size of the NN models through quantization and maintaining the FE to the original models. This goal aims to determine the extent to which model size can be optimized without adversely affecting the network's integrity and performance, ensuring that quantized models remain efficient and reliable for practical applications.

4.2 Evaluation metrics

In this section, we explore the metrics used in our evaluation to assess the performance and effectiveness of the CEG4N framework. The metrics align with the established goals and provide a structured framework for our experiments and analysis. By relating the metrics to the evaluation goals, we can ensure that they provide meaningful insights into the performance, effectiveness, and reliability of the CEG4N framework. This structured approach helps systematically evaluate the framework and set the basis for future improvements and research. The complete list of metrics is:

Timeout Failures: The number of times the CEG4N framework exceeded the predefined time limit for the verification step during the experiments. This metric is directly related to assessing the efficiency of the CEG4N framework (EG3). High timeout failures indicate areas where the framework may need optimization or improvement in handling complex verification tasks. This metric helps identify bottlenecks in the verification process, ensuring that the framework can operate within practical time limits in real-world applications.

Verification Failures: The number of failures observed in the Verifier Module during the experimental evaluation indicates the robustness of the verification module. This metric is linked to validating the integration of formal equivalence verification (EG1, EG4). A lower rate of verification failures suggests a more robust verification module supporting the integrity of the quantized NNs. Ensuring the reliability of the verification process is critical for maintaining the safety and functionality of the quantized models in practical deployments.

Quantization Failures: Instances of failure in the Search Module during the experimental evaluation, assessing the effectiveness of the optimization method in tackling the quantization issue. This metric evaluates the effectiveness of the optimization technique used in CEG4N (EG2, EG3). Quantization failures indicate challenges in the optimization process, highlighting areas for improvement. Understanding these limitations is important for improving the framework's handling of diverse NN architectures.

Successful Quantizations: Instances of successful termination of the quantization framework indicate the framework's ability to adaptively quantize different NN architectures. This metric measures the overall success of the quantization framework (EG2, EG3). A high success rate demonstrates the framework's capability to adaptively quantize NNs without significant loss of functionality, validating its practical applicability in creating functional quantized models for various scenarios.

Iterations Count: The number of iterations required to complete the quantization process. This metric indicates the efficiency of the quantization process (EG2, EG3). Fewer iterations suggest a more efficient optimization process, while a higher count might indicate potential inefficiencies. Optimizing this metric helps in reducing computation time and resources, making the framework more suitable for real-time or resource-constrained applications.

Counterexamples Count: The count of counterexamples produced during the iterations is used to refine the quantization. This metric reflects the iterative refinement process during quantization (EG3, EG5). A higher count of counterexamples indicates a more rigorous refinement process, potentially leading to more accurate quantized models. Ensuring the robustness of the quantization process involves continuously improving the model based on identified weaknesses.

Original Accuracy: The accuracy of the original NN models on their respective datasets. This metric serves as a baseline to measure the impact of quantization on model performance (EG3, EG5). Comparing original accuracy with post-quantization accuracy helps in assessing any performance degradation. This is essential for evaluating the trade-offs between model accuracy and the benefits of quantization, such as reduced model size and increased efficiency.

Accuracy Drop: The loss in accuracy before and after quantization, validating the framework's effectiveness in maintaining the integrity of NN functionalities. This metric measures the impact of quantization on model performance (EG3, EG6). Minimal accuracy drop indicates that the framework preserves the functional behavior of the original model. It is crucial to ensure that quantized models remain effective in their application domains, especially in safety-critical tasks where accuracy is paramount.

Equivalence Count: The number of quantized networks that satisfy the specified equivalence constraints. This metric assesses whether the quantized models meet the specified equivalence constraints (EG1, EG4, EG5). Ensuring equivalence is crucial for maintaining the integrity and expected behavior of the models. Validating the reliability of the quantized models ensures that they perform consistently with the original models, which is important for user trust and adoption in real-world applications.

The rationale behind our choice for these metrics considers some principles. Practical aspects such as the following: (1) they allow us to analyze and interpret the collected data, ensuring that we can understand the differences in the framework we propose and existing methods; (2) they allow us to consider practical implications of our findings; and (3) they allow us to extract insights that can iteratively refine our framework, emphasizing areas such as enhancing verification processes, reducing quantization failures, and maintaining accuracy post-quantization.

Ultimately, the chosen metrics should not only validate the effectiveness of the proposed framework and techniques but also offer insights into potential areas for further research and development. This ensures the continual advancement of methodologies for quantifying NNs that address both current and future challenges in the field.

4.3 Evaluation benchmarks

Our experimental setup was designed to provide an evaluation that utilized a diverse set of datasets and models to ensure appropriate testing of the CEG4N framework across various scenarios. These benchmarks cover a wide range of network architectures and applications.

4.3.1 Datasets

We selected a variety of datasets to comprehensively evaluate the performance of the CEG4N framework. Each dataset was chosen for its relevance to different application domains and its ability to test different aspects of NN performance post-quantization.

4.3.1.1 ACAS Xu

Acas Xu, which stands for Airborne Collision Avoidance System Xu, is an advanced version of collision avoidance systems used for manned aircraft tailored for the unique dynamics of UAVs (JULIAN et al., 2016). It is integral to the domain of autonomous systems and air traffic control. The dataset consists of inputs and outputs derived from simulations of various encounter scenarios between UAVs. These scenarios include a wide range of potential conflicts, varying in factors such as initial positions, velocities, headings, and altitudes of the UAVs involved. Each entry encapsulates a specific state of the UAVs, represented by a set of numerical features. The outputs are the recommended advisories or actions that the UAV should take to avoid a collision, determined by the NN model trained on the dataset.

The primary purpose of the Acas Xu dataset is to provide a robust foundation for evaluating the performance of NNs in making real-time, safety-critical decisions. By using this dataset, researchers and engineers can assess the decision-making capabilities of these systems under a variety of simulated conditions that reflect real-world complexities and uncertainties. Furthermore, the dataset serves as a benchmark for comparing different NN architectures and training methodologies, facilitating advancements in the development of more reliable and efficient collision avoidance systems. This dataset was chosen to evaluate the impact of quantization on NNs used in safety-critical applications.

4.3.1.2 MNIST

The MNIST (Modified National Institute of Standards and Technology) dataset is a widely recognized and extensively utilized resource within the field of ML and pattern recognition (LECUN; CORTES, 2005). This dataset comprises a collection of 70,000 grayscale images of handwritten digits, each size-normalized and centered within a fixed-size 28x28 pixel frame. The images are divided into two subsets: a training set containing 60,000 images and a test set containing 10,000 images. Each image is associated with a corresponding label that denotes the digit it represents, ranging from 0 to 9. This labeling facilitates supervised learning, wherein models are trained to learn the mapping between images and their respective labels.

One of the key features of the MNIST dataset is its pre-processing, which involves the normalization and centering of the digits. This step ensures that the digits are presented consistently, minimizing variations that could potentially affect the performance of ML

models. Despite its simplicity, the MNIST dataset poses several challenges, such as the need for models to generalize well from the training set to the test set, handle variations in handwriting styles, and maintain accuracy despite the limited resolution of the images.

4.3.1.3 Seeds

The seeds dataset is used in ML and statistical analysis for classification tasks. This dataset contains measurements of geometrical properties of kernels belonging to three different varieties of wheat: Kama, Rosa, and Canadian (CHARYTANOWICZ et al., 2010). Each entry in the dataset represents a single seed, characterized by seven numerical attributes that describe various morphological features, such as area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, and length of the kernel groove.

The dataset is structured to facilitate analysis. Each record contains values for each of the seven attributes followed by a class label indicating the variety of wheat. The balanced composition of the dataset, containing an equal number of samples from each wheat variety, provides a robust foundation for training and testing classification models.

4.3.1.4 Iris

The Iris dataset is widely recognized in ML and statistics, originally introduced by the British biologist and statistician Ronald A. Fisher in his 1936 paper, "The Use of Multiple Measurements in Taxonomic Problems" (FISHER, 1936). It consists of 150 observations, each representing an individual iris flower. These observations are evenly distributed across three species of iris: Iris setosa, Iris versicolor, and Iris virginica, with each species represented by 50 observations. The dataset is characterized by four primary features, which are the measurements of the following attributes of the iris flowers: sepal length, sepal width, petal length, and petal width.

The relatively small size of the dataset allows for rapid computation and straightforward visualization, aiding in the comprehension of complex concepts in data science and ML.

4.3.1.5 CIFAR-10

The CIFAR-10 dataset is a widely used benchmark in ML and computer vision, created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). It comprises $60,000$ color images, each with dimensions of 32x32 pixels, categorized into ten distinct classes. The dataset is divided into a training set of 50,000 images and a test set of 10,000 images. Each image is represented in RGB format and contains 3 channels corresponding to the components of red, green, and blue.

The CIFAR-10 dataset serves as a standard benchmark for evaluating the performance of different ML algorithms, particularly CNNs. Despite its simplicity compared to more extensive datasets, CIFAR-10 presents a non-trivial challenge due to the variability in object poses, backgrounds, and lighting conditions within each class.

4.3.2 Description of NNs models

To evaluate the CEG4N framework, we tested it on various NNs trained on the aforementioned datasets. These NNs were chosen to cover multiple architectures and complexities, ensuring a comprehensive assessment of the framework's performance.

For the **ACAS Xu** dataset, we evaluated CEG4N on nine pre-trained NNs (BAK; LIU; JOHNSON, 2021), each containing six layers and 300 ReLU nodes. These networks were obtained from the VNN benchmarks.

For the **MNIST** dataset, we evaluated CEG4N on nine NNs. Three models contain a single layer with 10, 25, and 50 ReLU nodes, following the architecture described by Eleftheriadis et al. (ELEFTHERIADIS et al., 2022). Three other models, obtained from the VNN's benchmarks, have 2, 4, and 6 layers, each with 256 ReLU nodes. The remaining three models were trained using resized MNIST images, similar to the first three single-layer ones. Additionally, we employed 8x8 resized images to reduce dimensionality and provide invariance to small image distortions.

For the **Seeds** dataset, we evaluated CEG4N on four NNs containing a single layer with 4, 6, 10, and 15 ReLU nodes. These networks were specifically trained for evaluating CEG4N.

For the **Iris** dataset, we evaluated CEG4N on three NNs containing two layers with 4, 10, and 15 ReLU nodes in each. These networks were trained specifically for evaluating CEG4N.

For the **CIFAR-10** dataset, we evaluated CEG4N on two pre-trained NNs from VNN. One network has 3 convolutional and two linear layers, each with 250 neurons, while the other has two convolutional and two linear layers, each with 250 neurons. Both networks use only ReLU activations.

This diverse set of NNs, tested across multiple datasets, allows for a thorough evaluation of the CEG4N framework's effectiveness in different scenarios, ensuring robust and reliable results.

4.4 Evaluation of CEG4N using different benchmarks and equivalence properties

Here, we discuss some experiments we planned to evaluate CEG4N's performance on different benchmarks, configurations, and equivalence properties. These experiments aim to address our evaluation goals EG1, EG2, and EG3; see Section 4.1 for more details. With this evaluation, we want to understand if it can generate QNNs with guarantees of preserved FE. The experiments also allow us to better understand CEG4N's strengths and limitations. We also want to evaluate the configuration options we have for CEG4N. Moreover, the proposed methodology presents inherent flexibility, allowing verifiers, constraints, and other parameters to be easily changed and promptly evaluated. Below, we provide an overview of the steps required to configure and run the experiments using CEG4N.

1. Model selection: Choose a model architecture from the selected benchmark dataset.
2. Select and configure an optimizer: Select NSGA-II as the optimizer to be used, define values for the bit width options, define the number of generations and population size, and specify the initial set of counterexamples.
3. Select and configure a verifier: Select an verifier, specify the equivalence type, specify the input space bounds, and define a timeout value.
4. Execution: Run CEG4N with the specified configurations.
5. Data collection: Record the relevant metrics.

In the remaining of this section, we discuss each step in the overview above and dive into the details of the parameter selection and configuration options we highlighted in it.

4.4.1 Benchmarks selection

In the SC1 we ran experiments using the NNs of the following benchmarks: Acas Xu, Iris, Seeds, MNIST, and CIFAR. The complete description of the benchmarks and the NNs is given in Section 4.3.

4.4.2 Optimizer configuration

As explained in Section Section 3.3.2, we use a search-based optimization algorithm (NSGA-II) in the BSM implementation to find optimal bit widths for NN quantization. The choice of NSGA-II is motivated by its multi-objective nature, allowing it to effectively

balance trade-offs between different objectives and constraints. Below, we outline the key configuration details and rationales for our the BSM setup.

In conclusion, the configuration of the BSM was designed to accommodate the precision requirements of different NN frameworks and programming languages while ensuring efficient and accurate quantization. The NSGA-II algorithm's multi-objective approach allows for effective balancing of various constraints and objectives, making it suitable for the quantization of our diverse set of NN benchmarks. NSGA-II was chosen over other optimization algorithms due to its ability to handle multiple objectives simultaneously, such as minimizing bit width while functional integrity. Previous studies have demonstrated NSGA-II's superiority in dealing with complex, multi-dimensional optimization problems, making it an ideal choice for our application(WANG et al., 2021b; REZK et al., 2022).

4.4.3 Verifiers

We run our experiments with two formal equivalence verifiers: ESBMC and NNEQUIV.

4.4.4 Lower and upper bounds configuration

First we need to inform BSM of the bounds for the search space of the optimization. In this case we set to lower and upper bounds to 2 and 32, respectively. This implies that the quantization schemes found by the BSM only contain combinations of bit width values between 2 and 32. A lower bound of 2 was chosen because it is the first valid integer that does not break our quantization scheme, ensuring that minimal bit widths still provide meaningful quantization without losing critical information. The upper bound of 32 matches the maximum number of bits typically used for integer representation in many NN frameworks, such as PyTorch and ONNX, and in programming languages like C.

Additionally, while the upper bound could be higher depending on the precision of NN weights, the standard choice for training and storing weights is single-precision floating-point format, which is commonly supported. Using 32 bits aligns with this standard and ensures compatibility across various platforms and frameworks. Studies in literature have shown that bit widths beyond 32 do not significantly improve performance for most practical applications, thus justifying our upper limit. Setting a greater lower bound or a lower upper bound is possible. For example, if the lower and upper bounds were set to 8 and 16, the practical implication would be that the quantization schemes found by the BSM would only contain bit widths values between 8 and 16.

4.4.5 Generations and population size configuration

The number of generations and the population size are parameters we need to specify for the BSM. We configured the BSM to run for 1000 generations with a population of 50 individuals. These numbers were determined empirically by incrementally testing to ensure the GA could find a solution for all benchmarks. Specifically, we conducted multiple trials where we provided an empty $\mathcal{R}_{\delta=0}^{t=0}$ to BSM and varied the number of generations: if the outputted b^t matched a vector such that $b = 2 \forall n \in b^t$, then GA could find a solution within the given number of generations. We found that 1000 generations and a population of 50 individuals consistently produced solutions across various NN architectures and sizes. This empirical approach ensured that the chosen numbers were sufficient for our experiments. For different use cases, depending on the size of the NNs, it may be necessary to fine-tune these parameters to avoid premature convergence to sub-optimal solutions genetic, to ensure diversity, and to thoroughly explore the solution space.

4.4.6 Initial set of counterexamples configuration

The initial setup of CEG4N involves selecting a representative sample for each class from the datasets to establish a baseline for the initial set $\mathcal{R}_{\delta=0}^{t=0}$ and the equivalence properties. This was done by randomly selecting one representative sample for each class from the datasets. For example, we selected ten samples for MNIST and 3 for Iris. The samples do not necessarily have to be from the benchmark dataset and can include synthetic data. In our experiments, we used real data from the chosen datasets. This choice is justified by three factors: (1) the practical aspect of using samples from the benchmarks ensures relevance to real-world data, (2) the representativeness of the test set maintains diversity in the initial population, and (3) the added variability in our experiments enhances the robustness of the evaluation. Random selection mitigates biases that could arise from systematic sampling, ensuring a fair evaluation of the framework's performance.

The definition of the initial set of counterexamples is straightforward. We simply add each sample to the respective set. For example, we selected 10 samples for MNIST, then these ten samples were used as the counterexamples for the MNIST experiments. Similarly, we selected 3 samples from the Iris dataset, then these 3 samples are used as the counterexamples for the Iris experiments. The set of counterexamples $\mathcal{R}_{\delta=0}^{t=0}$ is an important aspect of CEG4N optimization strategy, and its definition has practical implications and a direct effect on the CEG4N's ability to quantize the NNs. If we recall our discussion of CEG4N in Section 3.4, the counterexamples constrain and, therefore, guide the optimization done in the *BSM* in the right direction. Not defining an initial or non-representative set can result in inefficient optimization in *BSM*.

4.4.7 Equivalence properties configuration

To perform the equivalence verification we need to define: the original NN f, which is automatically done by a NN from the benchmarks; the QNN, whic is automatically done by CEG4N at runtime; the definition of the equivalence type, which is the CBE in our experiments with classifiers; and the definition of input space bounds, which also defined based on input samples. The main difference is that, instead of using concrete inputs from this sample set, we have to calculate a lower and upper bound for each input sample based on a specified input bound value δ.

The specification of the equivalence properties was defined by: (1) selecting one real input sample for each class (similar to the definition of $\mathcal{R}^{t=0}_{\delta=0}$), at random; (2) choosing CBE, and (3) selecting an input bound value δ. Regarding the latter, we have defined three possible values for δ (see Eq. Equation 2.19): $\delta = [0.01, 0.03, 0.05]$. Such values are defined based on input bounds reported by other works on NNE ((PAULSEN et al., 2020; TEUBER et al., 2021; ELEFTHERIADIS et al., 2022)) and were selected empirically, based on input data and experiments, and reflect the full structure here: benchmarks with narrow input range and CBE, which usually leads to tighter input regions. In addition, studies in the literature usually adopt only one, while our work provides a margin for its discussion. As an example, MNIST, which has 10 output classes, we could define a total of 3 sets with 10 input constraints each.

For Acas Xu, we followed the same strategy used by Teuber et al. (TEUBER et al., 2021), i.e., NBE as equivalence form, while the choices for the input space bounds of the equivalence properties are $\delta = [0.1, 0.3, 0.5]$. Again, three different values were adopted for δ, also empirically, but now taking into account Acas Xu's aspects: broader input range and NBE. The use of norms in the verification of properties of Acas Xu benchmarks (and other similar benchmarks (ELEFTHERIADIS et al., 2022)) is common in the literature (KATZ et al., 2017; PAULSEN et al., 2020; TEUBER et al., 2021).

If we revisit the definition of approximate equivalence Section 2.4, we must choose two additional parameters, namely p and ϵ. The value for ϵ is usually chosen according to the application domain of the NNs being verified so that it is possible to prove equivalence and, simultaneously, the resulting NNs are useful, i.e., they present tolerable output differences. It is also possible to find an optimal ϵ by incrementally looking at counterexamples and deciding if, from the user perspective, their outputs are equivalent(BAK; LIU; JOHNSON, 2021). Ultimately, we decided for $\epsilon = 0.05$ as it means a maximum difference of 10% in Acas Xu's scores. In addition, such a value was also adopted by Teuber et al. (TEUBER et al., 2021). Finally, we chose $p = \infty$ due to efficiency reasons (KATZ et al., 2017) and also aimed at consistency across different verifiers, which was also adopted by Teuber et al. (TEUBER et al., 2021).

4.4.8 Time limit configuration

A timeout is important to ensure termination and should ideally not be arbitrary. In the case of this experimental evaluation, each verification of the equivalence property has a time limit to complete, and it should take at most 20 minutes. This time is consistent across all verifiers and experiments. It was based on hardware configuration, expected run-time, and other aspects, but different limits can be set to suit distinct scenarios. There is no ideal value since it depends on computational resources and the specific use case and applications. If one can allow longer timeout values, it is advised to opt for longer timeouts, although it impacts the overall execution time of CEG4N. Our choice for 20 minutes was made because it suited our experimental needs; however, we were aware of this choice likely leading to more timeout observations.

4.4.9 Collected metrics

The metrics recorded during the experiments included: 1) the number of iterations; 2) the number of counterexamples; 3) the quantization scheme or the bits for each layer in QNN; and 4) the outcome of each experiment. The Section 4.2 gives a full discussion of these metrics.

4.5 Evaluation of the quality of the QNNs generated by CEG4N

Here we discuss some experiments we planned to evaluate the quality of QNNs generated by CEG4N. These experiments aim to address our evaluation goals EG4, EG5, and EG6; see Section 4.1 for more details. In summary, we want to understand the impact of preserving equivalence property in the QNNs. Additionally, we want to compare the QNNs generated by CEG4N with QNNs generated by other quantization tools. Below is a structured overview of the steps we performed to conduct the quality evaluation experiments.

1. Benchmark selection: choose a NN from the selected benchmark dataset.
2. Quantization by CEG4N: quantize the neural networks using CEG4N and record the initial counterexamples used.
3. Input sample set selection: define a set of samples for quantization purposes.
4. Quantization by GPFQ: quantize the same neural networks using GPFQ with the specified quantization scheme (Bits) and initial counterexamples.
5. Test set select: define a set of samples for validation purposes.

6. Accuracy measurement: measure and record the accuracy of both the original and quantized neural networks, calculating the accuracy drop.

4.5.1 Benchmark selection

The selection of the benchmarks is based on whether or not the benchmark is a classification. For this reason, only the following benchmarks satisfy this criteria: Iris, Seeds, MNIST, and CIFAR.

Two quantization tools were evaluated: CEG4N and GPFQ.

4.5.2 Input sample set configuration

The GPFQ algorithm allows us to specify a set of representative input samples which it uses to improve the quality of the QNNs it generates. For simplification purposes, we use the same input samples contained in the set of counterexamples that needs to be specified for CEG4N.

4.5.3 Test set configuration

To evaluate the quality of QNNs generated by the quantization tools we used in this experimentation, we need to validate the QNNs against some real and representative input samples.

4.5.4 Collected metrics

Metrics recorded included the accuracy of the original neural network, the accuracy of the quantized neural networks, and the accuracy drop. For simplification purposes, we use the test dataset of each benchmark.

4.6 Data and Tools Availability

Our experiments are based on publicly available benchmarks. All tools, benchmarks, and results employed here are available on the supplementary web pages <https://zenodo.org/record/7126601> and <https://codeocean.com/capsule/1811188/tree>. We invite the scientific community and readers in general to download, set up, and experiment with the tool.

4.7 Presentation of Results

We start the presentation of results for our first set of experiments as described in Section 4.4. To recapitulate, we want to achieve the experimental goals EG1, EG2, and EG3, by showing that CEG4N can successfully generate QNNs verifiably equivalent to their original counterparts. In practical terms, we want to: 1) perform an empirical scalability study to help us evaluate the computational demands for quantizing and verifying the equivalence of NN models and (2) evaluate different equivalence-checking techniques and their impacts on the performance of CEG4N. Our findings are presented as follows: Table 2 contains the results for the Iris benchmarks; Table 3 contains the results for the Seeds benchmarks, the results for the MNIST benchmarks are presented in Table 4, Table 5, and Table 6; Table 7 contains the results for the CIFAR benchmarks; and the results for the Acas Xu benchmarks are in Table 8 and Table 9.

The tables contain the metrics we collected during the experiments and information that identifies the configuration with which CEG4N is used in the experiment for the respective metrics. Specifically, the column names and their definitions are: Model - tells the specific benchmark, Verifier - informs if ESBMC or NNQUIV was used in the experiment, δ - discloses the parameter δ (see Eq. (2.19)) which specify the input space bounds of the equivalence properties; No. Iter. - shows the number of iterations until CEG4N terminated, No. CEs - informs the number of counterexamples found by CEG4N, Bits - informs the last quantization scheme passed to VM containing the bit width for each layer of the NN; and Status - tells the CEG4N's termination status.

Table 2 – Summary of the CEG4N's results for the Iris benchmark.

Model	Verifier	δ	No. Iter.	No. CEs.	Bits	Status
iris_4x2	ESBMC	0.01	1	0	3,3,3	SQ
		0.03	1	0	3,3,3	SQ
		0.05	11	10	12,10,10	SQ
	NNEQUIV	0.01	1	0	3,3,3	SQ
		0.03	1	0	3,3,3	VF
		0.05	3	2	4,2,3	SQ
iris_10x2	ESBMC	0.01	2	1	3,3,2	SQ
		0.03	3	2	4,2,4	SQ
		0.05	7	6	8,7,9	SQ
	NNEQUIV	0.01	2	1	2,3,4	SQ
		0.03	3	2	2,3,4	SQ
		0.05	2	0	3,2,4	SQ
iris_15x2	ESBMC	0.01	1	0	2,2,3	SQ
		0.03	1	0	2,2,3	SQ
		0.05	8	7		QF
	NNEQUIV	0.01	1	0	2,2,3	SQ
		0.03	1	0	2,2,3	SQ
		0.05	1	0	2,2,3	SQ

Table 3 – Summary of the CEG4N's results for the Seeds benchmark.

Model	Verifier	δ	No. Iter.	No. CEs.	Bits	Status
seeds_4x1	ESBMC	0.01	1	0	3,3	SQ
		0.03	1	0	3,3	SQ
		0.05	9	8	12,13	SQ
	NNEQUIV	0.01	1	0	3,3	SQ
		0.03	1	0	3,3	SQ
		0.05	2	1	4,4	SQ
seeds_6x1	ESBMC	0.01	1	0	4,2	SQ
		0.03	4	4	12,12	SQ
		0.05	8	9	12,12	SQ
	NNEQUIV	0.01	1	0	4,2	SQ
		0.03	2	2	4,3	SQ
		0.05	2	1	4,3	SQ
seeds_10x1	ESBMC	0.01	2	1	3,4	SQ
		0.03	13	14	18,12	SQ
		0.05	3	4	6,4	TO
	NNEQUIV	0.01	2	1	3,4	SQ
		0.03	2	1	4,3	SQ
		0.05	2	1	4,4	SQ
seeds_15x1	ESBMC	0.01	4	3	5,2	SQ
		0.03	5	5	8,6	TO
		0.05	7	8	8,7	TO
	NNEQUIV	0.01	2	1	5,2	SQ
		0.03	2	1	4,4	SQ
		0.05	3	1	5,3	SQ

The termination status can classified into four possible outcomes: 1) successful Quantization (SQ), meaning that CEG4N ran for one or more iterations and was able to produce a QNN; 2) timeout (TO), which means CEG4N was unable to verify the equivalence property within a given time limit previously set; 3) quantization failure (QF), which means that CEG4N was unable to find a suitable qantization scheme to quantize a given NN; and 4) verification failure (VF), which means that some error occurred during the equivalence verification step, e.g., exceptions thrown by the VM have occurred.

In summary, CEG4N running with ESBMC (CESBMC) successfully generated QNNs for 17 out of 81 runs, considering all datasets, which account for 20.99% of all processes. In addition, CEG4N running with NNQUIV (CNNQUIV) was successful in 33 out of 81 runs, representing 40.74% of all processes. Most of the CEG4N's failures, with ESBMC, was due to timeouts, with 55 occurrences, representing 67.90% of the total. In contrast, CEG4N with NNQUIV resulted in 30 timeouts, i.e., 37% of the total.

Such a difference in timeouts can be attributed to many factors. For example, ESBMC, as an approach based on bounded model checking (BMC), is known to suffer

Table 4 – Summary of the CEG4N's results for the MNIST benchmark.

Model	Verifier	δ	No. Iter.	No. CEs.	Bits	Status
mnist-64-10x1	ESBMC	0.01	1	0	4,5	TO
		0.03	1	0	4,5	TO
		0.05	1	0	4,5	TO
	NNEQUIV	0.01	2	1	4,5	SQ
		0.03	2	1	4,5	SQ
		0.05	3	6		QF
mnist-64-25x1	ESBMC	0.01	1	0	3,3	TO
		0.03	1	0	3,3	TO
		0.05	1	0	3,3	TO
	NNEQUIV	0.01	2	3	5,6	SQ
		0.03	6	10	6,5	SQ
		0.05	4	11		QF
mnist-64-50x1	ESBMC	0.01	1	0	2,3	TO
		0.03	1	0	2,3	TO
		0.05	1	0	2,3	TO
	NNEQUIV	0.01	2	1	2,6	SQ
		0.03	3	5	5,8	TO
		0.05	1	0	2,3	TO

from scalability issues, which greatly diminishes its ability to support larger NNs (SENA et al., 2021) and can be seen in more detail in some tables. In Table 2 and Table 3, which show information regarding CESBMC runs for the Iris and Seeds datasets (at most two layers with less than 20 node each), respectively, only three timeouts were noticed. However, if we take a look at Table 4, Table 5, Table 6, Table 7, and Table 8, which show information regarding CESBMC runs for MNIST, CIFAR-10, and Acas Xu benchmarks, respectively, which are composed mostly by medium to large NNs (at most eight layers), the number of timeouts presents a significant increase. Indeed, no successful execution could even be identified. Moreover, the rare runs with a timeout, with the dataset Seeds processed by CESBMC, also happened with (1) NNs containing more than 10 neurons per layer, (2) more than 4 iterations, and (3) larger constraint regions ($\delta = 0.03$ and 0.05), which reinforces the explanation based on NN complexity and BMC scalability. We can check such cases in detail in Table Table 3. In summary, combining factors (1) and (3) mostly contributes to timeouts. The more neurons an NN has, the more operations it has to perform (and CEG4N). Besides, the bigger the δ value, the bigger the search space a verifier has to cover.

Although runs with CNNQUIV suffered from fewer timeouts, it's interesting that ESBMC required more iterations overall than CNNQUIV. Considering only the successful runs, CESBMC needed, on average, 4 iterations to produce a QNN, while CNNQUIV used 2. The explanation for this behavior is that ESBMC can find counterexamples that

Table 5 – Summary of the CEG4N's results for the MNIST benchmark.

Model	Verifier	δ	No. Iter.	No. CEs. Bits	Status	
mnist-784-10x1	ESBMC	0.01	1	0	3,4	TO
		0.03	1	0	3,4	TO
		0.05	1	0	3,4	TO
	NNEQUIV	0.01	1	0	3,4	SQ
		0.03	2	2	3,6	SQ
		0.05	7	9	17,21	SQ
mnist-784-25x1	ESBMC	0.01	1	0	3,5	TO
		0.03	1	0	3,5	TO
		0.05	1	0	3,5	TO
	NNEQUIV	0.01	1	0	3,5	SQ
		0.03	1	0	3,5	TO
		0.05	1	0	3,5	TO
mnist-784-50x1	ESBMC	0.01	1	0	3,2	TO
		0.03	1	0	3,2	TO
		0.05	1	0	3,2	TO
	NNEQUIV	0.01	1	0	3,2	SQ
		0.03	1	0	3,2	TO
		0.05	1	0	3,2	TO

NNQUIV is not. Since ESBMC and NNQUIV use different verification approaches, i.e., SMT and reachability analysis, respectively, the obtained results are expected to eventually diverge for some verification instances. In addition, NNQUIV produces many spurious counterexamples, which is not beneficial to our scheme. Indeed, CEG4N already expects that the verifiers can eventually produce spurious counterexamples, which are ruled out as explained in Section Section 3.4.3.

Quantization failures are another possible type of error, which can occur only in QM. One may notice that CESBMC presented only 9 cases, i.e., 11.11%, while 12 were identified for CNNQUIV. i.e., 14.81%. A possible explanation is that the search for the bits sequence is highly non-linear and highly dependent on the set of counterexamples $\mathcal{R}_{\delta=0}$ QM receives at a given iteration, which makes this optimization problem a hard one to solve. We should also consider that new constraints are added to the search problem after each iteration, which makes it even harder to solve. This is corroborated by the fact that most quantization failures occurred after 2 or more runs.

Our experiments show many quantization-failure events for Acas Xu's benchmarks. One explanation for that is that some of those NNs are highly sensitive to errors introduced by quantization processes, which affects their behaviors. Thus, in such a context, NNs may easily violate the constraint $f(x) \backsimeq f^q(x), \forall x \in \mathcal{R}^o$ during the step performed by BSM. In addition, for other NNs, a higher value for ϵ can also help increase the chance of a successful quantization. However, in the specific scenario used here, we used an ϵ value

Table 6 – Summary of the CEG4N's results for the MNIST benchmark.

Model	Verifier	δ	No. Iter.	No. CEs.	Bits	Status
mnist-256x2	ESBMC	0.01	1	0	4,3,5	TO
		0.03	1	0	4,3,5	TO
		0.05	1	0	4,3,5	TO
	NNEQUIV	0.01	1	0	4,3,5	TO
		0.03	1	0	4,3,5	TO
		0.05	1	0	4,3,7	TO
mnist-256x4	ESBMC	0.01	1	0	3,2,3,3,4	TO
		0.03	1	0	3,2,3,3,4	TO
		0.05	1	0	3,2,3,3,4	TO
	NNEQUIV	0.01	1	0	3,2,3,3,4	TO
		0.03	1	0	3,2,3,3,4	TO
		0.05	1	0	3,2,3,3,4	TO
mnist-256x6	ESBMC	0.01	1	0	3,2,3,3,4,3,4	TO
		0.03	1	0	3,2,3,3,4,3,4	TO
		0.05	1	0	3,2,3,3,4,3,4	TO
	NNEQUIV	0.01	1	0	3,2,3,3,4,3,4	TO
		0.03	1	0	3,2,3,3,4,3,4	TO
		0.05	1	0	3,2,3,3,4,3,4	TO

Table 7 – Summary of the CEG4N's results for the CIFAR benchmark.

Model	Verifier	δ	No. Iter.	No. CEs.	Bits	Status
cifar10_2_255	ESBMC	0.01	1	0	9,5,6,6,6	TO
		0.03	1	0	7,8,6,6,6	TO
		0.05	1	0	10,5,6,6,6	TO
	NNEQUIV	0.01	1	0	7,8,6,6,6	VF
		0.03	1	0		QF
		0.05	1	0	9,5,6,7,7	VF
cifar10_8_255	ESBMC	0.01	1	0	4,8,6,7	TO
		0.03	1	0	6,8,6,8	TO
		0.05	1	0	4,8,6,7	TO
	NNEQUIV	0.01	1	0	4,8,6,7	VF
		0.03	1	0	4,8,6,7	VF
		0.05	1	0		QF

based on other works, which should be adequate for our benchmarks (TEUBER et al., 2021).

Indeed, the conditions presented in the last paragraph probably prevented CEG4N from producing QNN candidates for VM. Moreover, successful runs mostly failed with timeouts. The main factors behind the latter are (1) the size of the NN in the number of neurons and (2) the number of features in the input space. Indeed, Acas Xu's NNs are mostly affected by the NN size problem, as they present 300 neurons in total. MNIST, in

Table 8 – Summary of the CEG4N's results for the ACAS Xu benchmark.

Model	Verifier	δ	No. Iter.	Bits	Status
ACASXU_1_1	ESBMC	0.1	1		QF
		0.3	1		QF
		0.5	1		QF
ACASXU_1_2	ESBMC	0.1	1	12,9,8,9,7,9,5	TO
		0.3	1		QF
		0.5	1	11,7,7,8,9,9,7	TO
ACASXU_1_3	ESBMC	0.1	1		QF
		0.3	1		QF
		0.5	1		QF
ACASXU_1_4	ESBMC	0.1	1	5,6,5,6,6,8,4	TO
		0.3	1	6,6,7,6,6,7,4	TO
		0.5	1		QF
ACASXU_1_5	ESBMC	0.1	1	5,6,6,6,6,7,4	TO
		0.3	1	5,6,6,6,6,7,4	TO
		0.5	1	8,6,7,5,5,7,4	TO
ACASXU_1_6	ESBMC	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	5,9,4,2,6,4,4	TO
		0.5	1	2,2,2,2,2,6,4	TO
ACASXU_1_7	ESBMC	0.1	1	2,2,2,2,2,4,2	TO
		0.3	1	2,2,2,2,2,2,2	TO
		0.5	1	2,2,2,2,2,2,2	TO
ACASXU_1_8	ESBMC	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	2,2,2,2,2,2,2	TO
		0.5	1	2,2,2,2,2,2,2	TO
ACASXU_1_9	ESBMC	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	2,2,2,2,2,2,2	TO
		0.5	1	2,2,2,2,2,2,2	TO

turn, NNs have 64 or 784 features in the input space that, from the verification perspective, result in a very large search space dimension to cover. In addition, CIFAR-10's NNs are affected by both factors, having 1024 features in the input space and thousands of neurons in total.

Regarding ACAS Xu, we find it possible to tune p and ϵ so we mitigate most of the associated quantization failures. That is, use a different norm (we use the L_∞-norm), and an allowing the difference in the outputs of the NNs to be larger. A different norm may imply a different meaning for equivalence and a larger ϵ a more loosely defined equivalence property. However, it should be done carefully so the resulting QNNs are still applicable and the chosen verifier can perform accordingly. Notice that, quantization failures occur in the context of the BSM, where equivalence is measured over concrete input samples (or counterexamples). The implication is that poorly defining these parameters can result in NNs with distinct outputs generated by the BSM.

Table 9 – Summary of the CEG4N's results for the ACAS Xu benchmark.

Model	Verifier	δ	No. Iter.	Bits	Status
	NNEQUIV	0.1	1		QF
		0.3	1		QF
		0.5	1	10,9,9,7,12,10,8	SQ
	NNEQUIV	0.1	1		QF
		0.3	1	7,8,7,8,9,9,5	TO
		0.5	1		QF
	NNEQUIV	0.1	1		QF
		0.3	1		QF
		0.5	1	8,7,8,8,8,9,6	TO
	NNEQUIV	0.1	1		QF
		0.3	1	5,6,5,6,6,8,4	TO
		0.5	1	7,6,6,6,6,7,4	SQ
	NNEQUIV	0.1	1	7,8,5,6,7,7,8	TO
		0.3	1	5,6,6,6,6,7,4	SQ
		0.5	1		QF
	NNEQUIV	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	2,2,2,2,2,2,2	TO
		0.5	1	2,2,2,2,2,6,4	TO
	NNEQUIV	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	2,2,2,2,2,2,2	VF
		0.5	1	2,2,2,2,2,2,2	TO
	NNEQUIV	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	2,2,2,2,2,2,2	TO
		0.5	1	2,2,2,2,2,2,2	TO
	NNEQUIV	0.1	1	2,2,2,2,2,2,2	TO
		0.3	1	2,2,2,2,2,2,2	TO
		0.5	1	2,2,2,2,2,2,2	TO

The last type of failures are the verification ones, which only occur in VM. We noticed them only when running CNNQUIV, with a total of 6 cases, i.e., a failure rate of 7.4%. They were caused by exceptions thrown by NNQUIV's software dependencies. However, it is unclear that a flaw in NNQUIV caused those exceptions. As we did not notice any verification failures with ESBMC, this can indicate its maturity when compared with NNQUIV.

The number of counterexamples is a critical indicator in the context of the CEG4N tool experiments. This metric represents the instances where the VM identified discrepancies between the original and quantized models, requiring the BSM to make adjustments in the next iteration to achieve equivalence. ESBMC managed to find counterexamples more consistently than NNEQUIV, that is, considering the same NNs and equivalence properties, and disconsidering the timeouts, ESBMC found more counterexamples. For example, for the Iris benchmarks, ESBMC found 26 against 5 counterexamples found by NNEQUIV.

And for the Seeds benchmarks, ESBMC found 56 counterexamples, while NNEQUIV found only 10. This suggests better handling of verification within bounded regions.

For simpler models like Iris and Seeds, an increase in δ generally led to more counterexamples. This indicates that larger bounded regions introduce more potential points of failure in the quantized models. This can be attributed to being more sensitive to quantization and less likely to hold the equivalence. This helps us understand particular cases such as (iris_4x2, ESBMC, δ =0.05), (seeds_4x1, ESBMC, δ = 0.05), (seeds_10x1, ESBMC, δ =0.03) where CEG4N run for more than 10 iterations until finally finding a QNN. The counterexamples found that the more iterations CEG4N has to run, the bigger the bit widths in the quantization scheme. When few or no counterexamples are found the quantization scheme tend to become stable at lower bit width values, see the results for iris_15x2 and seeds_15x1 with NNEQUIV as examples. For more complex NNs like MNIST and CIFAR, our results showed fewer counterexamples or none at all, often due to the verifier timing out before completing the verification process. ESBMC was particularly affected and frequently timed out, resulting in no counterexamples being reported.

The increase in counterexamples with larger input space regions (δ) suggests that ensuring equivalence becomes more computationally intensive as the input space expands. This indicates a trade-off between the size of the input region and the computational resources required. The results for seeds_10x1 and seeds_15x1 with ESBMC serve as an examples of that, and also the results for mnist-64-50x1 and mnist-784-50x1 with NNEQUIV are another examples.

Finally, the choice of VM is also important. As shown here, CESBMC and CNNQUIV present different behaviors that may suit a given scenario. For instance, regarding large NNs, CNNQUIV seems to be a clear choice. Moreover, if new verification techniques are introduced, one may consider other VM options, which our framework can accommodate due to its modularity.

These results answer our evaluation goals EG1, EG2, and EG3. Overall, these experiments show that CEG4N can successfully produce equivalent QNNs.

For simpler models, such as those in the Iris and Seeds benchmarks, CEG4N often achieves successful quantization. oth ESBMC and NNEQUIV verifiers showed successful quantization in many instances, particularly for simpler benchmarks. This suggests that CEG4N is robust in ensuring FE across different verification methods. Also, CEG4N adjusted the quantization scheme (number of bits) effectively for different bounded regions, achieving successful quantization in several configurations. This adaptability helps optimize the computational load by tailoring the quantization process based on the specified input region. This indicates that the tool effectively quantizes these models without requiring extensive computational resources. General observations are:

- verifier performance: ESBMC often resulted in timeouts, especially for larger and more complex models like those in the MNIST and CIFAR benchmarks. While NNEQUIV showed better performance in some benchmarks (e.g., Iris, Seeds), it also had issues with verification and quantization failures.

- impact of the bound region r: generally, as δ increased, the difficulty of achieving successful quantization increased, as indicated by more iterations and counterexamples and the occurrence of failures or timeouts.

- NN complexity: simpler NNs (e.g., smaller Iris and Seeds NNs) were more likely to be successfully quantized. While more complex NNs (e.g., larger MNIST and CIFAR NNs) posed significant challenges for both verifiers. Scalability should be a point of concern for larger NN models.

- quantization scheme: successful quantization often required varying the bit schemes, particularly for larger models. Higher complexity in the bit configuration was sometimes necessary for achieving SQ.

In our second set of experiments, we want to achieve the remaining experimental goals, i.e., EG4, EG5, and EG6. We primarily want to understand the impact of quantization processes performed by CEG4N on the accuracy of the resulting NNs compared with other post-training quantization techniques.

We selected the models for which CEG4N presented successful quantization processes and proceeded to quantize them also using GPFQ. Next, we collected the accuracy of the original and QNNs to compare them. Tables Table 10 Table 11, and Table 12 summarize the accuracy of the models quantized with CNNQUIV, CESBMC, and GPFQ, using the Iris, Seeds, and MNIST benchmarks. However, there was no successful run regarding CESBMC with MNIST, as already mentioned. We do not report the accuracy of CIFAR-10's NNs from VNNCOMP since CEG4N could not quantize them. Also, we do not report the accuracy of Acas Xu's models. It happened partially because of the same problem identified with the VNNCOMP's models but mainly because GPFQ requires access to training and test datasets, which are not public for Acas Xu.

The tables are organized as follows. Column Model shows the name of the NN models, Quantizer tells the name quantization technique (i.e., CESBMC, CNNQUIV, or GPFQ), δ informs the value used to define quantization Equivalence Properties, and columns Original Accuracy, Quantized Accuracy, and Accuracy Drop show the accuracy of the original and QNNs and their difference, respectively. Lastly, the column Equivalence Status tells whether original and QNNs are equivalent. The Accuracy Drop is positive if the QNN has a worse accuracy when compared with its original model. Otherwise, it is negative.

Table 10 – Comparison using Top-1 accuracy for NNs from dataset Iris quantized using CEG4N and GPFQ

Model	Quantizer	δ	O.Ac.	Q.Ac.	Ac.D.	Eq.St.
iris_4x2	GPFQ	0.010	86.667	60.000	**-26.667**	False
iris_4x2	CESBMC	0.010	86.667	86.667	0.000	True
iris_4x2	CNNQUIV	0.010	86.667	86.667	0.000	True
iris_4x2	GPFQ	0.030	86.667	60.000	**-26.667**	False
iris_4x2	CESBMC	0.030	86.667	86.667	0.000	True
iris_4x2	GPFQ	0.050	86.667	40.000	**-46.667**	False
iris_4x2	CESBMC	0.050	86.667	86.667	0.000	True
iris_4x2	CNNQUIV	0.050	86.667	93.333	6.667	True
iris_10x2	GPFQ	0.010	90.000	36.667	**-53.333**	False
iris_10x2	CESBMC	0.010	90.000	96.667	6.667	True
iris_10x2	CNNQUIV	0.010	90.000	93.333	3.333	True
iris_10x2	GPFQ	0.030	90.000	36.667	**-53.333**	False
iris_10x2	CESBMC	0.030	90.000	86.667	-3.333	True
iris_10x2	CNNQUIV	0.030	90.000	93.333	3.333	True
iris_10x2	GPFQ	0.050	90.000	36.667	**-53.333**	False
iris_10x2	CESBMC	0.050	90.000	90.000	0.000	True
iris_10x2	CNNQUIV	0.050	90.000	96.667	6.667	True
iris_15x2	GPFQ	0.010	90.000	93.333	3.333	False
iris_15x2	CESBMC	0.010	90.000	86.667	-3.333	True
iris_15x2	CNNQUIV	0.010	90.000	86.667	-3.333	True
iris_15x2	GPFQ	0.030	90.000	93.333	3.333	False
iris_15x2	CESBMC	0.030	90.000	86.667	-3.333	True
iris_15x2	CNNQUIV	0.030	90.000	86.667	-3.333	True
iris_15x2	GPFQ	0.050	90.000	93.333	3.333	False
iris_15x2	CNNQUIV	0.050	90.000	86.667	-3.333	True

Our findings show that the highest drops in accuracy for NNs generated by CNNQUIV where 3.3% for Iris, 21.43% for Seeds, and -6.20% for MNIST. For NNs quantized with CESBMC, the highest drops were 3.3% for Iris and 21% for Seeds. Regarding GPFQ, the highest noticed drops were 53.3% for Iris, -47.61% for Seeds, and -3.41% for MNIST. Such results are interesting and show that an increase in accuracy is even possible, which we briefly discuss next.

On the one hand, the highest accuracy drop for CEG4N-generated NNs coincides with ($\delta = 0.01$) (See Table Table 10). Indeed, small δ values increase the chance of not finding counterexamples to drive the quantization process, which favors the generation of poorly QNNs. For higher δ values ($\delta = 0.03$) and ($\delta = 0.05$), we notice that the highest accuracy drop is 3.3%. On the other hand, for GPFQ, the highest accuracy drops involve the Iris and Seeds benchmarks. We believe the GPFQ's performance is due to two factors: 1) GPFQ relies on representative data (e.g., data from the training dataset), which is more

Table 11 – Comparison using Top-1 accuracy for NNs from dataset Seeds quantized using CEG4N and GPFQ

Model	Quantizer	δ	O.Ac.	Q.Ac.	Ac.D.	Eq.St.
mnist_64_10x1	GPFQ	0.010	79.040	79.480	0.440	False
mnist_64_10x1	CNNQUIV	0.010	79.040	77.900	-1.140	True
mnist_64_10x1	GPFQ	0.030	79.040	79.480	0.440	False
mnist_64_10x1	CNNQUIV	0.030	79.040	77.900	-1.140	True
mnist_64_25x1	GPFQ	0.010	83.420	81.960	-1.460	False
mnist_64_25x1	CNNQUIV	0.010	83.420	82.820	-0.600	True
mnist_64_25x1	GPFQ	0.030	83.420	81.960	-1.460	False
mnist_64_25x1	CNNQUIV	0.030	83.420	83.220	-0.200	True
mnist_64_50x1	GPFQ	0.010	84.600	82.410	-2.190	False
mnist_64_50x1	CNNQUIV	0.010	84.600	77.280	-7.320	True
mnist_784_10x1	GPFQ	0.010	90.260	86.850	-3.410	False
mnist_784_10x1	CNNQUIV	0.010	90.260	89.170	-1.090	True
mnist_784_10x1	GPFQ	0.030	90.260	86.850	-3.410	False
mnist_784_10x1	CNNQUIV	0.030	90.260	89.940	-0.320	True
mnist_784_10x1	GPFQ	0.050	90.260	90.330	0.070	True
mnist_784_10x1	CNNQUIV	0.050	90.260	90.260	0.000	True
mnist_784_25x1	GPFQ	0.010	91.940	91.660	-0.280	False
mnist_784_25x1	CNNQUIV	0.010	91.940	91.350	-0.590	True
mnist_784_50x1	GPFQ	0.010	92.150	91.240	-0.910	False
mnist_784_50x1	CNNQUIV	0.010	92.150	85.950	-6.200	True

difficult for small datasets; and 2) small models are more sensitive to quantization.

Overall, the accuracy of models quantized with CEG4N is better for the Iris and Seeds benchmarks. This observation can be explained by differences in the quantization strategy of the techniques. GPFQ uses midthread quantization a form of scalar quantization. It quantizes each weight or value independently to the nearest quantization level centered around zero and reduces the precision of each weight independently. GPFQ quantization approach promotes sparsity in the quantized weights, forcing the small weights to zero. Iris and Seeds NNs are small models and have sparse weight matrix. Thus, the fact that the weight matrix of the NNs (NNs) being already sparse can explain the accuracy drop. GPFQ may introduce additional sparsity by setting small weights to zero. If a network is already sparse, further increasing sparsity can degrade performance because even fewer weights are left to contribute to the network's functionality. This can severely impact the network's ability to learn and generalize from the data.

On the other hand, the accuracy of models quantized with GPFQ is better for the MNIST benchmarks, but only by a small margin. The underlying objectives of the quantization techniques can explain this observation. CEG4N, although an optimization-

Table 12 – Comparison using Top-1 accuracy for NNs from dataset MNIST quantized using CEG4N and GPFQ

Model	Quantizer	δ	O.Ac.	Q.Ac.	Ac.D.	Eq.St.
mnist_64_10x1	GPFQ	0.010	79.040	79.480	0.440	False
mnist_64_10x1	CNNQUIV	0.010	79.040	77.900	-1.140	True
mnist_64_10x1	GPFQ	0.030	79.040	79.480	0.440	False
mnist_64_10x1	CNNQUIV	0.030	79.040	77.900	-1.140	True
mnist_64_25x1	GPFQ	0.010	83.420	81.960	-1.460	False
mnist_64_25x1	CNNQUIV	0.010	83.420	82.820	-0.600	True
mnist_64_25x1	GPFQ	0.030	83.420	81.960	-1.460	False
mnist_64_25x1	CNNQUIV	0.030	83.420	83.220	-0.200	True
mnist_64_50x1	GPFQ	0.010	84.600	82.410	-2.190	False
mnist_64_50x1	CNNQUIV	0.010	84.600	77.280	-7.320	True
mnist_784_10x1	GPFQ	0.010	90.260	86.850	-3.410	False
mnist_784_10x1	CNNQUIV	0.010	90.260	89.170	-1.090	True
mnist_784_10x1	GPFQ	0.030	90.260	86.850	-3.410	False
mnist_784_10x1	CNNQUIV	0.030	90.260	89.940	-0.320	True
mnist_784_10x1	GPFQ	0.050	90.260	90.330	0.070	False
mnist_784_10x1	CNNQUIV	0.050	90.260	90.260	0.000	True
mnist_784_25x1	GPFQ	0.010	91.940	91.660	-0.280	False
mnist_784_25x1	CNNQUIV	0.010	91.940	91.350	-0.590	True
mnist_784_50x1	GPFQ	0.010	92.150	91.240	-0.910	False
mnist_784_50x1	CNNQUIV	0.010	92.150	85.950	-6.200	True

based quantization technique, does not prioritize loss minimization (and hence accuracy maximization) as its primary goal. Instead, it focuses on ensuring equivalence between the original and quantized models. This means that CEG4N aims to preserve the inherent characteristics of the original model in the quantized version. On the other hand, GPFQ explicitly incorporates loss minimization into its objective, directly aiming to enhance model accuracy post-quantization. Therefore, while CEG4N ensures model equivalence, GPFQ's focus on loss minimization can result in better accuracy measures, as observed in the MNIST benchmarks. We find that the CEG4N's NN accuracy performance presents interesting results, given that it can produce QNNs relying only on a small set of representative data.

We have also conducted another experiment in which NNQUIV was used to verify equivalence between QNNs generated by GPFQ and their original counterparts. When at least one counterexample was found for every case, we considered the QNN not equivalent to the original one. We have observed that, out of 31 NNs generated by GPFQ, only 8 were in fact equivalent, which represents only 25.8% of the total amount of resulting NNs. Indeed, GPFQ was not designed to consider equivalence properties in its quantization approach, while CEG4N was. Nevertheless, these experiments prove that

statistical accuracy measures do not capture equivalence aspects. Moreover, it reinforces the benefits of formulating guarantees for properties (e.g., equivalence and robustness) that formal verification techniques can offer. In addition, we noticed that both CEG4N and GPFQ produced QNNs with better accuracy when compared with their original counterparts. Indeed, that is possible and has already been reported in the literature since the quantization techniques act as favorable weight regularization mechanisms that help NNs prevent biased behavior and provide better generalization (JIN; YANG; LIAO, 2019).

These results answer our experimental goals EG4, EG5, and EG6. Overall, these experiments show that CEG4N can successfully produce QNNs with accuracy figures similar to those obtained with other state-of-the-art techniques.

4.8 Summary

In this chapter we presented the experimental evaluation and the empirical findings of our research. The evaluation was conducted to assess the CEG4N framework's ability to quantize NNs while ensuring FE to their original counterparts. Key benchmarks, metrics, and configurations were examined to evaluate the framework's performance comprehensively. Below, we highlight the main topics we presented in this chapter.

The primary objectives of the experimental evaluation were as follows:

- **EG1: Effectiveness of CEG4N** - Demonstrating the framework's ability to generate quantized NNs that are functionally equivalent to their original models.
- **EG2: Performance Metrics Analysis** - Analyzing the computational efficiency and scalability of CEG4N across various benchmarks.
- **EG3: Equivalence Checking Techniques** - Evaluating the impact of different verifiers (ESBMC and NNEQUIV) on the quantization process.
- **EG4: Quality of Quantized Models** - Comparing the accuracy of quantized NNs generated by CEG4N with those produced by other quantization techniques (e.g., GPFQ).
- **EG5: Impact on Accuracy** - Investigating the preservation of accuracy in quantized models under equivalence constraints.
- **EG6: Impact on Model Size Reduction** - Assessing the trade-offs between size reduction and FE.

The evaluation used the following key metrics:

- **Timeout Failures:** Instances where verification exceeded the time limit.
- **Verification Failures:** Failures in the verifier module.
- **Accuracy Drop:** Difference between the accuracy of original and quantized models.
- **Equivalence Count:** Number of quantized models satisfying equivalence constraints.

Summary of configurations varied across:

- Equivalence relations: Top-1 classification equivalence $\approx_1$, and Norm-based approximate equivalence $_{p,\epsilon}$
- Output bounds for approximate equivalence: $\epsilon = 0.05$ and $p = \infty$
- Input region bounds: ($\delta = \{0.01, 0.03, 0.05\}$)
- Verifiers: ESBMC and NNEQUIV
- Benchmarks: (Iris, Seeds, MNIST, CIFAR-10, and ACAS Xu)
- CEG4N verification timeout: 20 minutes

Summary of our key finding:

1. Effectiveness and scalability: CEG4N successfully produced quantized models for simpler benchmarks (e.g., Iris, Seeds) with minimal accuracy drop. However, scalability issues were observed for larger models (e.g., MNIST, CIFAR-10), primarily due to verifier timeouts
2. Verifier performance: ESBMC demonstrated challenges with larger models due to bounded model-checking scalability limits, whereas NNEQUIV showed better performance in these scenarios but occasionally produced spurious counterexamples
3. Accuracy analysis: CEG4N generally preserved or slightly improved model accuracy compared to original models. GPFQ showed better accuracy for MNIST but failed equivalence verification for 74.2% of its quantized models
4. Equivalence guarantees: CEG4N consistently ensured FE, whereas GPFQ lacked this capability
5. Model size reduction: Larger input bounds (δ) and increased NN complexity resulted in more counterexamples and iterations, emphasizing a trade-off between computational resources and quantization quality

6. CEG4N effectively maintains equivalence in simpler NNs (e.g., Iris, Seeds), achieving minimal accuracy drops and maintaining functional integrity

7. Larger, complex models like MNIST and CIFAR-10 posed scalability challenges, especially with ESBMC verifier

8. Equivalence and accuracy of quantized NNs by CEG4N are superior to GPFQ, particularly for smaller, sparse models

9. Equivalence properties were not ensured by GPFQ, confirming CEG4N's advantage in safety-critical applications

Our evaluation and experiments demonstrated CEG4N's robustness in generating QNNs with formal guarantees of FE, particularly for simpler benchmarks. While challenges remain in scalability and computational demands for larger models, the framework shows promise as a reliable tool for equivalence-aware quantization.

5 Discussion

In this chapter we offer a critical analysis of the empirical evaluation conducted in Chapter 4, providing a reflection on how our findings help addressing the research problem. The chapter discusses the research implications for deploying NNs in resource-constrained environments and explores the broader impact on the field. Limitations of the research and potential areas for improvement are also discussed, providing a balanced view of the research outcomes.

5.1 Analysis of results

The findings in the preceding section, when examining the research goals, offer a thorough assessment of the effectiveness and constraints of the CEG4N framework in tackling the issues of NN quantization in resource-constrained domains. In the following, we present an evaluation based on the research objectives.

5.1.1 Addressing the quantization challenge for NNs in low-resource domains

The research demonstrates that the CEG4N framework effectively addresses the challenge of quantizing NNs for deployment in low-resource domains without significant accuracy degradation. Through its innovative approach combining search-based quantization with equivalence checking, CEG4N minimizes computational requirements while ensuring that the quantized NN (QNN) behaves similarly to its original version. This achievement directly aligns with the primary research objective and offers a promising solution to a significant barrier in the wider application of NNs.

5.1.2 To develop a framework for NN quantization

The development of the CEG4N framework is a cornerstone of the research objectives, aiming to provide a robust solution to NN quantization issues. The research results detail the framework's components, including the Bits Search Module (BSM), Abstractions Module (AM), and Verifier Module (VM), and describe their integrated function in the iterative process of refining NN quantization. The framework's successful implementation and operation, as presented in the research, indicate the achievement of this objective.

5.1.3 Evaluating the efficacy of the CEG4N framework

The research presents extensive experimental evaluation results across various benchmarks, including ACAS Xu, MNIST, Seeds, Iris, and CIFAR-10 datasets. These

results show that the CEG4N framework can successfully produce quantized NNs that maintain functional equivalence with their original counterparts. Success rates, quantization bit-width configurations, and instances of verification or quantization failure are discussed, providing a nuanced understanding of the framework's efficacy and limitations. The evaluation results suggest that, while the framework is broadly successful, its performance can vary depending on the complexity of the NN and the specific demands of the application domain.

5.1.4 Exploring the scalability and applicability of the CEG4N framework

The analysis of results concerning different NN sizes and architectures suggests that the CEG4N framework has varying degrees of scalability. As demonstrated in the benchmarks, success in quantizing smaller networks contrasts with the challenges encountered with larger, more complex networks, particularly those requiring extensive computation or possessing many features. These findings indicate that while CEG4N holds significant promise, further refinement may be necessary to enhance its scalability and applicability across all NNs, especially in real-world scenarios with strict resource constraints. As shown in our experimental evaluation, see Chapter 4, a sensible amount of CEG4N executions failed due to timeouts, especially when attempting the quantization of larger models. Timeouts indicate an inability of the verification tools to handle large models, highlighting that scalability is currently an issue. Increasing the time the verification tools are allowed to run can decrease the observed timeouts. Still, its effectiveness is limited to NNs as large as the one we list in our experiments. A more rewarding effort would be to invest time in improving the scalability of FEV tools. This would allow CEG4N to work with NNs larger than the ones in our experiments.

5.1.5 Advancing the field of NN quantization

The research contributes to the field of NN quantization by presenting a novel framework that tackles one of the key challenges in the area: the loss of accuracy and functionality post-quantization. The detailed analysis of the CEG4N framework, alongside the comprehensive evaluation across multiple benchmarks, offers valuable insights into the quantization process and the importance of equivalence checking. The findings contribute to the ongoing dialogue in the field, suggesting pathways for future research and potential improvements in NN quantization techniques.

Our findings, in relation to the research goals, emphasize the potential of the CEG4N framework to substantially influence the implementation of NNs in settings with limited resources. While showing significant achievements, the results also identify areas that warrant further exploration, especially regarding scalability and the framework's suitability for larger NNs. The CEG4N framework marks a notable advancement in NN

quantization, providing a hopeful strategy for addressing a key barrier to the extensive adoption of NNs in environments with resource constraints.

5.2 Implications of our findings

The results of the experimental assessment of the CEG4N framework have important implications for AI, especially in utilizing (NNs in settings with limited resources. The research shows the effectiveness of CEG4N in reducing the computational demands of NNs while maintaining their functionality, which is crucial for applications in environments with restricted computing resources. By proving the capability to quantize NNs efficiently and ensuring that their performance remains satisfactory in terms of behavior, not just computation, this study highlights the potential for wider use of AI technologies in scenarios like embedded systems, mobile devices, and other platforms where resources are limited.

The observations that GPFQ, despite not being designed to guarantee equivalence, still produces 25.8% of quantized models that are equivalent to the original and that CEG4N, while not designed to prioritize accuracy, still preserves a significant amount of original accuracy has several important implications and conclusions.

Firstly, these findings suggest an inherent overlap between the goals of loss minimization and equivalence preservation in the context of NN quantization. Although GPFQ focuses on minimizing quantized model loss, a notable percentage of its models remain equivalent, indicating that achieving low loss can sometimes align with maintaining functional equivalence. This implies that techniques designed to optimize accuracy might also benefit equivalence, albeit indirectly and inconsistently.

Conversely, CEG4N's ability to preserve a significant amount of the original accuracy, despite its primary focus on behavior equivalence guarantees, indicates that ensuring equivalence does not necessarily compromise accuracy. This suggests that the constraints imposed by the equivalence property inherently contribute to preserving the model's accuracy.

The key conclusion from these observations is that there is potential for developing hybrid quantization techniques that can simultaneously optimize for both equivalence and accuracy. By leveraging the strengths of both approaches, such methods could provide robust quantization solutions that ensure model reliability without sacrificing performance. Additionally, these findings highlight the importance of considering multiple objectives in quantization strategies, as optimizing for one aspect (e.g., accuracy) can inadvertently benefit another (e.g., equivalence).

Furthermore, these results underline the necessity of FV in the quantization process. The fact that a considerable proportion of GPFQ's models were not equivalent despite their

accuracy highlights the limitation of relying solely on statistical measures. As incorporated in CEG4N, FV provides crucial guarantees for applications where reliability and correctness are paramount.

In summary, these observations imply that achieving a balance between equivalence and accuracy is feasible and beneficial. They also point towards the potential development of more sophisticated quantization methods that incorporate formal verification and accuracy optimization to produce reliable and high-performing quantized NNs.

Moreover, the study emphasizes the significance of equivalence verification during the quantization phase, presenting a systematic approach to guarantee that the performance of quantized models remains consistent with their initial requirements. This is especially crucial for applications with high safety requirements like autonomous driving and medical diagnosis, where even slight deviations in model performance could lead to substantial outcomes. By incorporating search-based quantization and equivalence verification, CEG4N introduces a framework that addresses the demand for computational effectiveness while ensuring model precision and dependability.

5.3 Limitations and challenges

While CEG4N has made significant contributions, it is important to highlight some limitations and obstacles. One major constraint is the computational complexity of equivalence checking, especially when dealing with large and intricate NNs. This complexity hinders the scalability of CEG4N, indicating a necessity for more effective verification methods that can handle more complex models. Moreover, the research primarily revolves around ReLU-activated NNs, potentially restricting its relevance to networks utilizing alternative activation functions or structures, like recurrent NNs (RNNs) or generative adversarial networks (GANs).

Although the results demonstrated that CEG4N can generate QNNs while maintaining equivalence with the original ones, it is important to acknowledge that the architecture of the NNs (NNs) used in our evaluation may not fully represent the current state-of-the-art. The NNs employed in our evaluation had a limited number of layers and ReLU nodes compared to the more advanced models that can consist of hundreds of layers and thousands of ReLU nodes. The primary challenge arises from the current limitations of state-of-the-art verification algorithms, such as SMT or GPE, which struggle to scale to large NNs. Consequently, CESBMC and CNNEQUIV could only quantize 20% and 40% of the selected benchmarks, respectively, due to timeouts. Despite most unsuccessful quantization attempts, we have demonstrated that CEG4N is a feasible and adaptable approach, paving the way for enhancements with verifiers beyond ESBMC and NNEQUIV.

Scalability emerged as a significant limitation in our research, particularly affecting

the performance and applicability of the CEG4N framework. The high rate of timeouts observed with ESBMC, accounting for 67.90% of the total runs, underscores its struggles with handling larger and more complex NNs. This issue was particularly pronounced in datasets like MNIST, CIFAR-10, and AcasXu, where the computational demands overwhelmed the verifier, leading to substantial delays and incomplete processes. Even though NNQUIV showed better scalability with a lower timeout rate of 37%, it still faced challenges with larger networks. The need for multiple iterations further exacerbated these issues, highlighting the inherent difficulty in verifying and quantizing extensive NNs within a reasonable timeframe. These scalability challenges indicate that while CEG4N is effective for smaller models, there is critical to enhance the verification processes to ensure the framework can efficiently manage larger and more complex NNs. Addressing these limitations is crucial for expanding the practical applicability of CEG4N to a broader range of real-world scenarios.

The NN equivalence is relatively recent and lacks a well-established set of benchmarks that researchers can utilize. We address this limitation by proposing new benchmarks and the utilization of new datasets and NNs models, and ensuring they are made public available. While the benchmark selections in the experiments provide a good starting point, there is ample room for additional contributions. This study introduces a framework for NN quantization that integrates NN equivalence verification as a crucial process element. This distinctive approach differentiates our work from existing literature, as no directly comparable method is currently available. Moreover, our emphasis is on the practical and feasible aspects of this innovative quantization approach and the formalization of NN equivalence in terms of functional equivalence. Therefore, our discussion does not extensively explore the relationship between NN equivalence, functional equivalence, robustness verification, and their impact on NN accuracy and error, as our primary focus is on practical implementation rather than theoretical discussions.

An additional research gap lies in investigating the trade-offs between the level of quantization detail and the model's performance. The research predominantly focuses on uniform quantization methods and does not delve into mixed-precision techniques that could enhance precision and resource efficiency. Furthermore, the influence of quantization on the model's ability to withstand adversarial attacks has not been examined, leading to uncertainties regarding the security implications of using quantized models in adversarial settings.

5.4 Summary

In this chapter, we critically evaluated the findings of our research, emphasizing their contribution to addressing NN (NN) quantization challenges in resource-constrained

environments. We explored the implications for deploying NNs, highlighted the limitations of the proposed framework (CEG4N), and identified potential areas for improvement. Bellow we highlight the key aspects we covered in this chapter include.

Analysis of Results:

- The CEG4N framework effectively quantizes NNs while preserving functional equivalence, demonstrating minimal accuracy degradation.
- Components of CEG4N—Bits Search Module (BSM), Abstractions Module (AM), and Verifier Module (VM)—function iteratively to refine NN quantization.
- Evaluation across benchmarks (e.g., ACAS Xu, MNIST, Seeds, Iris, CIFAR-10) confirms the framework's efficacy but also reveals scalability issues, especially for larger networks.
- Scalability challenges, primarily due to verification tool limitations, restrict the framework's applicability to larger models. Addressing these challenges could improve its utility in real-world scenarios.

Implications of findings:

- CEG4N enables effective deployment of NNs in resource-constrained environments, with potential applications in embedded systems and mobile devices.
- The interplay between loss minimization and equivalence preservation highlights opportunities for hybrid quantization techniques.
- The necessity of equivalence verification (EV) is emphasized, especially for high-stakes applications like autonomous driving and medical diagnosis.

Limitations and challenges:

- Computational complexity of EV limits scalability for larger NNs.
- Focus on ReLU-activated networks restricts applicability to other architectures, such as recurrent NNs (RNNs) or generative adversarial networks (GANs).
- Current benchmarks lack diversity, and additional research is required to explore trade-offs in quantization precision, robustness, and resistance to adversarial attacks.
- Mixed-precision quantization techniques remain unexplored.

Contributions and future work:

- Introduced a novel framework integrating search-based quantization with EV.
- Proposed new benchmarks for NN quantization evaluation.
- Identified areas for improvement, including better scalability of verification tools and exploration of mixed-precision quantization.

6 Related work

This chapter surveys the literature on NN quantization, focusing on employed techniques, challenges encountered in preserving functional behavior post-quantization, and the role of formal equivalence verification and optimization in maintaining integrity. It starts by reviewing the topics of NN, NN quantization, FV techniques for NNs, and equivalence checking. Then, it identifies gaps in current methodologies and related work, underscoring the need for an effective approach to address these challenges.

In addition, it explores the existing research on the challenges and methodologies related to NN quantization, a process crucial for reducing the model size and computational requirements, and its impact on model behavior and performance. Furthermore, the literature review highlights the importance of explainability and safety in AI systems, particularly in high-stakes applications such as aerospace and edge computing. It also discusses the role of FV techniques in ensuring the functional equivalence and security of QNNs. Through this examination, the related work section aims to identify the gaps and limitations in NN quantization methodologies and proposes potential directions for future research to address these challenges.

In summary, this review situates the thesis within the broader field of NN research and quantization techniques, setting a foundation for it. It sets the necessary background so any reader can understand the meaning and relevance of each investigation and contribution.

6.1 NN quantization

NN quantization techniques have garnered significant attention due to their ability to compress NN models by reducing the precision of the weights and activations to lower bit widths (JACOB et al., 2017). This significantly decreases memory footprint, computational complexity, and power consumption, which is especially beneficial for deploying deep learning models on edge devices with limited resources (CHEN et al., 2020).

Quantization in NNs is a process that involves several key areas, including but not limited to the types of quantization, quantization-aware training (QAT) (NGUYEN; ALEXANDRIDIS; MOUCHTARIS, 2020), post-training quantization (PTQ) (WANG et al., 2018), and the study of the impact of quantization on model accuracy and performance. These techniques are pivotal for deploying sophisticated NNs on resource-constrained environments, precision-sensitive applications, and safety-critical systems.

One fundamental aspect of quantization is the distinction between uniform and

non-uniform quantization. Uniform quantization involves uniformly mapping continuous or large sets of values to a smaller set. In contrast, non-uniform quantization adapts to the distribution of the quantized parameters for better efficiency and accuracy (JACOB et al., 2017; BASKIN et al., 2018; GHOLAMI et al., 2022). Notably, non-uniform quantization has shown potential in achieving higher accuracy with minimal computational resources by adjusting to the data's inherent distribution (GAO et al., 2022; GHOLAMI et al., 2022). Furthermore, fast non-uniform quantization techniques that achieve efficiency and accuracy without extensive training or access to full training sets are highlighted, showcasing their applicability across various computer vision tasks (GAO et al., 2022). This efficiency makes them particularly valuable for real-world applications.

Quantization can be further classified as symmetric or asymmetric quantization, also often referred to as scale and affine quantization, respectively. The distinction between them lies in the choice of the clipping ranges. The choice between symmetric and asymmetric quantization is another critical consideration. Symmetric quantization centers the quantization range around zero, whereas asymmetric quantization allows for an arbitrary range, potentially improving the quantization of activations that do not center around zero(JACOB et al., 2017; NAGEL et al., 2021; GHOLAMI et al., 2022). The choice between asymmetric and symmetric quantization depends on the data distribution and the specific requirements of the NN application (JACOB et al., 2017; NAGEL et al., 2021).

Weight quantization focuses on reducing the bit-width of the weights of the NN, while activation quantization applies to the activations (outputs of layers). Both types are crucial for compressing NNs but might require different strategies to maintain performance. For instance, some techniques emphasize quantizing both weights and activations through a differentiable non-linear function that can be optimized end-to-end, showing superior performance in image classification and object detection tasks (YANG et al., 2019).

QTA involves simulating the effects of quantization during the training process, allowing a model to adjust its parameters to minimize the loss in accuracy caused by quantization. Besides, it is known for achieving better accuracy for quantized models than post-training quantization methods. However, it requires access to the training dataset and involves fine-tuning with quantization in mind(NAGEL et al., 2021; GHOLAMI et al., 2022).

PTQ applies quantization after the model has been trained. It is a simpler approach that does not require retraining but can sometimes lead to significant accuracy loss, especially for models quantized to very low bit-widths. Strategies have been developed to mitigate these losses, making PTQ a viable option for many applications (NAGEL et al., 2021; GHOLAMI et al., 2022).

Mixed-precision quantization employs different precision levels for different network parts based on their sensitivity to quantization. This approach allows for higher precision

where needed while still benefiting from the reduced complexity in other parts of the network. In summary, it allows for a better trade-off between model size, computational efficiency, and accuracy, effectively balancing quantization requirements (GARG et al., 2021).

Another significant advancement in the field is Hardware-Accelerated Quantization Hardware-Accelerated Quantization (HAQ). HAQ is a technique that involves designing or using hardware specifically optimized for quantized models, such as FPGAs or ASICs. These platforms can offer significant speedups and efficiency gains for quantized models by exploiting hardware features tailored to low-precision arithmetic. HAQ allows critical parts of a network to retain higher precision while less critical parts are quantized more aggressively(CHEN et al., 2021).

Several studies have explored mixed-precision quantization techniques, emphasizing the optimization-based strategies for allocating bit-widths across different layers or parts of NNs. These methods aim to balance the precision of NN parameters with preserving functional behavior integrity.

Wang, Ma, and Yang (WANG; MA; YANG, 2023) introduced a hardware-aware mixed precision quantization framework utilizing reinforcement learning to predict optimal bit-width allocation. Their approach adapts to low bit-width requirements and aims to maintain model accuracy while reducing computational resources. This method demonstrates the potential for intelligent and universal application in mixed precision quantization and embedded model deployment .

Chen, Wang, and Cheng (CHEN; WANG; CHENG, 2021) formulated mixed-precision quantization as a discrete constrained optimization problem, employing a second-order Taylor expansion to approximate the objective function . They introduced an efficient method to compute the Hessian matrix, reformulating the problem as a Multiple Choice Knapsack Problem (MCKP) and solving it via a greedy search algorithm. This principled approach is computationally efficient and has shown superiority over existing methods.

The technique proposed by Kimhi et al. (KIMHI et al., 2022), performs quantization by optimizing bit-width allocation by simulating specific hardware performance. Their method, aimed at reducing exploration space and carbon footprint, shows a significant trade-off improvement between accuracy and hardware efficiency. Author's approach highlights the importance of tailored bit allocation strategies for specific hardware deployments.

Moreover, evolutionary methods and crossbar-aware strategies have also been explored, focusing on automatically determining optimal bit-widths and enhancing robustness to non-ideal effects in RRAM-based accelerators (LIU et al., 2021; PENG et al., 2022). These approaches emphasize quantization strategies' dynamic and adaptive nature to meet various constraints, including hardware limitations and model accuracy.

In conclusion, the evolution of NN quantization techniques continues to push the boundaries of model efficiency and performance. From uniform and non-uniform quantization to hardware-accelerated approaches, the field is rich with innovative strategies designed to meet the diverse demands of modern neural network applications.

6.2 Background on FV for NNs

FV for NNs is a crucial area of research that focuses on proving or ensuring that an NN behaves as expected for all possible inputs. This is particularly important for safety-critical applications such as autonomous vehicles, medical diagnosis systems, and financial forecasting. QNNs, which use reduced precision for both computations and storage, present unique challenges and opportunities for FV due to their computational efficiency and potential alterations in behavior or performance due to quantization.

FV for NNs aims to ensure that a network behaves as expected under all possible inputs, which is crucial for safety-critical applications. Research by Seshia et al. (SESHIA et al., 2018) surveys the landscape of formal specifications for deep NNs, highlighting the importance of high-quality formal specifications for meaningful verification . Narodytska (NARODYTSKA, 2018) provides an overview of approaches to verifying NN properties, including encoding networks into Integer Linear Programs and using global optimization techniques.

Applying formal methods to machine learning, including NNs, introduces soundness, precision, and scalability challenges. These methods are limited when verifying systems that include machine-learned components, highlighting the need for advanced FV techniques (URBAN; MIN'E, 2021). One major challenge in the FV of NNs is scalability due to the high computational complexity involved in verifying large networks. Webb et al. (WEBB et al., 2018) propose a statistical verification approach that estimates the probability of property violations, offering a way to assess network robustness and scale to larger networks.

The literature on FV of NNs has applied various methodologies, including Satisfiability Modulo Theories (SMT)/SAT solving, model checking, interval analysis, abstraction methods, reachability analysis, and mixed-integer linear programming (MILP) (KATZ et al., 2017; HUANG et al., 2017; NARODYTSKA, 2018; HUANG et al., 2020). Here are examples of papers that exemplify major techniques in this field.

Verification of QNNs, which trade numerical precision for computational efficiency, poses unique challenges. Henzinger et al. (HENZINGER; LECHNER; ZIKELIC, 2020) show that verifying the bit-exact implementation of QNNs is PSPACE-hard, presenting heuristics to make verification more scalable. They developed a symbolic verification framework using SMT-based model checking to verify vulnerabilities in ANNs, especially

under conditions of quantization errors (SENA et al., 2021). Song et al, 2021 proposed a tool for verifying QNNs using SMT-Based Model Checking. Besides, they introduced a novel approach in which the NN and the safety properties are translated/encoded into C/C++ program and handled to FV tool that can verify properties of C/C++ programs (SONG et al., 2021).

Amir et al. (AMIR et al., 2020) presented an SMT-based technique for verifying binarized NNs, which are popular for memory and energy-efficient. This approach includes optimizations integrated into the SMT procedure and a method for parallelizing verification queries. Cheng et. al, 2017 studied the FV of Binarized NNs (BNNs) by encoding the verification problem as a *combinational miter* (BRAYTON; MISHCHENKO, 2010) and transforming it into a propositional satisfiability (SAT) problem, utilizing optimizations to enhance scalability (CHENG; NüHRENBERG; RUESS, 2017).

Boudardara et al. (BOUDARDARA et al., 2023) argued that abstraction methods are crucial for simplifying NN models, making verification faster and more feasible. They conclude that abstraction techniques for activation functions and model size reduction are particularly promising for feed-forward NNs. Ehlers (EHLERS, 2017), in turn, introduced an approach for verifying feed-forward NNs with piece-wise linear activation functions using global linear approximations and specialized verification algorithms for inferring additional node phases. Tran et al., 2017 proposed novel reachability algorithms for exact and over-approximation analysis of DNNs using star sets, improving verification speed significantly compared to previous methods (TRAN et al., 2019b).

6.3 Background on NN Equivalence Checking

At its core, equivalence checking, or formal equivalence verification, aims to ascertain that two models—typically an original and a modified version—behave identically across all possible inputs. This is challenging due to the high-dimensional input space of NNs and their non-linear nature(WANG et al., 2019; TEUBER et al., 2021).

Formal equivalence verification for NNs is a critical research area focusing on the validation and verification of NN models, ensuring that transformations or optimizations of a model do not alter its intended functionality(TEUBER et al., 2021). This topic is especially relevant in safety-critical applications, such as autonomous vehicles, medical diagnosis systems, and financial services, where the correctness of NN decisions is paramount(TEUBER et al., 2021; SAJI; AGRAWAL; SOOD, 2022). The literature on this subject spans theoretical frameworks, methodologies, and practical applications, reflecting its interdisciplinary nature and the growing interest in making AI systems reliable and trustworthy.

Various mathematical frameworks, including Boolean satisfiability (SAT) solvers,

Satisfiability Modulo Theories (SMT)(MUNAKATA et al., 2023; ELEFTHERIADIS et al., 2022; MATOS et al., 2022), and other formal methods, have been adapted for equivalence verification (BÜNING; KERN; SINZ, 2020; TEUBER et al., 2021; SESHIA et al., 2018). These frameworks provide the theoretical underpinnings for developing algorithms capable of handling the complexity of NNs.

Symbolic propagation techniques use symbolic representations of NN computations to analyze equivalence. These methods can handle certain transformations and optimizations in NNs by abstracting their computations symbolically(KHEDR; FERLEZ; SHOUKRY, 2020).

Partitioning strategies divide the input space into smaller, more manageable regions to manage the complexity and scalability of verification. By verifying equivalence in each region separately, these strategies can provide guarantees for the entire input space(SELIGMAN; SCHUBERT; KUMAR, 2015; CHENG; NüHRENBERG; RUESS, 2017).

Bounded model checking (BMC) approaches have been adapted for NNs, setting bounds on the inputs or the depth of computation for verification purposes. While not exhaustive, BMC can efficiently identify discrepancies in many practical scenarios (MATOS et al., 2024; ELEFTHERIADIS et al., 2022).

6.4 Related Work

In recent years, deploying NNs in various domains, especially in safety-critical and resource-constrained environments, has raised significant concerns regarding their efficiency, reliability, and interoperability (SZE et al., 2017; CHEN et al., 2020). As NNs become increasingly prevalent, the need to optimize these models for deployment on hardware with limited resources while ensuring their accuracy and robustness has become a focal point in the field of DL (HAN et al., 2015; WANG et al., 2018).

It is crucial to understand the gaps in existing research to effectively address the challenges of NN quantization. Quantization, while essential for deploying DNN models in resource-constrained environments, often introduces errors that compromise these models' accuracy and functional behavior (LOHAR et al., 2023). The literature highlights the need for interpretable, reliable, and secure AI systems, especially in safety-critical and resource-constrained contexts (CHEN et al., 2020; HUANG et al., 2020). This underscores the importance of balancing model compression with performance preservation (JACOB et al., 2017; LOHAR et al., 2023).

The primary concern is that current quantization techniques overlook quantized models' behavioral integrity (LOHAR et al., 2023). They often rely on statistical measures

that do not fully capture the impact of quantization on model behavior, potentially leading to significant performance degradation in specific scenarios (LOHAR et al., 2023). Furthermore, existing methods fail to guarantee that quantized models will behave equivalently to their full-precision counterparts across all possible inputs (LOHAR et al., 2023).

It is worth noticing that the present thesis proposes a novel quantization technique that addresses these limitations. The proposed method ensures that quantized models maintain their original decision-making processes and functional behavior by focusing on formal verification and equivalence checking. The following sections delve into the related work, exploring the current state of quantization techniques and identifying the gaps this research seeks to fill.

6.4.1 Concerns and limitations of NNs

The problem of quantization changing the behavior of NNs has been a critical focus in the field of DL, especially regarding the efficient deployment of these models on hardware with limited resources. Quantization compresses NNs by reducing the numerical precision of weights and activations, which can significantly affect their functional behavior due to the introduced quantization error. The challenge is minimizing this impact while balancing model size, computational efficiency, and accuracy.

In safety-critical domains, AI and ML models must be accurate and interpretable by humans. This ensures that decisions made by AI systems can be understood and trusted by end-users (BRUNTON et al., 2020).

The integration of NNs and AI in safety-critical applications requires careful consideration. Such applications demand high reliability (HUANG et al., 2020), robustness (YANG et al., 2021), safety (KATZ et al., 2017; HUANG et al., 2017), and explainability (Barredo Arrieta et al., 2020). For instance, in the aerospace industry, there is a critical need for interpretable and certifiable machine learning techniques to ensure flight safety and efficiency (BRUNTON et al., 2020).

In resource-constrained environments, such as Internet of Things (IoT) networks, edge computing emerges as a solution that brings processing closer to data sources, reducing latency and bandwidth use. However, applying AI in these contexts involves challenges like managing computational complexity and ensuring efficient resource orchestration (CHEN et al., 2020; BOURECHAK et al., 2023).

Applying AI in environments with limited computational resources requires algorithms optimized for efficiency(JACOB et al., 2017; NAGEL et al., 2021). Techniques for model optimization, like model compression (CHENG et al., 2018; MARINÓ et al., 2023), which includes model pruning(LAHAV; KATZ, 2021; HAN; MAO; DALLY, 2016) and quantization (XU et al., 2018; CAI et al., 2020), are crucial for enabling smart

applications in such contexts (BOURECHAK et al., 2023). In both safety-critical and resource-constrained environments, ensuring the privacy and security of data processed by AI systems is paramount. Techniques such as secure multi-party computation, homomorphic encryption, and differential privacy are being explored to address these concerns (WU et al., 2020a).

6.4.2 Impact of Quantization on Model Accuracy and Performance

Quantization can significantly reduce model size and computational requirements, enabling deployment on edge devices (JACOB et al., 2017). However, this often comes at the cost of reduced model accuracy. The extent of accuracy loss and performance gain depends on various factors, including the quantization strategy, model architecture, and application domain (JACOB et al., 2017; MISHRA; GUPTA; DUTTA, 2020; NAGEL et al., 2021). A critical area of research is understanding and mitigating the impact of quantization on model performance. Techniques such as knowledge distillation, where a quantized (student) model is trained to mimic a full-precision (teacher) model, and robust training methods to improve the quantized model's generalization are among the strategies explored. More advanced techniques, such as QAT and mixed-precision quantization, have been developed to mitigate accuracy loss while maximizing performance benefits.

Quantization techniques have garnered significant attention due to their ability to compress NN models without severely compromising performance (JACOB et al., 2017; MISHRA; GUPTA; DUTTA, 2020). These techniques are pivotal for deploying sophisticated models on portable devices with limited computational resources, memory, and power (HAN et al., 2015; HAN; MAO; DALLY, 2016). A range of quantization methods exists, each with its unique strategy and focus. For instance, some techniques emphasize quantizing both weights and activations through a differentiable non-linear function that can be optimized end-to-end, showing superior performance in image classification and object detection tasks (YANG et al., 2019; HUBARA et al., 2017). Others have delved into post-training quantization (PTQ) and quantization-aware training (QAT) as methods to mitigate the impact of quantization noise on network performance, with PTQ being highlighted for its minimal requirements and QAT for enabling lower bit quantization with competitive results (NAGEL et al., 2021).

Guo (GUO, 2018) provides a comprehensive review of QNNs and presents a deep dive into the current challenges and trends in the field, emphasizing the balance between predictive performance and computational efficiency. The topic of vector quantization for NNs has been explored by Chu and Bose *Vector quantization of NNs*(CHU; BOSE, 1998), who demonstrated the trade-offs between implementation complexity and performance through various quantization schemes. Fast non-uniform quantization techniques that achieve efficiency and accuracy without extensive training or access to full training sets

are highlighted, showcasing their applicability across various computer vision tasks (GAO et al., 2022).

Several methods have been suggested for NN quantization, showing encouraging outcomes on machine learning benchmarks (CHENG et al., 2018; LIN; TALATHI; ANNAPUREDDY, 2016; ZHOU et al., 2017; HAN; MAO; DALLY, 2016; ZHANG; ZHOU; SAAB, 2023). However, these schemes are not suitable for safety-critical systems, primarily due to two factors (HUANG et al., 2017; HUANG et al., 2020; SENA et al., 2021; SONG et al., 2021; LOHAR et al., 2023). First, they usually monitor the loss and accuracy degradation of quantized NNs (QNNs) with statistical measures defined in the training set (YANG et al., 2020; GHOLAMI et al., 2022; LOHAR et al., 2023). In other words, the emphasis is on dynamically evaluating the classification accuracy of specific test datasets. The second factor is that these models lack soundness as they cannot ensure accurate classification for every possible input (GHOLAMI et al., 2022; LOHAR et al., 2023). Such a characteristic is crucial in the context of safety-critical systems (LOHAR et al., 2023).

Statistical accuracy measures do not capture the vulnerability of a network to adversarial input. There may be specific inputs for which the behavior of an NN degrades significantly (GHOLAMI et al., 2022; HUANG et al., 2017). These metrics are paramount for evaluating the performance and security of QNNs but do not fully capture the impact of quantization on the NN's behavior. While they provide insights into the overall performance of QNNs, they also offer a high-level view that might overlook subtle yet critical changes in model behavior due to quantization. For instance, minor shifts in decision boundaries might not significantly impact overall accuracy but could lead to incorrect predictions for specific inputs.

6.4.3 Preserving the functional behavior of QNNs

The literature on preserving QNNs functional behavior highlights several challenges and solutions in the field. In the following, an overview based on the recent papers offers insights into the challenges and advancements in the field of QNNs, focusing on preserving functional behavior while optimizing for efficiency and performance.

In terms of quantization and its effects. Quantization applied to NNs refers to representing weights and activations in low-precision formats and poses challenges such as performance degradation due to quantization errors. A comprehensive review by Guo (GUO, 2018) discusses different aspects and current trends in the field of QNNs, highlighting the balance between model size reduction and performance preservation. Liang et al. (LIANG et al., 2021), in turn, provided a comprehensive survey on network compression techniques, including pruning and quantization, to accelerate DNN while maintaining high accuracy. Besides, Xu et al. (XU et al., 2023) introduced a novel framework for image classification that incorporates innovative quantization functions and self-distillation methods to reduce

quantization errors and improve model performance.

Chatterjee, A. and Varshney, L. (CHATTERJEE; VARSHNEY, 2017) developed an approach to quantizing deep networks using high-rate quantization theory, aiming for an optimal quantizer that minimizes performance loss while significantly reducing model size. Umuroglu, Y. and Jahre M. (UMUROGLU; JAHRE, 2017) addressed the challenge of deploying QNNs on mobile CPUs, where operations on highly quantized datatypes are not natively supported. They proposed a streamlining flow to convert QNN inference operations to integer ones, demonstrating a significant speedup with minimal loss of accuracy. Hubara et al. (HUBARA et al., 2016) introduced a method to train QNNs with low-precision weights and activations, achieving model performance comparable to its 32-bit counterparts.

Several studies focus on synchronization and control issues in quantized memristive and chaotic NNs. For example, (SUN et al., 2020) and (WANG et al., 2021a) explore quantized synchronization control in memristive NNs with time-varying delays, proposing quantized control schemes to ensure synchronization with reduced computational complexity. (PARK et al., 2017) tackle the quantization of memory-augmented NNs (MANNs), identifying memory address as a primary source of performance degradation in quantized MANNs and proposing robust quantization methods to address this challenge.

Contrary to the general belief that quantization always leads to performance degradation, some studies have found that quantization can sometimes improve accuracy by imposing regularization on weight representations and achieving significant memory reductions. This finding is presented in *Quantization of deep NNs for accurate edge computing* (CHEN et al., 2021).

To address these concerns, the field of FV offers a set of techniques designed to ensure the correct behavior of systems (CLARKE et al., 2018). FV methods aim to mathematically prove the correctness of algorithms, providing guarantees that they meet certain specifications (CORDEIRO; FISCHER; MARQUES-SILVA, 2012; BESSA et al., 2016; RAMALHO et al., 2013). In the context of NNs, these techniques can be applied to assess properties such as robustness and safety (HUANG et al., 2017; YANG et al., 2021), ensuring that an NN behaves as expected, even in the face of adversarial inputs and exploitation attempts (HUANG et al., 2017; GEHR et al., 2018). Verifying NN safety is a challenging task but several recent approaches and tools have been proposed and are already available (RUAN; HUANG; KWIATKOWSKA, 2018; KATZ et al., 2017; SENA et al., 2021; SONG et al., 2021).

In this direction, some works propose specific verification techniques for QNNs (ZHOU et al., 2017; ZHANG et al., 2022; ZHANG; SONG; SUN, 2023). However, these techniques tend to focus on preserving loss, accuracy, and robustness to adversarial inputs as a proxy for NN behavior, which, to some extent, suffers from the same problems

discussed earlier. Focusing on adversarial robustness is essential, given the increasing sophistication of adversarial attacks (HENDRYCKS; DIETTERICH, 2019). However, this approach primarily assesses QNN "security" rather than fidelity to an original counterpart, i.e., it does not assess the degree of fidelity a QNN presents regarding an original NN. Consequently, it does not guarantee that a quantized model faithfully represents the decision-making process of its predecessor.

Consequently, a proper way to ensure that a QNN has preserved its behavior compared to its original counterpart is to reformulate the verification problem under the notion of equivalence checking (EC) (BÜNING; KERN; SINZ, 2020; TEUBER et al., 2021; ELEFTHERIADIS et al., 2022). EC, a specific form of formal verification, states that two NNs are equivalent when they produce similar outputs for the same inputs in a given domain. It involves verifying that two NN versions, e.g., before and after quantization, behave identically under a wide range of conditions. This process is vital to ensure that quantization techniques can produce QNNs that are safe to be deployed in resource-constrained environments. In other words, it ensures that QNNs do not compromise the integrity or safety of the systems that rely on them. Thus, the risks associated with NN quantization can be mitigated through EC.

6.4.4 Limitations of FV and FEV

Scalability and efficiency are the main issues with FEV applications in NN. As NNs become more complex, scalability remains a significant challenge for formal equivalence verification. Research is ongoing into more efficient algorithms and heuristics that can handle the scale of modern NNs (SHIH; DARWICHE; CHOI, 2019; DETHISE; CANINI; NARODYTSKA, 2021).

The coverage of non-linear transformations pose significant challenges to existing verification methods. Developing techniques that can accurately and efficiently verify equivalence in non-linearity presence is an area of active research (NARODYTSKA, 2018; HENZINGER; LECHNER; ZIKELIC, 2020).

Integrating formal equivalence verification into NN development and deployment pipelines is crucial for its adoption. Efforts to make these tools more accessible and to educate practitioners on their importance are needed(NARODYTSKA, 2018; HENZINGER; LECHNER; ZIKELIC, 2020).

In computational theory, problems are classified into complexity classes based on the resources needed for resolution (LOUI, 1996). P (polynomial time) and NP (nondeterministic polynomial time) are two primary classes. P encompasses problems solvable in polynomial time, making them relatively simple. At the same time, NP includes problems where a potential solution can be verified in polynomial time, even if finding the solution is

challenging. Verification tasks often fall under NP-hard or undecidable categories, requiring substantial computational resources for resolution if solvable. On the other hand, validating a solution's correctness (verification) typically falls within P, making it computationally more manageable(LOUI, 1996; GAO; XU, 2014; GHANEM; SINIORA, 2021).

The verification process can be understood by comparing search problems with decision problems. When dealing with a verification problem, one must explore many potential states or arrangements to demonstrate the existence or absence of specific behaviors or characteristics(BELLARE; GOLDWASSER, 1991). As the size of the system increases, the search space expands exponentially, making this a computationally demanding task. On the contrary, validating the correctness of a solution constitutes a decision problem, where the focus is on confirming whether a given solution or proof satisfies the defined criteria (LOVASZ et al., 1991). This verification step does not involve examining every possible state, but involves directly assessing the solution against the established criteria.

In the realm of FV, establishing the correctness of a system entails developing a demonstration that shows that the system's actions conform to its specifications across all potential inputs and scenarios (CLARKE; GRUMBERG; PELED, 1999; BAIER; KATOEN, 2008). This task can be highly challenging due to its intricate nature and many conditions that must be considered. On the other hand, validating a proof's correctness involves verifying its logical progression to confirm its compliance with logical and mathematical principles (NIPKOW; PAULSON; WENZEL, 2002; AVIGAD; HARRISON, 2014). This validation process is typically more direct as it does not involve creating the proof but scrutinizing an already established one .

Model checking and theorem proving are two methodologies within formal methods that illustrate this concept. Model checking focuses on automatically validating finite-state concurrent systems against specified properties (CLARKE; GRUMBERG; PELED, 1999; BAIER; KATOEN, 2008). The main difficulty arises from the state explosion issue, where the number of states increases exponentially with the quantity of components. Nonetheless, checking if the model upholds this property for a given state becomes uncomplicated once a particular model is confirmed to meet a property. On the other hand, theorem proving entails establishing proofs for theorems related to a system (GORDON; MELHAM, 1993; HARRISON, 2009). Constructing these proofs can be highly intricate, yet verifying that the proof steps adhere to logical principles is comparatively more straightforward.

The challenge in addressing verification issues stems from the requirement to examine a comprehensive range of options and guarantee correctness under all possible circumstances. On the contrary, validating the correctness of a solution involves confirming a particular case against established criteria, a task that is less computationally intensive and easier to grasp conceptually. This core differentiation highlights the difficulties and strategies in utilizing formal system development and verification methods.

6.4.5 The need for a new approach

Preserving the functional behavior of QNNs involves addressing quantization effects on performance, developing efficient deployment strategies, and exploring novel quantization and control techniques. The balance between model size reduction and accuracy retention remains a primary focus, with ongoing research to overcome these challenges. As the literature review shows, several attempts have been made to address and tackle the trade-off between model compression and performance degradation. The literature review on NN quantization in the previews sections and the discussions regarding how quantization changes the behavior of NNs reveal several gaps and limitations in existing quantization methodologies. These gaps highlight the complexity and challenges in achieving effective quantization without compromising NN performance.

Current quantization techniques often overlook the behavioral integrity of quantized models and the changes introduced by quantization that may cause them to behave slightly differently than their floating-point counterparts – often using the performance metrics of floating-point models as those for quantized models without considering the alterations caused by quantization (YANG et al., 2022). The accuracy gap between full-precision and quantized models is partially attributed to quantization error. Research in binary quantization, which maps values to -1 and 1, indicates the need to analyze different scaling strategies and develop novel quantization techniques that minimize this error(POURANSARI; TUZEL, 2020).

The noise introduced by quantization is a significant factor leading to performance degradation. There is a critical need for algorithms that mitigate the impact of quantization noise on network performance(JACOB et al., 2017; NAGEL et al., 2021). The Minimum Squared Error (MSE) criterion is often used to measure the discrepancy between the original (high-precision) weights and the quantized (low-precision) weights (CHOUKROUN; KRAVCHIK; KISILEV, 2019). The goal is to minimize this error during quantization to ensure that the quantized network performs as closely as possible to the original network (CHOUKROUN; KRAVCHIK; KISILEV, 2019).

MSE quantifies the average squared difference between the original and quantized weights. By minimizing this error, you aim to retain the original network's behavior as much as possible. The lower the MSE, the closer the quantized network's behavior is to the original network. NNs are highly sensitive to weight changes, as small perturbations in weights can significantly alter their behavior(CHOUKROUN; KRAVCHIK; KISILEV, 2019). Therefore, minimizing the MSE helps preserve the network's behavior by ensuring that the quantized weights closely resemble the original weights (XU et al., 2018; CHOUKROUN; KRAVCHIK; KISILEV, 2019). Minimizing MSE in quantization can contribute to maintaining the network's ability to generalize to unseen data and perform well on various tasks (PARK; AHN; YOO, 2017). This is because the quantized network

retains similar weight distributions and thus similar decision boundaries compared to the original network.

However, there are limitations to the assumption that minimizing MSE guarantees behavior preservation in QNNs. NNs are inherently nonlinear systems. Minimizing the MSE may not fully capture the complex interactions between weights and activations (XU et al., 2018). Even a small change in weights can lead to nonlinear effects that unexpectedly alter the network's behavior. NNs often exhibit robustness to small perturbations in weights. However, quantization introduces significant perturbations, especially in low-precision settings. Minimizing MSE may not adequately address the network's sensitivity to such perturbations, leading to behavior divergence. Minimizing MSE may lead to overfitting the quantized network to the training data, especially if the quantization process is not regularized properly. Overfitting can hinder the network's generalization of unseen data and adaptation to different tasks. For complex NN architectures with multiple layers, non-linear activations, and intricate weight dependencies, minimizing MSE alone may not capture the full behavioral preservation requirements. Other factors such as activation distributions, layer interactions, and task-specific behaviors need to be considered(LIU; CAI; ZHUANG, 2021).

The growing need to formally verify NNs, especially to prove equivalence between an original and a compressed version, emphasizes the complexity of existing approaches. Exploring formal equivalence checking in this area could provide a more rigorous foundation for evaluating quantization strategies. Equivalence checking could be crucial in developing and evaluating such algorithms, ensuring that quantization does not significantly alter the network's behavior. This gap points to the need for new evaluation criteria that account for model equivalence in quantization, which could be supported by formal equivalence checking. This area reveals the potential of integrating formal equivalence checking more deeply into quantization processes to improve accuracy and computational efficiency.

In summary, while existing quantization and verification techniques for QNNs focus on essential aspects of model performance and security, they fall short of providing a broader view of how quantization affects a model's behavior. Consequently, this thesis proposes the CEG4N method to fill this gap by ensuring that quantized models remain true to the decision-making process, that is, the desired behavior of their original versions.

6.5 Summary

This chapter provides an extensive review of literature on neural network (NN) quantization and related topics, identifying key methodologies, challenges, and gaps. Key points include:

- Neural Networks (NNs) are foundational tools in artificial intelligence (AI) with diverse applications, but their deployment on resource-constrained environments necessitates efficient compression techniques like quantization;
- NN quantization reduces model precision, leading to smaller memory footprints and lower computational costs, but it can also impact model accuracy and functional behavior;
- The literature highlights several quantization methods, including uniform and non-uniform quantization, post-training quantization (PTQ), quantization-aware training (QAT), mixed-precision quantization, and hardware-accelerated quantization (HAQ);
- Challenges in quantization include preserving functional behavior, minimizing quantization noise, and ensuring robustness in safety-critical applications like aerospace and edge computing;
- Formal Verification (FV) and Equivalence Checking (EC) techniques are essential to ensure that quantized neural networks (QNNs) maintain functional equivalence with their full-precision counterparts, especially for high-stakes applications;
- Existing quantization methods often rely on statistical measures for accuracy evaluation, which may not fully capture the behavioral equivalence of quantized models;
- Gaps in current methodologies include insufficient consideration of behavioral integrity, scalability issues in verification processes, and limited integration of formal methods into quantization workflows;
- The need for advanced FV techniques and novel quantization approaches that prioritize equivalence checking is emphasized, aiming to bridge the gap between model compression and functional integrity.

These findings establish the foundation for the proposed CEG4N method, which integrates formal equivalence checking to ensure that quantized models preserve the original functional behavior of NNs.

7 Conclusion and future work

This chapter summarizes this thesis's key contributions to the NN quantization field. It reflects on the significance of the thesis findings and their potential applications. The chapter also outlines directions for future research, suggesting ways to build upon the thesis research and explore new avenues to improve NN quantization techniques.

7.1 Summary of key contributions

The main contributions of this thesis are:

1. A new FV technique that treats NN equivalence as a SMT problem
2. A new optimization-based quantization technique that formulates quantization as a multi-objective optimization problem that has equivalence as an objective
3. A modular and automated quantization framework that effectively combines FV and optimization to preserve the functional behavior of NNs during quantization
4. An empirical scalability study, evaluating the computational demands of quantizing and verifying the equivalence of neural network models. This includes comparing different equivalence-checking techniques and their impacts on the performance of CEG4N
5. Empirical results using multiple benchmarks, including Iris, Seeds, MNIST, and CIFAR datasets
6. Analysis of the effects of quantization on model accuracy and performance, demonstrating that maintaining functional equivalence does not necessarily compromise accuracy
7. Ensures that all tools, benchmarks, and results used in the research are publicly available, facilitating reproducibility and further research in the field of neural network quantization

This thesis introduces the CEG4N framework as a notable advancement in NN quantization. Its primary innovation lies in integrating search-based quantization methods with EC mechanisms. This integration decreases the computational demands of NN deployment and preserves the network's functionality post-quantization. This dual focus is crucial in resource-constrained settings where optimizing memory and computational resources is paramount. CEG4N has significantly influenced the field of NN quantization

by merging search-based quantization with EC in a unique manner. This strategy aims to lessen computational requirements while upholding model accuracy. The research contributes to understanding how quantization impacts NN performance, paving the way for future advancements in quantization strategies. Moreover, by leveraging various validation tools and exploring diverse network structures, the study broadens the potential applications of quantization methods across various artificial intelligence domains.

The core of this thesis's methodology lies in recognizing the crucial importance of preserving the behavioral similarity of NNs (NNs) after quantization. This is achieved through a systematic optimization and verification process that thoroughly compares quantized models with their original versions to detect any differences in functionality. A key feature of this approach is the use of EC, a unique element that distinguishes CEG4N from other quantization methods that usually employ less stringent accuracy evaluations.

Moreover, this study illustrates how the versatility of the CEG4N framework extends to many NN architectures and sizes. By conducting thorough experiments on various benchmarks, encompassing both large and small NNs on datasets like CIFAR-10, MNIST, Iris, and Seeds, the framework has shown its ability to generate quantized models that not only maintain behavioral equivalence but also, in some cases, achieve up to 163% higher accuracy compared to state-of-the-art quantization methods. These findings highlight the significant potential of the framework to improve the efficiency of deploying NNs in diverse application areas, particularly those where model accuracy and behavior integrity are crucial.

The CEG4N framework's importance goes beyond its direct practical uses. It initiates a change in perspective towards incorporating EC as a key element of the NN quantization process. This helps fill a crucial gap in implementing advanced AI models in settings with limited computational and memory capacities. Furthermore, the framework establishes a solid basis for future research directions, such as developing more effective EC approaches, broadening quantization techniques to support various network structures, and exploring the impact of quantization on model resilience to adversarial attacks.

7.2 Summary of key findings

This section presents a comprehensive overview of the significant discoveries and insights gained from our research on the CEG4N framework for quantizing neural networks (NNs). These findings highlight CEG4N's strengths and limitations, providing a detailed analysis of its performance across various benchmarks and under different conditions. By examining CEG4N's effectiveness, scalability, and accuracy preservation capabilities, we aim to underscore its potential and the areas that require further improvement.

Key discoveries consist of (1) CEG4N can efficiently quantize NNs, resulting in

models that exhibit up to 163% higher accuracy than current cutting-edge methods; (2) The framework illustrates the significance of EC in preserving the functional integrity of quantized models;(3) CEG4N showed some degree of versatility, applying to a range of neural network architectures, from smaller networks like Iris and Seeds to larger ones like MNIST; (4) Our research showed the viability of a modular quantization framework NN quantization that provides guarantees of NN equivalence; However, (5) the research also revealed that CEG4N, encountered significant scalability issues with the verifiers, leading to a high number of timeouts, especially with larger networks. This highlights the need for further improvements in the verification process to handle more complex models efficiently.

In addition, our results revealed several interesting patterns and findings. CEG4N, when running with NNQUIV, exhibited a higher success rate of 40.74%, significantly outperforming its performance with ESBMC, which had a success rate of 20.99%. This difference highlights NNQUIV's greater effectiveness in handling larger and more complex networks.

A notable issue with ESBMC was its high rate of timeouts, accounting for 67.90% of the total runs, compared to 37% for NNQUIV. This pattern suggests that ESBMC struggles with scalability and managing the computational demands of larger neural networks, as evidenced by the increased timeouts for datasets like MNIST, CIFAR-10, and AcasXu. Additionally, ESBMC required more iterations, averaging four per successful run, while NNQUIV averaged only two. This discrepancy can be attributed to the differing verification approaches, with ESBMC's SMT approach potentially generating more counterexamples.

Quantization failures were particularly prevalent in the AcasXu benchmarks, indicating that these networks are highly sensitive to errors introduced during quantization. Higher parameter values r also led to more iterations and increased computational demands, exacerbating quantization challenges. This trend was observed in the Seeds dataset, where larger r values correlated with increased timeouts and quantization issues.

When comparing the accuracy of QNNs, those generated by CEG4N generally showed better performance for the Iris and Seeds benchmarks. The accuracy drops for CEG4N-generated NNs were significantly lower than those for the GPFQ method, which experienced substantial accuracy losses. For example, GPFQ showed accuracy drops up to 53.3% for Iris, whereas CEG4N's highest drop was only 3.3% for the same dataset. On the other hand, for the MNIST benchmarks, GPFQ performed slightly better in terms of accuracy, although the margin was small. Considering the CEG4N does not seek to optimize the accuracy of its quantized models, our research suggests that the constraints imposed by the equivalence property inadvertently contribute to preserving the model's accuracy.

Verification failures were only observed with NNQUIV, accounting for 7.4% of the total runs. These failures were likely due to exceptions thrown by the software dependencies.

In contrast, ESBMC did not exhibit such failures, suggesting greater software maturity and reliability.

The experiments also showed that both CEG4N and GPFQ could occasionally improve the accuracy of the original models due to the regularization effect of quantization. This phenomenon, where quantization acts as a weight regularization mechanism, helps neural networks prevent biased behavior and enhance generalization. However, it was clear that ensuring equivalence is a significant challenge for GPFQ, as only 25.8% of the generated QNNs were found to be equivalent to their original models.

The findings indicate that NNQUIV is more suitable for larger neural networks due to fewer timeouts and better scalability. At the same time, ESBMC remains reliable for smaller models with fewer verification failures. These results underscore the importance of FV techniques in maintaining model equivalence and suggest that CEG4N offers a robust approach to generating QNNs with comparable accuracy to state-of-the-art techniques, particularly for smaller datasets.

7.3 Future directions

In this section, we outline several promising avenues for advancing the field of NN quantization and NN equivalence. We focus on potential future research and directions to enhance existing methodologies' efficiency, scope, scalability, and robustness, especially those whose limitations were highlighted in this thesis. Key areas of focus include improving the efficiency of EC techniques, exploring a more diverse set of NN structures, investigating alternative quantization approaches, assessing the robustness of quantized models against adversarial attacks, and extending the CEG4N framework to support QAT techniques. Each of these directions presents unique opportunities to address existing challenges and push the boundaries of what is possible in NN quantization and FEV, ultimately contributing to more efficient and secure NNs deployment.

7.3.1 Improving the scalability of the FEV techniques

EC or FEV is a cornerstone of the NN quantization performed by CEG4N. It ensures that the functional integrity of the model is maintained post-quantization. However, one of the most significant challenges faced by current EC methodologies is timeouts and scalability, particularly as models grow in size and complexity. Addressing these issues is crucial for practically deploying quantized models in real-world applications.

Timeouts in EC arise when the verification process exceeds a predefined time limit, failing to conclusively determine equivalence or discrepancy between the original and quantized models. These timeouts are especially prevalent in large-scale models where the

vast state space needs to be explored. Future research can explore several strategies to mitigate timeout issues:

1. One promising approach is incremental verification, where the model is verified in smaller, manageable segments rather than as a whole. The likelihood of encountering a timeout is reduced by breaking down the verification task into sub-tasks, each corresponding to a smaller section of the model. Techniques such as compositional verification, which assembles the verification results of individual components, could be particularly beneficial.

2. Heuristics can guide the verification process toward the most likely areas of discrepancy, thereby reducing the search space and the time required for verification. Machine learning-based heuristics, which learn from past verification instances, can dynamically adapt to the model's characteristics, providing a more efficient search process.

3. Leveraging the power of parallel and distributed computing can significantly reduce the verification time. By distributing the verification task across multiple processors or machines, the overall time required can be reduced, thus mitigating timeout issues. Research into efficient parallel algorithms tailored for EC could further enhance this approach.

Scalability is another critical challenge in EC, with current techniques often struggling to handle the increasing complexity and size of modern NNs. Several research directions can be pursued to improve the scalability of these techniques:

1. Probabilistic approaches, such as Monte Carlo methods, can provide scalability by approximating the EC process. Instead of exhaustively checking all possible states, these methods sample a subset of states, providing a probabilistic guarantee of equivalence. This approach can dramatically reduce the computational burden, making it feasible to verify larger models.

2. Symbolic methods, including Satisfiability Modulo Theories (SMT) solvers, represent another promising avenue. These methods abstract the model into symbolic representations, allowing for more efficient manipulation and checking of equivalence. Enhancing the SMT encoding techniques to better handle the specifics of NN quantization can lead to significant scalability improvements.

3. Techniques such as model reduction, where the complexity of the NN is reduced without significantly affecting its functionality, can be employed before EC. By simplifying the model, the verification process becomes more manageable. Research

into advanced reduction techniques that preserve critical model characteristics while minimizing size could greatly benefit scalability.

4. Developing adaptive verification frameworks that adjust their verification strategy based on the model's characteristics can also enhance scalability. These frameworks can dynamically choose the most appropriate verification technique, effectively balancing precision and computational resources.

5. Integrating EC into the NN compression process can create a unified framework that inherently considers verification during model reduction. This integrated approach can streamline the entire quantization and verification pipeline, making it more scalable and efficient.

Addressing the issues of timeouts and scalability in EC for NN quantization is essential for the CEG4N and also means advancing the EC field. Researchers can develop more robust and scalable verification methodologies by exploring incremental verification, heuristic-guided approaches, parallel computing, probabilistic methods, symbolic techniques, model reduction, and adaptive frameworks and find applicability in a broader range of NN architectures. These advancements will benefit CEG4N and significantly impact the AI landscape by enabling the deployment of quantized NNs across various applications and environments, ensuring both efficiency and reliability.

7.3.2 Improving the efficiency of EC techniques

Investigating a wider range of methods for verifying equivalence is a compelling path. For example, analyzing reachability (TEUBER et al., 2021) and encoding using SMT present viable options for guaranteeing model equivalence after quantization with enhanced precision and effectiveness. The research conducted by Zhang et al. (ZHANG; SONG; SUN, 2023) on verifying error bounds introduces a fresh outlook on quantization and its impact on NN behavior, showcasing the various strategies that can enrich the advancing field of NN quantization studies.

Enhancing the SMT encoding of the quantization problem to evaluate its efficiency compared to CEG4N and move towards creating a unified framework for NN compression and equivalence has the potential to provide novel perspectives and findings in the quantization domain. Subsequent studies could enhance the formalization of NN equivalence, such as integrating the QEBVerif approach into CEG4N and conducting comparative assessments. By integrating advanced quantization and equivalence-verification techniques, CEG4N is poised to enhance its outcomes by attaining a more optimal trade-off between precision and scalability.

7.3.3 Exploring a more diverse set of NN structures

While the CEG4N framework marks a significant advancement, its examination of quantization effects has mainly been limited to feedforward networks utilizing ReLU activations. However, AI includes many structures, such as recurrent NNs (RNNs), generative adversarial networks (GANs), and networks that utilize different activation functions. Future studies should strive to broaden the scope of quantization and equivalence verification techniques to encompass these diverse architectures. By delving into the specific challenges and advantages associated with each type of architecture, researchers can better understand how quantization influences model performance across various AI domains. This broader perspective will facilitate the creation of more universal and resilient quantization approaches, ensuring their broad applicability and effectiveness.

7.3.4 Investigating alternative quantization approaches

The CEG4N framework marks a notable advancement in the NN quantization domain by using a mixed-precision quantization method. This technique allows for different bit-width precision for weights, biases, and activation functions in each model layer. Such a sophisticated approach enables a more customized enhancement of computational efficiency and model effectiveness, enabling resources to be allocated strategically within the model. While CEG4N introduces innovative layer-specific quantization tactics, there is potential for additional enhancement and elaboration of these techniques.

Future research efforts could gain insights from further investigating the level of detail in mixed-precision quantization approaches, focusing on examining the diverse effects of different bit-width assignments within specific layers of an NN. It is essential to comprehend how these assignments impact the model's accuracy, memory utilization, and computational needs. Additionally, creating adaptive quantization methods that can flexibly modify bit widths based on the model's real-time performance and available resources during training or inference stages could greatly improve the feasibility and efficiency of deploying NNs in environments with limited resources.

In addition, there is a compelling case for expanding the scope of research to include innovative quantization methods that extend beyond the current search-based algorithms. Furthermore, investigating methods that solely depend on integer arithmetic may help improve the scalability of the verification process in CEG4N.

7.3.5 Robustness of quantized NNs against adversarial attacks

As AI models become increasingly integrated into real-world applications, their security against adversarial attacks emerges as a paramount concern. By altering model parameters, the quantization process could inadvertently affect a model's vulnerability

to such attacks. Future research should, therefore, prioritize the evaluation of quantized models' robustness, investigating whether quantization introduces new vulnerabilities or mitigates existing ones. This line of inquiry is crucial for deploying quantized models in security-sensitive environments, ensuring that efforts to enhance computational efficiency do not compromise model security.

7.3.6 Support for QAT techniques

Quantization-aware training represents a promising avenue for enhancing the efficacy of the quantization process, potentially improving model accuracy and efficiency by incorporating quantization considerations directly into the training process. Future research could explore integrating quantization-aware training techniques within the CEG4N framework, assessing how such techniques impact the EC process and the overall performance of quantized models. This exploration could lead to more sophisticated quantization refinement methodologies that preserve and possibly enhance model behavior post-quantization, further advancing state-of-the-art AI model deployment within resource-constrained environments.

7.4 Final thoughts

The CEG4N framework is a significant step in pursuing efficient and reliable NN quantization methods. Its emphasis on EC as a cornerstone of the quantization refinement process underscores the importance of maintaining model behavior post-quantization, especially in safety-critical applications. While challenges remain in terms of scalability and the exploration of alternative quantization strategies, this research paves the way for more sophisticated approaches to deploying AI models in resource-constrained environments. Future advancements in this domain will undoubtedly build upon the foundation laid by CEG4N, driving further innovations in AI efficiency and applicability.

References

AHAMED, K. A study on neural network architectures. *Computer Engineering and Intelligent Systems*, v. 7, p. 1–7, 2016. No citation in the text.

AMIR, G. et al. An smt-based approach for verifying binarized neural networks. *Tools and Algorithms for the Construction and Analysis of Systems*, v. 12652, p. 203 – 222, 2020. No citation in the text.

ANANTRASIRICHAI, N.; BULL, D. Artificial intelligence in the creative industries: a review. *Artificial Intelligence Review*, v. 55, p. 589–656, 2020. No citation in the text.

AVIGAD, J.; HARRISON, J. Formally verified mathematics. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 57, n. 4, p. 66–75, apr 2014. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/2591012>. No citation in the text.

BAI, J. et al. *ONNX: Open Neural Network Exchange*. [S.l.]: GitHub, 2019. No citation in the text.

BAIER, C.; KATOEN, J.-P. *Principles of Model Checking*. [S.l.]: The MIT Press, 2008. ISBN 026202649X. No citation in the text.

BAK, S.; LIU, C.; JOHNSON, T. The second international verification of neural networks competition (vnn-comp 2021): Summary and results. *ArXiv:abs/2109.00498*, 2021. No citation in the text.

BANNER, R.; NAHSHAN, Y.; SOUDRY, D. Post training 4-bit quantization of convolutional networks for rapid-deployment. In: _____. *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019. No citation in the text.

Barredo Arrieta, A. et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, v. 58, p. 82–115, 2020. ISSN 1566-2535. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1566253519308103>. No citation in the text.

BASKIN, C. et al. Uniq: Uniform noise injection for non-uniform quantization of neural networks. *arXiv: Learning*, 2018. No citation in the text.

BELLARE, M.; GOLDWASSER, S. The complexity of decision versus search. *SIAM J. Comput.*, v. 23, p. 97–119, 1991. No citation in the text.

BERTOLINI, M. et al. Machine learning for industrial applications: A comprehensive literature review. *Expert Syst. Appl.*, v. 175, p. 114820, 2021. No citation in the text.

BESSA, I. V. et al. Verification of fixed-point digital controllers using direct and delta forms realizations. *Des. Autom. Embedded Syst.*, Kluwer Academic Publishers, USA, v. 20, n. 2, p. 95–126, jun 2016. ISSN 0929-5585. Disponível em: <https://doi.org/10.1007/s10617-016-9173-5>. No citation in the text.

BHALGAT, Y. et al. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, p. 2978–2985, 2020. Disponível em: <https://api.semanticscholar.org/CorpusID:216036085>. No citation in the text.

BISHOP, C. *Pattern Recognition and Machine Learning*. Springer, 2006. Disponível em: <https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/>. No citation in the text.

BOONE, J.; SIGILLITO, V.; SHABER, G. Neural networks in radiology: an introduction and evaluation in a signal detection task. *Medical physics*, v. 17 2, p. 234–41, 1990. No citation in the text.

BOUDARDARA, F. et al. A review of abstraction methods towards verifying neural networks. *ACM Transactions on Embedded Computing Systems*, 2023. No citation in the text.

BOURECHAK, A. et al. At the confluence of artificial intelligence and edge computing in iot-based applications: A review and new perspectives. *Sensors (Basel, Switzerland)*, v. 23, 2023. No citation in the text.

BRAYTON, R.; MISHCHENKO, A. Abc: An academic industrial-strength verification tool. In: TOUILI, T.; COOK, B.; JACKSON, P. (Ed.). *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 24–40. ISBN 978-3-642-14295-6. No citation in the text.

BRUNTON, S. et al. Data-driven aerospace engineering: Reframing the industry with machine learning. *ArXiv*, abs/2008.10740, 2020. No citation in the text.

BÜNING, M. K.; KERN, P.; SINZ, C. Verifying equivalence properties of neural networks with relu activation functions. In: *Principles and Practice of Constraint Programming*. [S.l.]: Springer, 2020. (Lecture Notes in Computer Science, v. 12333), p. 868–884. No citation in the text.

CAI, Y. et al. Zeroq: A novel zero shot quantization framework. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2020. p. 13166–13175. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01318>. No citation in the text.

CHARYTANOWICZ, M. et al. Complete gradient clustering algorithm for features analysis of x-ray images. In: _____. *Information Technologies in Biomedicine*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 15–24. ISBN 978-3-642-13105-9. No citation in the text.

CHATTERJEE, A.; VARSHNEY, L. Towards optimal quantization of neural networks. *2017 IEEE International Symposium on Information Theory (ISIT)*, p. 1162–1166, 2017. No citation in the text.

CHEN, W. et al. Quantization of deep neural networks for accurate edgecomputing. *ArXiv*, abs/2104.12046, 2021. No citation in the text.

CHEN, W.; WANG, P.; CHENG, J. Towards mixed-precision quantization of neural networks via constrained optimization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, p. 5330–5339, 2021. No citation in the text.

CHEN, Y. et al. Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions. *ACM Comput. Surv.*, Association for Computing Machinery, New York, NY, USA, v. 53, n. 4, aug 2020. ISSN 0360-0300. Disponível em: <https://doi.org/10.1145/3398209>. No citation in the text.

CHENG, C.-H.; NüHRENBERG, G.; RUESS, H. Verification of binarized neural networks. *ArXiv*, abs/1710.03107, 2017. No citation in the text.

CHENG, Y. et al. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, v. 35, n. 1, p. 126–136, 2018. No citation in the text.

CHOUKROUN, Y.; KRAVCHIK, E.; KISILEV, P. Low-bit quantization of neural networks for efficient inference. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, p. 3009–3018, 2019. No citation in the text.

CHU, W.; BOSE, N. Vector quantization of neural networks. *IEEE transactions on neural networks*, v. 9 6, p. 1235–45, 1998. No citation in the text.

CLARKE, E. M.; GRUMBERG, O.; PELED, D. A. *Model checking*. London, Cambridge: MIT Press, 1999. ISBN 0-262-03270-8. No citation in the text.

CLARKE, E. M. et al. *Handbook of Model Checking*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2018. ISBN 3319105744. No citation in the text.

CORDEIRO, L.; FISCHER, B.; MARQUES-SILVA, J. Smt-based bounded model checking for embedded ansi-c software. *IEEE Transactions on Software Engineering*, v. 38, n. 4, p. 957–974, 2012. No citation in the text.

CUN, Y. L.; DENKER, J. S.; SOLLA, S. A. Optimal brain damage. In: _____. *Advances in Neural Information Processing Systems 2*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990. p. 598–605. ISBN 1558601007. No citation in the text.

DAS, S. et al. Recurrent neural networks (rnns): Architectures, training tricks, and introduction to influential research. In: _____. *Machine Learning for Brain Disorders*. New York, NY: Springer US, 2023. p. 117–138. ISBN 978-1-0716-3195-9. Disponível em: <https://doi.org/10.1007/978-1-0716-3195-9_4>. No citation in the text.

DETHISE, A.; CANINI, M.; NARODYTSKA, N. Analyzing learning-based networked systems with formal verification. *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, p. 1–10, 2021. No citation in the text.

EGMONT-PETERSEN, M.; RIDDER, D.; HANDELS, H. Image processing with neural networks - a review. *Pattern Recognit.*, v. 35, p. 2279–2301, 2002. No citation in the text.

EHLERS, R. Formal verification of piece-wise linear feed-forward neural networks. p. 269–286, 2017. No citation in the text.

ELEFTHERIADIS, C. et al. On neural network equivalence checking using SMT solvers. In: *Formal Modeling and Analysis of Timed Systems*. Cham: Springer, 2022. p. 237–257. ISBN 978-3-031-15839-1. No citation in the text.

FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, v. 7, p. 179–188, 1936. No citation in the text.

GADELHA, M. R. et al. Esbmc 5.0: An industrial-strength c model checker. In: *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. [S.l.: s.n.], 2018. p. 888–891. No citation in the text.

GAO, Q.; XU, X. The analysis and research on computational complexity. *The 26th Chinese Control and Decision Conference (2014 CCDC)*, p. 3467–3472, 2014. No citation in the text.

GAO, Y. et al. Fast non-uniform quantization of neural networks. *2022 7th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, p. 8–12, 2022. No citation in the text.

GARG, S. et al. Confounding tradeoffs for neural network quantization. *ArXiv*, abs/2102.06366, 2021. No citation in the text.

GEHR, T. et al. Ai2: Safety and robustness certification of neural networks with abstract interpretation. *2018 IEEE Symposium on Security and Privacy (SP)*, p. 3–18, 2018. No citation in the text.

GHANEM, M.; SINIORA, D. On theoretical complexity and boolean satisfiability. *ArXiv*, abs/2112.11769, 2021. No citation in the text.

GHOLAMI, A. et al. A survey of quantization methods for efficient neural network inference. In: *Low-Power Computer Vision*. [S.l.]: Chapman and Hall/CRC, 2022. p. 291–326. No citation in the text.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: The MIT Press, 2016. ISBN 0262035618. No citation in the text.

GORDON, M. J. C.; MELHAM, T. F. *Introduction to HOL: a theorem proving environment for higher order logic*. USA: Cambridge University Press, 1993. ISBN 0521441897. No citation in the text.

GUO, Y. A survey on methods and theories of quantized neural networks. *ArXiv*, abs/1808.04752, 2018. No citation in the text.

HAN, S.; MAO, H.; DALLY, W. J. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: *4th International Conference on Learning Representations*. [S.l.: s.n.], 2016. No citation in the text.

HAN, S. et al. Learning both weights and connections for efficient neural networks. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 1135–1143. No citation in the text.

HAPPEL, B. L. M.; MURRE, J. Design and evolution of modular neural network architectures. *Neural Networks*, v. 7, p. 985–1004, 1994. No citation in the text.

HARRISON, J. *Handbook of Practical Logic and Automated Reasoning*. 1st. ed. USA: Cambridge University Press, 2009. ISBN 0521899575. No citation in the text.

HENDRYCKS, D.; DIETTERICH, T. G. Benchmarking neural network robustness to common corruptions and perturbations. *ArXiv:abs/1903.12261*, 2019. No citation in the text.

HENZINGER, T.; LECHNER, M.; ZIKELIC, D. Scalable verification of quantized neural networks (technical report). p. 3787–3795, 2020. No citation in the text.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. No citation in the text.

HOFER-SCHMITZ, K.; STOJANOVIć, B. Towards formal verification of iot protocols: A review. *Comput. Networks*, v. 174, p. 107233, 2020. No citation in the text.

HUANG, X. et al. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, Elsevier BV, v. 37, p. 100270, ago. 2020. ISSN 1574-0137. Disponível em: <http://dx.doi.org/10.1016/j.cosrev.2020.100270>. No citation in the text.

HUANG, X. et al. Safety verification of deep neural networks. In: MAJUMDAR, R.; KUNČAK, V. (Ed.). *Computer Aided Verification*. Cham: Springer International Publishing, 2017. p. 3–29. ISBN 978-3-319-63387-9. No citation in the text.

HUBARA, I. et al. Quantized neural networks: Training neural networks with low precision weights and activations. *ArXiv*, abs/1609.07061, 2016. No citation in the text.

HUBARA, I. et al. Quantized neural networks: Training neural networks with low precision weights and activations. *ArXiv:abs/1609.07061*, 2017. No citation in the text.

JACOB, B. et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, p. 2704–2713, 2017. Disponível em: <https://api.semanticscholar.org/CorpusID:39867659>. No citation in the text.

JIN, Q.; YANG, L.; LIAO, Z. A. Towards efficient training for neural network quantization. *ArXiv:abs/1912.10207*, 2019. No citation in the text.

JULIAN, K. D. et al. Policy compression for aircraft collision avoidance systems. In: *DASC*. [S.l.: s.n.], 2016. p. 1–10. No citation in the text.

KABRA, R. et al. Automated content generation system using neural text generation. In: JACOB, I. J.; SHANMUGAM, S. K.; BESTAK, R. (Ed.). *Data Intelligence and Cognitive Informatics*. Singapore: Springer Nature Singapore, 2022. p. 821–829. ISBN 978-981-16-6460-1. No citation in the text.

KATZ, G. et al. Reluplex: An efficient SMT solver for verifying deep neural networks. In: *International Conference on Computer Aided Verification*. [S.l.: s.n.], 2017. No citation in the text.

KHEDR, H.; FERLEZ, J.; SHOUKRY, Y. Effective formal verification of neural networks using the geometry of linear regions. *ArXiv*, abs/2006.10864, 2020. No citation in the text.

KIMHI, M. et al. Fbm: Fast-bit allocation for mixed-precision quantization. *ArXiv*, abs/2205.15437, 2022. No citation in the text.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, v. 60, p. 84–90, 2012. No citation in the text.

KRUSE, R. et al. Introduction to neural networks. p. 9–13, 2016. No citation in the text.

LAHAV, O.; KATZ, G. Pruning and slicing neural networks using formal verification. *2021 Formal Methods in Computer Aided Design (FMCAD)*, p. 1–10, 2021. No citation in the text.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Springer Science and Business Media LLC, v. 521, n. 7553, p. 436–444, maio 2015. ISSN 1476-4687. Disponível em: <http://dx.doi.org/10.1038/nature14539>. No citation in the text.

LECUN, Y.; CORTES, C. *The MNIST database of handwritten digits*. 2005. No citation in the text.

LEE, K.-H.; JEONG, J.; BAE, S.-H. An inter-layer weight prediction and quantization for deep neural networks based on a smoothly varying weight hypothesis. *ArXiv*, abs/1907.06835, 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:196831335>. No citation in the text.

LI, Z. et al. Psaq-vit v2: Toward accurate and general data-free quantization for vision transformers. *IEEE Transactions on Neural Networks and Learning Systems*, v. 35, p. 17227–17238, 2022. Disponível em: <https://api.semanticscholar.org/CorpusID:252211787>. No citation in the text.

LI, Z. et al. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, v. 33, p. 6999–7019, 2020. No citation in the text.

LIANG, T. et al. Pruning and quantization for deep neural network acceleration: A survey. *ArXiv*, abs/2101.09671, 2021. No citation in the text.

LIN, D. D.; TALATHI, S. S.; ANNAPUREDDY, V. S. Fixed point quantization of deep convolutional networks. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. [S.l.]: JMLR.org, 2016. (ICML'16), p. 2849–2858. No citation in the text.

LIU, J.; CAI, J.; ZHUANG, B. Sharpness-aware quantization for deep neural networks. *ArXiv*, abs/2111.12273, 2021. No citation in the text.

LIU, Z. et al. Evolutionary quantization of neural networks with mixed-precision. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 2785–2789, 2021. No citation in the text.

LOHAR, D. et al. Sound mixed fixed-point quantization of neural networks. *ACM Trans. Embed. Comput. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 5s, sep 2023. ISSN 1539-9087. Disponível em: <https://doi.org/10.1145/3609118>. No citation in the text.

LOUI, M. Computational complexity theory. *ACM Comput. Surv.*, v. 28, p. 47–49, 1996. No citation in the text.

LOVASZ, L. et al. Search problems in the decision tree model. In: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*. [S.l.: s.n.], 1991. p. 576–585. No citation in the text.

MAIER, H.; DANDY, G. Neural networks for the prediction and forecasting of water resource variables: a review of modelling issues and applications. *Environ. Model. Softw.*, v. 15, p. 101–124, 2000. No citation in the text.

MARINÓ, G. C. et al. Deep neural networks compression: A comparative survey and choice recommendations. *Neurocomputing*, v. 520, p. 152–170, 2023. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231222014643>. No citation in the text.

MATOS, J. a. B. P. et al. Counterexample guided neural network quantization refinement. *Trans. Comp.-Aided Des. Integ. Cir. Sys.*, IEEE Press, v. 43, n. 4, p. 1121–1134, abr. 2024. ISSN 0278-0070. Disponível em: <https://doi.org/10.1109/TCAD.2023.3335313>. No citation in the text.

MATOS, J. B. P. et al. Ceg4n: Counter-example guided neural network quantization refinement. In: ISAC, O. et al. (Ed.). *Software Verification and Formal Methods for ML-Enabled Autonomous Systems*. Cham: Springer International Publishing, 2022. p. 29–45. ISBN 978-3-031-21222-2. No citation in the text.

MENDIAS, J. et al. Efficient verification of scheduling, allocation and binding in high-level synthesis. *Proceedings Euromicro Symposium on Digital System Design. Architectures, Methods and Tools*, p. 308–315, 2002. No citation in the text.

MISHRA, R.; GUPTA, H. P.; DUTTA, T. A survey on deep neural network compression: Challenges, overview, and solutions. *ArXiv*, abs/2010.03954, 2020. Disponível em: <https://api.semanticscholar.org/CorpusID:222208626>. No citation in the text.

MONTEIRO, F. R.; GADELHA, M. R.; CORDEIRO, L. C. Model checking c++ programs. *Software Testing, Verification and Reliability*, v. 32, n. 1, p. e1793, 2022. No citation in the text.

MUNAKATA, S. et al. Towards formal repair and verification of industry-scale deep neural networks. *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, p. 360–364, 2023. No citation in the text.

NAGEL, M. et al. Data-free quantization through weight equalization and bias correction. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, p. 1325–1334, 2019. Disponível em: <https://api.semanticscholar.org/CorpusID:184487878>. No citation in the text.

NAGEL, M. et al. A white paper on neural network quantization. *ArXiv:abs/2106.08295*, 2021. No citation in the text.

NARODYTSKA, N. Formal verification of deep neural networks. *2018 Formal Methods in Computer Aided Design (FMCAD)*, p. 1–1, 2018. No citation in the text.

NGUYEN, H. D.; ALEXANDRIDIS, A.; MOUCHTARIS, A. Quantization aware training with absolute-cosine regularization for automatic speech recognition. In: *Interspeech*. [s.n.], 2020. Disponível em: <https://api.semanticscholar.org/CorpusID:226203265>. No citation in the text.

NIPKOW, T.; PAULSON, L. C.; WENZEL, M. A proof assistant for higher-order logic. *Lecture Notes in Computer Science*, 2002. Disponível em: <https://api.semanticscholar.org/CorpusID:59771319>. No citation in the text.

OBERKAMPF, W.; TRUCANO, T. Verification and validation in computational fluid dynamics. *Progress in Aerospace Sciences*, v. 38, p. 209–272, 2002. No citation in the text.

PALIWAL, M.; KUMAR, U. A. Neural networks and statistical techniques: A review of applications. *Expert Syst. Appl.*, v. 36, p. 2–17, 2009. No citation in the text.

PARK, E.; AHN, J.; YOO, S. Weighted-entropy-based quantization for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 7197–7205, 2017. No citation in the text.

PARK, S. et al. Quantized memory-augmented neural networks. *ArXiv*, abs/1711.03712, 2017. No citation in the text.

PASZKE, A. et al. Pytorch: An imperative style, high-performance deep learning library. In: _____. *Advances in Neural Information Processing Systems 32*. [S.l.]: Curran Associates, Inc., 2019. p. 8024–8035. No citation in the text.

PAULSEN, B.; WANG, J.; WANG, C. Reludiff: Differential verification of deep neural networks. In: *ISCE*. [S.l.]: Association for Computing Machinery, 2020. p. 714–726. No citation in the text.

PAULSEN, B. et al. NEURODIFF: Scalable differential verification of neural networks using fine-grained approximation. *35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, p. 784–796, 2020. No citation in the text.

PENG, J. et al. Cmq: Crossbar-aware neural network mixed-precision quantization via differentiable architecture search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 41, n. 11, p. 4124–4133, 2022. No citation in the text.

POURANSARI, H.; TUZEL, O. Least squares binary quantization of neural networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, p. 2986–2996, 2020. No citation in the text.

RAMALHO, M. et al. Smt-based bounded model checking of c++ programs. In: *2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*. [S.l.: s.n.], 2013. p. 147–156. No citation in the text.

REZK, N. M. et al. Mohaq: Multi-objective hardware-aware quantization of recurrent neural networks. *J. Syst. Archit.*, Elsevier North-Holland, Inc., USA, v. 133, n. C, dec 2022. ISSN 1383-7621. Disponível em: <https://doi.org/10.1016/j.sysarc.2022.102778>. No citation in the text.

RUAN, W.; HUANG, X.; KWIATKOWSKA, M. Reachability analysis of deep neural networks with provable guarantees. In: *International Joint Conference on Artificial Intelligence*. [s.n.], 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:19173163>. No citation in the text.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson Education, 2021. ISBN 9781292401171. Disponível em: <https://books.google.pt/books?id=cb0qEAAAQBAJ>. No citation in the text.

SAJI, S. A.; AGRAWAL, S.; SOOD, S. Formal verification of deep neural networks in hardware. *2022 IEEE Women in Technology Conference (WINTECHCON)*, p. 1–6, 2022. No citation in the text.

SELIGMAN, E.; SCHUBERT, T.; KUMAR, M. K. Formal equivalence verification. p. 225–259, 2015. No citation in the text.

SENA, L. et al. Verifying quantized neural networks using SMT-based model checking. *ArXiv:abs/2106.05997*, 2021. No citation in the text.

SENNRICH, R.; HADDOW, B.; BIRCH, A. Neural machine translation of rare words with subword units. *ArXiv*, abs/1508.07909, 2015. Disponível em: <https://api.semanticscholar.org/CorpusID:1114678>. No citation in the text.

SESHIA, S. et al. Formal specification for deep neural networks. p. 20–34, 2018. No citation in the text.

SHIH, A.; DARWICHE, A.; CHOI, A. Verifying binarized neural networks by angluin-style learning. p. 354–370, 2019. No citation in the text.

SILVER, D. et al. Mastering the game of go with deep neural networks and tree search. *Nature*, v. 529, p. 484–489, 2016. Disponível em: <https://api.semanticscholar.org/CorpusID:515925>. No citation in the text.

SONG, X. et al. QNNVerifier: A tool for verifying neural networks using SMT-based model checking. *ArXiv:abs/2111.13110*, 2021. No citation in the text.

STURSBERG, O. et al. Verification of a cruise control system using counterexample-guided search. *Control Engineering Practice*, v. 12, p. 1269–1278, 2004. No citation in the text.

SUN, B. et al. Quantized synchronization of memristive neural networks with time-varying delays via super-twisting algorithm. *Neurocomputing*, v. 380, p. 133–140, 2020. No citation in the text.

SZE, V. et al. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, v. 105, n. 12, p. 2295–2329, 2017. No citation in the text.

TEUBER, S. et al. Geometric path enumeration for equivalence verification of neural networks. *in ICTAI*, IEEE, Nov 2021. No citation in the text.

TRAN, H.-D. et al. Star-based reachability analysis of deep neural networks. In: *FM*. [S.l.: s.n.], 2019. No citation in the text.

TRAN, H.-D. et al. Star-based reachability analysis of deep neural networks. p. 670–686, 2019. No citation in the text.

UMUROGLU, Y.; JAHRE, M. Streamlined deployment for quantized neural networks. *ArXiv*, abs/1709.04060, 2017. No citation in the text.

URBAN, C.; MIN'E, A. A review of formal methods applied to machine learning. *ArXiv*, abs/2104.02466, 2021. No citation in the text.

WANG, K. et al. Haq: Hardware-aware automated quantization with mixed precision. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 8604–8612, 2018. Disponível em: <https://api.semanticscholar.org/CorpusID:102350477>. No citation in the text.

WANG, X. et al. Towards formal verification of neural networks: A temporal logic based framework. p. 73–87, 2019. No citation in the text.

WANG, Y.; MA, Z.; YANG, C. A new mixed precision quantization algorithm for neural networks based on reinforcement learning. *2023 IEEE 6th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, p. 1016–1020, 2023. No citation in the text.

WANG, Y. et al. Quantized control for extended dissipative synchronization of chaotic neural networks: A discretized lkf method. *ISA transactions*, 2021. No citation in the text.

WANG, Z. et al. Evolutionary multi-objective model compression for deep neural networks. *IEEE Computational Intelligence Magazine*, v. 16, n. 3, p. 10–21, 2021. No citation in the text.

WEBB, S. et al. Statistical verification of neural networks. *arXiv: Machine Learning*, 2018. No citation in the text.

WEISS, R. et al. Applications of neural networks in biomedical data analysis. *Biomedicines*, v. 10, 2022. No citation in the text.

WONG, B. K.; SELVI, Y. Neural network applications in finance: A review and analysis of literature (1990-1996). *Inf. Manag.*, v. 34, p. 129–139, 1998. No citation in the text.

WU, H. et al. Research on artificial intelligence enhancing internet of things security: A survey. *IEEE Access*, v. 8, p. 153826–153848, 2020. No citation in the text.

WU, H. et al. Integer quantization for deep learning inference: Principles and empirical evaluation. *ArXiv*, abs/2004.09602, 2020. Disponível em: <https://api.semanticscholar.org/CorpusID:216035831>. No citation in the text.

XU, C. et al. Alternating multi-bit quantization for recurrent neural networks. *ArXiv*, abs/1802.00150, 2018. No citation in the text.

XU, X. et al. Quantized graph neural networks for image classification. *Mathematics*, 2023. No citation in the text.

XU, Y. et al. Deep neural network compression with single and multiple level quantization. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. [S.l.]: AAAI Press, 2018. (AAAI'18/IAAI'18/EAAI'18). ISBN 978-1-57735-800-8. No citation in the text.

YANG, H. et al. Neural network quantization based on model equivalence. *2022 International Conference on High Performance Big Data and Intelligent Systems (HDIS)*, p. 8–12, 2022. No citation in the text.

YANG, J. et al. Quantization networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 7300–7308, 2019. No citation in the text.

YANG, P. et al. Enhancing robustness verification for deep neural networks via symbolic propagation. *Form. Asp. Comput.*, Springer-Verlag, Berlin, Heidelberg, v. 33, n. 3, p. 407–435, jun 2021. ISSN 0934-5043. Disponível em: <https://doi.org/10.1007/s00165-021-00548-1>. No citation in the text.

YANG, Y.-Y. et al. A closer look at accuracy vs. robustness. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2020. (NIPS'20). ISBN 9781713829546. No citation in the text.

ZHANG, J.; ZHOU, Y.; SAAB, R. Post-training quantization for neural networks with provable guarantees. *SIAM Journal on Mathematics of Data Science*, v. 5, n. 2, p. 373–399, 2023. No citation in the text.

ZHANG, W. et al. Sentiment analysis in the era of large language models: A reality check. *ArXiv*, abs/2305.15005, 2023. Disponível em: <https://api.semanticscholar.org/CorpusID:258866189>. No citation in the text.

ZHANG, Y.; SONG, F.; SUN, J. QEBVerif: Quantization error bound verification of neural networks. In: *Computer Aided Verification*. Cham: Springer Nature Switzerland, 2023. p. 413–437. ISBN 978-3-031-37703-7. No citation in the text.

ZHANG, Y. et al. Qvip: An ilp-based formal verification approach for quantized neural networks. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022. No citation in the text.

ZHOU, A. et al. Incremental network quantization: Towards lossless cnns with low-precision weights. In: *5th International Conference on Learning Representations*. [S.l.]: OpenReview.net, 2017. No citation in the text.

ZHOU, M. et al. Progress in neural nlp: Modeling, learning, and reasoning. *Engineering*, v. 6, n. 3, p. 275–290, 2020. ISSN 2095-8099. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2095809919304928>. No citation in the text.